UNIVERSITY OF SOUTHERN QUEENSLAND

Development of a Decision Support System for Furrow and Border Irrigation.

A Dissertation submitted by
David McClymont, B Eng (hons)

For the award of
Doctor of Philosophy

2007

Abstract

Furrow and border irrigation practices in Australia and around the world are typically inefficient. Recent advances in computer-based surface irrigation decision support technology have the potential to improve performance, but have had little uptake. Despite considerable academic achievements with individual components of the technology, the implementation of this knowledge into usable tools has been immature, hindering adoption. In particular, there has been little progress in encapsulating the different decision support components into a standalone system for surface irrigation. Therefore, the research problem addressed in this dissertation aims to develop a new decision support system for furrow and border irrigation aimed at increasing the usability of the technology, and improving decision making capabilities. Specifically the research hypothesis is:

"That calibration, optimisation, and parameter analysis capabilities can be developed and integrated with an accurate and robust simulation model into a decision support system to improve furrow and border irrigation performance."

Six research objectives have been identified to support the hypothesis including: *(RO1)* investigate existing surface irrigation modelling technology to determine a model and solution technique structure suitable for incorporating into a decision support system; *(RO2)* develop a robust reliable simulation engine for furrow and border irrigation for automation within a decision support system under optimisation and systematic response evaluation; *(RO3)* investigate and develop parameter estimation (calibration) capabilities for the decision support system; *(RO4)* investigate and develop optimisation capabilities for the decision support system; *(RO5)* investigate and develop parameter response (design charts) capabilities for the decision support system; and *(RO6)* develop an object-oriented framework to combine the components developed in Research Objectives 2 to 5 with data management facilities and a graphical user interface.

Successful completion of these objectives has resulted in the development of a decision support system for furrow and border irrigation featuring an automation-capable hydrodynamic simulation engine, automated full-hydrodynamic inverse solution, automated optimisation of design and management variables, and automated user-definable real-time generation of system response. This was combined with a highly flexible object-oriented program structure and web-browser-like graphical user interface. Each of these components represents a unique implementation of the required functionalities, differing from the established software packages (such as *SIRMOD* and *WinSRFR*) that use alternate technologies with no automation or optimisation capabilities.

Development of the hydrodynamic simulation engine has involved the refinement of the commonly used implicit double-sweep methodology with the objectives of achieving robustness and reliability under automation. It was subsequently found that only subtle changes and manipulations were required in much of the numerical methodology, including derivation of simplified solution equations. The main focus of this research has targeted the computational algorithms that drive the numerical solution process. Key factors effecting robustness and reliability

were identified in a study of simulation operation, and treated through these algorithms. Validation was undertaken against output from the *SIRMOD* simulation engine, with robustness and reliability tested through tens of thousands of simulations under optimisation and automated system response evaluation.

The calibration facilities demonstrated that the inverse-solution using the full-hydrodynamic model is a viable and robust methodology for the unique identification of up to three infiltration/roughness parameters. Two optimisation-methods were investigated during this research with objective-functions based upon either a volume-balance time-of-advance equation, or complete simulations of the hydrodynamic model. A simple but robust optimisation algorithm was designed for this purpose. While the volume-balance method proved fast and reliable, its accuracy is reduced due to the underlying assumptions and simplistic model structure. The hydrodynamic method was shown to be accurate, although it suffered slow execution times. It was therefore decided to use the two methods in tandem during the solution process where the faster volume-balance method is used to provide starting estimates for the more accurate hydrodynamic method. Response-surface investigation for the advance-based objective function identified a unique solution when solving for three parameters.

It was found that the automated unconstrained optimisation of design and management practices is limited to the selection of one solution variable (time to cut-off) due to non-unique multi-variable solutions. Nevertheless, the developed facilities provide a unique benchmarking of irrigation performance potential. This research has used the earlier-developed optimisation algorithm to automate simulations using a prototype objective-function based upon user-defined weightings of key performance measures. A study of the response-surfaces of different configurations of the objective-function identified parabolic ridges of alternate solutions, so, in practice, the optimisation process simplifies down to optimising only one parameter: time-to-cutoff. It was also recognized that the performance-based objective functions are highly sensitive to numerical discretisation inconsistencies that occur between simulations, which impede solution convergence.

The highly customisable, automated, system response evaluation facilities developed in this research offer potential as both a research and practitioner tool, capable of multidimensional analysis of irrigation systems subject to temporal and spatial infiltration variations. A preliminary study demonstrated the importance of infiltration variation on irrigation decision-making, and provided initial guideline layout designs that combined the effects of variable infiltration and three decision variables using a fixed management strategy of minimising runoff. A limited range of response outputs for a fixed management objective negated the potential benefit of visualising a large number of dimensions. Nevertheless, this study provided direction for the subsequent software development with recommendations including: representing system outputs as contours and iso-curves, rather than by the chart axes; representing different infiltration conditions in separate design charts; allowing the user to assign variables to each chart axis; and representing only two decision variables in each chart.

Finally, the simulation, calibration, optimisation and parameter analysis components were combined with a database and graphical user interface to develop the *FIDO* (Furrow Irrigation Decision Optimiser) decision support system. There were three focus areas during this marriage of components; *firstly*, an object-oriented structure was developed to accommodate program elements concentrating on separating the graphical user interface components from other task related objects for flexible future development; secondly, a database was developed using *XML*-based technologies to store property, paddock, event and model information; and *thirdly*, a user-friendly graphical user interface was created with web-browser-like functionality. The software design evolved through many different prototypes with its current design being heavily influenced from the successes and mistakes of the previous attempts.

This work represents the first coordinated attempt to develop a decision support system for furrow irrigation linking a database, simulation engine, calibration facilities, optimisation facilities, and parameter analysis capabilities. A major feature of this work is that all components of the system have been developed from first principles using an object-oriented structure, with the primary goal of implementation into a decision support system. This research has contributed to the development of a professional-quality software package to improve the decision-making capabilities of researchers, irrigation consultants, and irrigators.

Certification of dissertation

I certify that the ideas, experimental work, results, analyses, software and conclusions reported in this dissertation are entirely my own effort, except were otherwise acknowledged. I also certify that the work is original and has not been previously submitted for any other award, except where otherwise acknowledged.

_____ _____

Signature of Candidate Date

ENDORSEMENT

_____ _____

Signature of Supervisor Date

_____ _____

Signature of Supervisor Date

Acknowledgements

I shall take this opportunity to thank and acknowledge those who provided valuable assistance and inspiration throughout the duration of this research.

Firstly, I am particularly grateful to Professors Rod Smith and Steve Raine for their forward vision with the irrigation research being conducted at USQ. Rod stimulated my initial research efforts and taught me the concepts of researching and presenting the work. His background in hydraulic and irrigation engineering was regularly tested whenever technical problems arose in refining the simulation engine. Steve has provided the practical guidelines for developing the decision support system, and continually managed to lift my enthusiasm for the project when other work was dominating.

I would like to thank Professor Wynn Walker of Utah State University for introducing me to Borland C++ Builder, without which, this project (and my career) would have had an entirely different outcome. Professor Walker was also gracious in providing the source code of his own software *SIRMOD* for us to review and study in the early stages of this project.

In addition, my mother Mrs Grace McClymont has been a continual source of inspiration, support and encouragement.

Finally I would thank my loving friend and wife Hoda for her patience, love and support while enduring significant absence of my time over the eight years we have been together during development of this dissertation.

I would like to dedicate this dissertation to my late father, Mr “Bill” McClymont, who never had a formal education but has taught me more in life than anyone else, and who would be thrilled beyond words to see his son submit this dissertation.

Contributions to theory and practice from this research

This dissertation has covered a wide range of topic areas in developing a decision support system for furrow and border irrigation. A summary of the major contributions to theory and practice arising from this research include:

- Development of a robust, reliable and flexible hydrodynamic simulation engine for furrow and border irrigation for automation within a decision support system under optimisation and systematic response evaluation. This includes:
    - Refinement of existing simulation technologies including the simplification of the algebraic equations for the Preismann double-sweep solution technique;
    - Development of an object-oriented structure to simplify operation and improve flexibility and future development; and
    - Identification and solution to convergence problems associated with hydrodynamic modelling.
- Development of an automated hydrodynamic inverse solution technique for estimating soil infiltration (and roughness) parameters. This includes:
    - Development of a simple reliable optimisation algorithm;
    - Development of a simple volume-balance-based inverse technique for calibrating with a minimum of field data, and for generating initial parameter estimates for the hydrodynamic solution; and
    - Investigation of parameter response-surfaces for the inverse problem identifying a unique solution when estimating three infiltration parameters.
- Development of an automated optimisation technique for design and management practices, with an emphasis on quantifying the potential of irrigation performance. This includes:
    - Development of a user-defined objective-function for optimisation of irrigation design and management practices;
    - A study of the system response for optimising irrigation performance identifying that an unlimited range of management options exists to achieve an optimum level of performance;
    - The subsequent recommendation of optimisation on only one design or management parameter due to the non-unique solution and noise in the system response; and
    - Provides and automated facility for benchmarking the performance potential of an irrigation.
- Development of an automated guideline generation facility for design and management of surface irrigation, and system response evaluation, based on repeated runs of the hydrodynamic simulation. This includes:
    - Evaluation of a range of alternative design and management guidelines that emphasise the sensitivity of temporal variations in infiltration; and
    - Development of highly configurable user interface tools for selection of design and management variables/parameters, configuration of charts, and filtering of results.
- Encapsulation of these technologies within a user-friendly, and highly automated decision support system. This includes:

- Development of a four-level database for property, paddock, event and simulation data, and for monitoring temporal and spatial system changes;
- Development of an object-oriented program structure focusing on adaptability for future enhancements and inclusions; and
- Development of a simple to use web-browser-like graphical user interface, emphasising progressive-disclosure concepts, to help improve adoption of the technology to more users.

This research has aided in the development of a new decision support system for furrow and border irrigation, which will serve as both a practical tool, and a research platform for many years to come.

Publications arising from this research

This dissertation is the culmination of ten years of research. The majority of the research was undertaken in the first four years until work and family commitments intervened and very little output was generated for a period of four to five years. Publications produced during the initial years of this research include:

McClymont, DJ & Smith, RJ 1996, 'Infiltration parameters from optimisation on furrow irrigation advance data', *Irrigation Science*, 17(1): 15-22.

Smith, RJ & McClymont, DJ 1996, 'Toward real time control of surface irrigation: Estimation of soil infiltration parameters under surface and surge irrigation', *Proc. AgEng96, the European Conference on Agricultural Engineering*, Madrid, Sept 1996.

McClymont, DJ 1996, 'A simple reliable nonlinear optimisation /solution technique suitable for agricultural applications', *Proc. of Conference on Engineering in Agriculture and Food Processing* 24-27 November 1996, University of Queensland, Gatton College. SEAg 96/063

Smith, RJ & McClymont, DJ 1996, 'Surge irrigation infiltration parameters from surge advance data', *Proc. of Conference on Engineering in Agriculture and Food Processing* 24-27 November 1996, University of Queensland, Gatton College. SEAg 96/051

McClymont, DJ, Raine, SR & Smith, RJ 1996, 'The prediction of furrow irrigation performance using the surface irrigation model SIRMOD', *Proc. of Irrigation Australia Conference and Exhibition - Australian Solutions*, May 14-16 1996, Adelaide Convention & Exhibition Centre South Australia

Smith, RJ & McClymont, DJ 1996, 'Towards real time control of surface irrigation: Estimation of soil infiltration parameters under surface irrigation and surge irrigation', *Proc. of Irrigation Australia Conference and Exhibition - Australian Solutions,* May 14-16 1996, Adelaide Convention & Exhibition Centre South Australia

Smith, RS, Raine, SR, & McClymont, DJ 1997, 'Design for improved efficiency of surface irrigation applications: A best management practice', *Proc. AWWA Regional Conference*

Raine, SR, McClymont, DJ, & Smith, RJ 1997, 'The development of guidelines for surface irrigation in areas with variable infiltration', *Proc. Aust. Soc. Sugar Cane Technologists*, 29 April-1st May, Cairns.

McClymont, DJ, Smith, R.J. & Raine, SR 1998, 'An integrated numerical model for design and management of surface irrigation', *Proc. International Conference on Engineering in Agriculture*, 27-30 September, Perth. Paper 98/039.

McClymont, DJ, Smith, R.J. & Raine, SR 1999, 'An integrated numerical model for the design and management of surface irrigation', *Proc. International Conference on Multi-Objective Decision Support Systems*, 1-6 August, Brisbane.

It is expected that several new papers will also be generated from the output of this dissertation and published in the near future.

Table of contents

List of figures

List of tables

List of symbols

$\sigma_1, \sigma_2, \rho_1, \rho_2$	Furrow shape parameters
a_i, b_i, c_i, d_i, e_i	Partial derivative terms of residual of continuity
p_i, q_i, r_i, s_i, u_i	Partial derivative terms of residual of momentum
σ_z	(Dimensionless) is subsurface storage shape parameter
f_o	(m^3/min/m) is the steady-state or basic infiltration rate for the soil
Q_{in}	Flow rate (m^3/sec)
$Z_{t,0}$	Infiltration depth at the top end of the furrow (m)
V_I	Volume of water infiltrated (m^3)
V_S	Volume of water temporarily stored on the surface. (m^3)
R_C	Residual of continuity
R_M	Residual of momentum
ϕ	Space-averaging coefficient
σ_y	Surface storage shape parameter
θ	Time-averaging coefficient.
t_c	Time-to-cutoff (min)
Z_{req}	Z-required (m)
$\Delta R_{C1}^{\ n}$, $\Delta R_{M1}^{\ n}$	Difference matrices
$T_{(1)}$ *to* $T_{(7)}$	Temporary variables used in double sweep calculation
$\nabla R_{C1}^{\ n}$, $\nabla R_{M1}^{\ n}$	Jacobian matrices
A_o.	Average cross-sectional area of surface water at the upstream end of the furrow or bay
A	Cross-sectional area of flow (m^2)
a	Kostiakov infiltration exponent (dimensionless)
AE	Application efficiency (%)
$A_{t,x}$	Cross-sectional area of flow at node (m^2)
AU	Application uniformity (%)
C	Deferred infiltration volume (m^3)

D	Drag force (friction force)
DPV	Deep percolation volume (m^3)
dt	Simulation time step (sec)
E, H and F	Auxiliary coefficients used in double sweep calculation
f_o	Steady state infiltration rate ($m^3/s/m$)
g	Gravitational constant
h'	Suction at the wetting front
i	Cell index
I	Infiltration rate ($m^3/s/m$)
i_l	Saturated hydraulic conductivity of restricting layer
J, M, L and R	Grid cell subscripts
K	Hydraulic conductivity in the wetted zone
k	Kostiakov infiltration equation coefficient ($m^3/min^a/m$)
L	Field-length (m)
n	Manning roughness parameter
OFV	Objective function value
P	Hydrostatic pressure (N/m^2)
p	Power advance curve multiplier
q	Inflow rate per unit width ($m^3/sec/m$)
Q,	Flowrate (m^3/sec)
r	Power advance curve exponent (dimensionless)
RV	Runoff volume (m^3)
S	Volume of cracks per unit area
S_0	Furrow slope
SE	Storage efficiency (%)
S_f	Energy slope
SSE	Sum square error
t	Irrigation time (mins)
t_{op}	Opportunity time for infiltration to occur (min)
TV	Total applied volume (m^3)
U,V, Z, and W:	Auxiliary coefficients used in double sweep calculation
v	Velocity of the surface flow (m/s)
VBE	Volume-balance error (%)

w_1,w_2,w_3,w_4	Objective function weighting coefficients that add up to 1.0
x	Distance along the furrow (m)
$X_{t,x}$	Node location (m)
y	Depth of flow (m)
Z	Cumulative infiltration (m)
Z	Infiltrated volume/unit length (m^3/m)
z	Rate of infiltration per unit length ($m^3/sec/m$)
$z(dt)$	Incremental infiltrated volume for the time-step (m)
Z_c	Initial depth of infiltration required for crack filling and absorption (m)
$Z_{t,x}$	Cumulative infiltration depth at node (m)
γ	Cumulative infiltration at t_{op}=0 if steady state had been reached at t_{op}=0 (m)
θ_i	Initial moisture content
θ_s	Saturated moisture content
τ	Time that water is available for infiltration into the soil, otherwise known as the opportunity time (min)
$Q_{t,x}$	Flowrate at node (m^3/sec)
r	Lateral inflow (rainfall) rate (mm)

Chapter 1 Introduction: Towards the development of a decision support system for furrow & border irrigation

1.1 Introduction

Surface irrigation, including furrow and border irrigation, is the simplest and most common method of irrigating crops used throughout the world today. In Australia it accounts for 57% of the total irrigated area (Australian Bureau of Statistics 2005). These systems have the potential to be very efficient, but in practice, they are probably the most inefficient method of irrigation with typical water use efficiencies ranging from 30% to 60% (Raine and Backer 1996; Smith 1988). Computer software programs developed over the last twenty years can potentially increase these efficiencies through helping irrigators improve design and management decisions. However, few irrigators or extension officers currently use any form of simulation model or decision support tool to optimise performance (Raine and Walker 1998). Complexity, limited functionality and reliability problems are possible barriers to the adoption of these tools.

The goal of this chapter is to introduce and discuss the research problem of developing a new *decision support system* aimed at improving the practices of furrow and border irrigation, which are the prevalent forms of surface irrigation in Australia. This chapter has five main objectives: *(1)* it will present the background to this research; *(2)* the research question and hypotheses are introduced; *(3)* justification for this research is presented; *(4)* the methodology used in this research is discussed; and *(5)* the outline of this dissertation is presented.

1.2 Background

Surface irrigation is the technique of artificially applying water to agricultural soils where the soil is used to transmit and infiltrate the water over the field. The water is transported along the field in furrows or borders utilising gravity and hydrostatic pressure differences as the transport mechanisms. Water infiltrates into the soil during this process, which serves to supply moisture for plant growth and provides a delivery mechanism for essential nutrients while leaching and diluting salts in the soil.

Surface irrigation systems have the potential to be very efficient and return high crop yields (Hodgson *et al. 1990*). However, in practice they are typically inefficient with design and management based upon traditional and primitive methods with little knowledge of the efficiency and uniformity of the design.

Recent studies of the Australian cotton industry found that average surface irrigation application efficiencies are as low as 48%, with individual irrigation efficiencies lying between 17% and 100% (Dalton *et al*. 2001; Smith *et al*. 2005).

Similarly, typical application efficiencies of 40% to 80% were measured for irrigated vineyards (Smith 1988). In the Australian sugar industry, a review of surface irrigation practices revealed that application efficiencies averaged from 31% to 62%, with individual irrigations ranging from 14% to 90% (Raine and Bakker 1996). Factors most effecting efficiency includes field-length, irrigation cut-off times, water application rates, furrow shape, soil type and amount of cultivation.

However, these studies have also shown that significant improvements can easily be achieved through the adoption of better design and management practices, to minimise losses caused by tail-water runoff and deep percolation. Optimising these practices through the use of computer simulation models has revealed that irrigation water-use efficiencies of over 90% can be achieved at the field level (Raine *et al.* 1995; Anthony 1995; Smith *et al.* 2005).

Therefore, computer simulation models offer considerable potential to aid in the decision-making processes of irrigation design and management. They represent a cheap and accessible means to experiment, trial and optimise surface irrigation practices. This was proven in many studies that have shown that they are sufficiently accurate to be used in practical applications (Maheshwari *et al*. 1993a, 1993b; Hornbuckle *et al.* 2003; Abbasi *et al*. 2003; Ismail & Depeweg 2005). Also, as an added bonus of their use, they force the irrigator to account for, and measure, his existing management practices.

However, despite more than twenty years of research and development, these tools are yet to reach their potential for improving on-field irrigation performance, having seldom been used in engineering practice (Playan *et al*. 2000). In Australia (and around the world), the adoption of this technology by irrigators and consultants has been poor (Raine and Walker 1998), despite recent workshops and training courses (http://www.ncea.org.au) to promote their use. While model-developers promote the virtues of their products, the reality is that these software applications have been developed as research tools and not practitioner tools, using primitive software engineering technologies. General opinion indicates that the existing models are complex, not robust, sensitive to input data and difficult to operate. Also, they typically perform only the task of performance-evaluation and neglect other decision support requirements such as data management, infiltration parameter estimation, automatic design optimisation, and design-chart generation.

A fundamental cause of all of these limitations and problems lies not in the science of the models, but in deficiencies in structural and interface design. Typically, these programs are developed by engineers and scientists who have been researching the simulation mathematics, and who have limited software engineering experience. Modern software engineering methodologies can potentially overcome these problems by simplifying the complexities of the design task (including solution of the model) while greatly improving the utility of the product.

This dissertation presents an essential initial step in improving the utility of simulation model technology through applying modern software engineering practices to furrow and border irrigation modelling science. This research

assumes that a thorough understanding of both irrigation hydro-informatics and modern software engineering is a prerequisite to design a decision support tool that will see practical use, and consequently improve surface irrigation practices.

Therefore, the research problem for this dissertation is designed to investigate surface irrigation decision support technologies; develop strategies for overcoming gaps in the existing technologies; and incorporating these strategies into a new decision support system for furrow and border irrigation.

1.3 The research problem

In brief, the research problem is to develop an integrated decision support system for furrow and border irrigation aimed at increasing the usability of the technology to improve decision-making capabilities. Specifically the research hypothesis is:

"That calibration, optimisation, and parameter analysis capabilities can be developed and integrated with an accurate and robust simulation model into a decision support system to improve furrow and border irrigation performance."

Six specific objectives have been designed to support this hypothesis and solve the research problem:

Research Objective 1*: Investigate existing surface irrigation modelling technology to determine a model and solution technique structure suitable for incorporating into a decision support system*. Determine why existing surface irrigation software tools have been poorly adopted by industry. It aims to identify, describe and analyse the processes used by surface irrigation researchers in simulating the processes of furrow and border irrigation, and interfacing this technology with decision makers.

Research Objective 2*: Develop a robust, reliable simulation engine for furrow and border irrigation for automation within a decision support system under optimisation and systematic response evaluation.* Based upon the findings of Research Objective 1, develop a simulation engine for furrow and border irrigation that is reliable, flexible and reusable to incorporate into the decision support system, and that can cope under the rigors of automation.

Research Objective 3: *Investigate and develop parameter estimation (calibration) capabilities for the decision support system*. This facility will allow for the automatic determination of any combination of soil infiltration parameters and/or roughness parameter. This includes calibration on irrigation advance and/or runoff hydrographs.

Research Objective 4: *Investigate and develop optimisation capabilities for the decision support system*. This involves developing a user-defined objective-function for irrigation design and management. Optimisation capabilities should allow for inclusion of any input parameter combination.

Research Objective 5: *Investigate and develop parameter response (design charts) capabilities for the decision support system*. This involves developing

tools to allow design and management charts to be automatically generated for a range of infiltration conditions. This facility should also allow response-surfaces of simulation parameter interactions to be generated for sensitivity, parameter, and objective-function analyses.

Research Objective 6: Develop an object-oriented framework to combine the components developed in Research Objectives 2 to 5 with database facilities and a graphical user interface. The objective is concerned with interfacing the science with the user in the simplest way possible. This will ultimately effect how well the software will be adopted by users of various levels.

1.4 Justification of research

This research is justified on four interrelated bases. *Firstly*, decision support for furrow and border irrigation is an under-researched area. A review of the literature provided no evidence of a holistic approach to combining the field-level decision support requirements. Instead, research has focused on the individual components of decision support such as; simulation and performance evaluation (e.g. Walker and Skogerboe 1987; Katopodes 1994; Singh and Bhallamudi 1996); infiltration parameter estimation (e.g. Khatri and Smith 2005; Gillies and Smith 2005); optimisation of practices (e.g. Bautista and Wallender 1993; Ito *et al.* 1999; Valiantzas 2001); and design chart generation (e.g. Zerihun *et al*. 1993; Hornbuckle *et al*. 2003). Questions then arise as to what constitutes a decision support system for furrow and border irrigation, and how the individual components of the system should be combined and interact.

Secondly, problems and limitations are generally known to exist in all of the current furrow and border irrigation tools; including issues of reliability, complexity, and versatility. Evidence for this is difficult to source in the literature, which tends to focus on the benefits of the technologies, but is found amongst users, and through self-investigation of the different software. It is thought that these problems and limitations are hindering the adoption of the technology.

Very little of the irrigation research undertaken over the last 20 years has been developed into an operational form of software available to users. Only two models have succeeded to gain limited, but widespread, acceptance: $SIRMOD$ (Walker 2003) and $SRFR$ (Strelkoff *et al.* 1998). Research bodies around the world seem to be split into different allegiances to either one of these products.

Thirdly, there is a need to combine a range of decision support tools for furrow and border irrigation design and management into a single, easy to use package. Currently, different tools exist for different decision-making purposes. For example, the $INFILT$ (developed in Chapter 4) software package is used solely for determining the infiltration properties of the soil. These results could then be entered into one of the singularly focused simulation models such as $SIRMOD$ or $SRFR$ to determine irrigation performance (Interestingly, the underlying models used in these separate processes are often fundamentally different). Optimisation of irrigation management can then be undertaken using a trial and error approach of repeated simulations of the model. Therefore a tool is required to integrate commonly performed tasks such as performance evaluation, data management, spatial and temporal performance review, infiltration parameter

estimation, automatic optimisation of design variables, parameter response investigation, and sensitivity analysis of design variables.

Finally, the simplicity, utility, flexibility and reliability of decision support tools could be greatly enhanced through the use of modern software engineering practices. These practices offer the potential to simplify the transformation of the modelling mathematics into computer equivalent code, with associated benefits including improved readability of code, powerful debugging capabilities, interchangability of components, enhanced exception handling, faster operating speeds, flexible inputs and outputs, and accessibility for reuse. New software engineering tools allow simple yet powerful graphical user interfaces to be developed using advanced third party libraries, which are intuitive to use, progressively disclose advanced capabilities, and are compatible with other software and the operating system.

1.5 Definitions

This section will define the key terms and concepts of this research so that the direction and focus of the dissertation can be established. These key terms and concepts include: terminology of surface irrigation; design and management of surface irrigation; surface irrigation modelling; and decision support systems for surface irrigation. The importance of these definitions warrants detailed discussion.

1.5.1 Defining surface irrigation practices

Surface irrigation in the context of this dissertation pertains to the practices of furrow and border irrigation. In particular, the software tool that is developed is most suitable for both of these practices, even though most of the validation produced in this following chapters is for the special case of furrow irrigation. In the context of this dissertation, the term “surface irrigation” will be used to imply both furrow and border irrigation, but not practices such as basin irrigation, which would require a different modelling approach. It also doesn’t include surge or cutback irrigation (even though these are forms of furrow irrigation) since the technology is not widely used in Australia.

1.5.2 Defining design and management of surface irrigation.

In Australia, surface irrigation design and management consists of three main components: designing the field layout, scheduling irrigations; and managing the irrigation events. Field design encapsulates decisions pertaining to field-length and width, furrow size and shape, furrow spacing, and field slope (including variable slopes). Water delivery systems must also be considered. Scheduling relates to how much water to apply and when to apply it. Managing irrigation events considers choosing decision variables such as flowrate, time-to-cutoff, and application depth to water the field. Management can also consider irrigation-advance location to guide cutoff, but this will not be investigated in this dissertation.

1.5.3 Defining surface irrigation modelling

A surface irrigation model is a computer based simulation tool used in aiding the design and management of a surface irrigation event. The model consists of a series of mathematical equations that simulate the physical hydraulic processes of the irrigation event. These equations are linked to an interface through which the user manipulates the simulation by entering data representing the physical properties of the irrigation system. The equations are usually solved at discrete time and distance intervals using a suitable finite difference or finite element technique. Operation of the model produces a series of graphical or textural outputs representing the simulated performance of the event.

The term "model" can be used to define both the mathematical constructs used to simulate irrigation, and the computer program in which these equations reside. To differentiate between the two, the term "simulation engine" will hereafter be used interchangeably with the term model to define the mathematical components; while the software program will be referred to as the "decision support system".

1.5.4 Defining decision support systems for furrow and border irrigation.

A decision support system for furrow and border irrigation integrates a surface irrigation model (simulation engine) with a range of other design and management tools into a single software package.

The principal role of the decision support system is to simulate an irrigation event given a set of measured inputs; that is, to predict quantities that are time-dependent and difficult or impractical to measure, given a set of time independent measured input quantities. During a typical simulation, the flow rate and cross-sectional area of flow are predicted at various locations along the furrow length for each time interval. As well, the advance is predicted during the initial phase of the event.

However, if the advance is known, the model can be used to obtain estimates of other parameters, which are normally measured; for instance, infiltration. The solution of the infiltration parameters from a measured advance is known as the inverse solution. This is an important technique, as infiltration is a property that is very difficult to physically measure due to the temporal and spatial variability of the soil.

Lastly, the model should contain an optimisation algorithm allowing automatic solution of the irrigation design parameters; that is parameters that the irrigator has direct control over such as time to cut-off and inflow rate. Absence of this algorithm leads to excessive operator input while manually trying different design parameters configurations in repeated trial simulations.

1.6 Delimitations of scope

This research has five main delimitations of scope: focus upon Australian furrow and border irrigation practices; focus upon engineering aspects of in-field design and management; focus upon the conceptual design of the decision support

system; focus upon a target audience of irrigators, consultants and researchers; and focus on validation against existing technology. These will be discussed in turn.

1.6.1 Focus upon Australian furrow and border irrigation practices.

This research focuses upon the two most predominant forms of surface irrigation in Australia; furrow and border irrigation. Both of these practices can be modelled using the same set of 1D-algorithms with modifications to the furrow geometry parameters. Other forms of surface irrigation such as basin and bay irrigation are more suited to a 2D modelling approach (although the 1D approach has been commonly used in the past), and will not be considered in this dissertation. Also, surge and cutback irrigation will not be considered, as they are currently not widely used in Australia. Even though the tools developed in this dissertation will work for both furrow and border irrigation, validation will predominantly focus on furrow irrigation, since the research focus is at risk of becoming too broad. Also blocked furrow conditions were also not validated for the same reason.

1.6.2 Focus upon engineering aspects of in-field design and management

This research is concerned with the in-field related irrigation design and management issues such as field and furrow design, and determination of optimal inflow rates and cut-off times. This includes all aspects related to the hydraulic modelling of water flowing down along a furrow. This does not include management issues such as irrigation scheduling, or the economic assessment of irrigation performance. Decision support capabilities will be targeted at the field level, and does not include on-farm factors such as distribution in channels, farm storages and agronomic considerations.

1.6.3 Focus upon conceptual design of a decision support system.

This research aims to present a conceptual design of a decision support system for furrow and border irrigation. A prototype decision support system will be developed as part of the research to validate and test the design effectiveness. Nevertheless, the emphasis of this dissertation will be focused upon design concepts rather than the physical product. Unfortunately, it is beyond the scope of this dissertation to study the adoption of the newly developed technology by irrigators and consultants.

1.6.4 Focus on irrigators, consultants and researchers.

The design of the decision support system targets a wide range of user types including irrigators, consultants and researchers. Appropriate and innovative software engineering principles must be adhered to in order to avoid biasing the software towards any particular group.

1.6.5 Focus upon validation against existing proven technology.

The decision support tools developed as part of this study are validated against existing proven technology, rather than actual field experiments. For the simulation engine, this involved comparing modelled outputs against that from

SIRMOD, which has been the subject of several previous studies, including a case study as part of this dissertation (which was based on actual field-data).

1.7 Outline of dissertation

This research in this dissertation is presented in eight parts. It follows a structure resembling that of the decision support system being developed. After a brief introduction and literature review, different components of the decision support system are presented in individual chapters. This includes component topics such as the simulation engine; calibration module; optimisation module; parameter analysis module; and program structure and user interface. Finally, the conclusions arising from this research are presented in the last chapter along with implications of this research and recommendations for future work.

Chapter 1 (this chapter)***:*** An introduction to the dissertation is presented. Firstly, a brief background is given into the research problem before the research hypotheses are proposed. The subsequent sections are based upon defining the terminology, basic theory and limitations faced during the research.

Chapter 2: A literature review of existing surface irrigation modelling techniques and limitations is presented. It focuses on four modelling technologies used in surface irrigation; that is, the volume-balance, hydrodynamic, zero-inertia, and volume-balance models. Finally, it presents a case study of *SIRMOD*, which is one of the most successful models used for furrow and border irrigation decision support.

Chapter 3: A new simulation "engine" for furrow and border irrigation modelling is developed. This engine aims to overcome the limitations of existing models outlined in *Chapter 2*. This will form the "central core" of the decision support system being developed in this dissertation. The chapter presents a redevelopment of the most commonly used simulation methodology into a simpler form. These techniques are then incorporated into a new object-oriented structure designed to coexist inside a modern user-friendly decision support system. Techniques are discussed to achieve simulation robustness. The engine is validated against the *SIRMOD* model that was studied in *Chapter 2*.

Chapter 4: The parameter estimation or "calibration" requirements of a decision support system for furrow and border irrigation are investigated. Not all input parameters used in a simulation model can be directly measured in the field. Soil parameters representing infiltration and roughness need to be estimated indirectly through some form of "inverse" modelling. A simple volume-balance "inverse" technique (*INFILT*) was developed in the stages of this research and presented in this chapter. This work is then followed up with the development of a more powerful hydrodynamic-model technique for incorporation into the decision support system, which uses the *INFILT* methodology to provide starting estimates. The method is validated against real field data while objective-function response-surfaces are generated providing insight into the complexities of the solution process.

Chapter 5: The work in this chapter focuses upon the development of the optimisation capabilities for the decision support system. A flexible new

objective-function equation is derived for optimising irrigation management and design practices. Different formations of this function are examined with results highlighting ridges of constant objective-function value for different design combinations. This equation is linked with an optimisation engine using an object-oriented structure to create the optimisation module for the software. Because of the nature of the system response, it is recommended that only time-to-cutoff be included in the optimisation process which simplifies the operation in the presence of minute variations in the volume-balance caused by numerical discretisation errors.

Chapter 6: The concept of "decision support" in terms of design charts and guidelines is introduced in this chapter. An initial case study is undertaken as an early part of this research. A key feature of this work is the application of historical records to disseminate guidelines for low, average and high infiltration conditions. The design problem is then considered for the automatic generation of design charts within the decision support system. Features of the developed tool include; 4D analysis capabilities by incorporating the fourth variable through multiple charts, or slider-bar functionality; variable exchange functionality; and response-surface filtering. Sample charts are provided.

Chapter 7: The components developed in the previous four chapters are now combined with a database and a simple user interface to develop a new decision support system for furrow and border irrigation. Software engineering issues are initially discussed in this chapter before a suitable object-oriented program structure is developed to accommodate program elements. A new XML-based surface irrigation database is developed as the central core of the decision support system. Finally, a simple graphical user interface is developed with "hyperlinking" capabilities to mimic web-browser functionality.

Chapter 8: The conclusions and implications of this research are presented in this chapter. The work undertaken in the previous chapters is summarised in order to highlight the logical progression of ideas and issues studied in addressing the research hypotheses. Conclusions are presented for the research hypothesis, and associated research objectives. Practical Implications of this research are discussed along with limitations of the results. Finally, recommendations for future research and development are presented.

1.8 Conclusions

This chapter has laid the foundations for this research. Principally it has defined the research problem as developing a decision support system for furrow and border irrigation to improve surface irrigation practices by combining modern software engineering practices with proven irrigation science theory.

Chapter 2 Background to surface irrigation decision support

2.1 Introduction

Computer-based decision support software has the potential to greatly improve surface irrigation design and management practices. Over the last fifty years, much research has been undertaken into the development of decision support tools focusing on the separate research areas of simulating irrigation events, estimating the soil infiltration parameters, and optimising current practices. However, very little of this research has been applied operationally indicating that there are problems with the current technology.

A major objective of this dissertation is to develop a new decision support system for surface irrigation that will overcome these problems in order to place this technology in more hands. Therefore, a comprehensive review of the literature is required to understand the strengths and weakness of the current technology. The goal of this chapter is to address this task in order to determine the structural and functional requirements of new software.

The research reported in this chapter focuses upon four main tasks: *(1)* to present the background theory for the decision support of surface irrigation practices; *(2)* to undertaken a comprehensive literature review of surface irrigation decision support technology, including simulating irrigation events, estimating the soil infiltration parameters, and optimising current practices; *(3)* to evaluate the most commonly used surface irrigation software through a literature review and case study; and *(4)* to consolidate these findings to establish a direction for the development of a new decision support system.

This chapter is accompanied by two appendices containing a full derivation of the continuity and energy equations used in simulation modelling (Appendix 2.1), and a case study to evaluate the commonly used *SIRMOD* (Walker 2003) software package (Appendix 2.2).

2.2 Surface irrigation background

2.2.1 History

Surface irrigation is one of the oldest known forms of irrigation with records showing the practice being used in the Middle East going back thousands of years. Today it is still the most common method for applying water to promote crop growth despite post Second World War interest in pressurised forms of irrigation such as sprinkler and trickle irrigation. While pressurised forms have reduced labour requirements including the need for land-levelling, surface irrigation systems are still often favoured due to lower capital and operating costs, simplicity of maintenance, and the need for unskilled labour (Walker & Skogerboe 1987).

2.2.2 Techniques

Surface irrigation practices have evolved into a variety of configurations, although the distinctions between them are not always clear. Depending upon the way that the water is transported across the field, they can loosely be classified into furrow, border, basin, and wild flooding:

- Furrow irrigation involves the water being transmitted down the field via small channels or furrows. The crop is usually located between the furrows and the water is allowed to run freely from the end of the field (although blocked furrows do exist).
- Bay or border irrigation is the irrigation of long rectangular fields divided into graded borders, with longitudinal slope and free draining. Level borders, however, are not free draining.
- Basin irrigation is the irrigation of small, relatively flat enclosed areas, not allowing for runoff. Interest in level basin irrigation has resulted due to advantages of potentially high distribution uniformities, high efficiency (due to minimal deep percolation and no runoff), and reduced labour requirements (Hoffman and Martin 1993).

Inflow techniques add further dimension to this classification by distinguishing between continuous, cutback, and surge forms of irrigation practice:

- Continuous inflow involves a uniform inflow rate throughout the irrigation until termination.
- Cutback regimes involve an initially high inflow rate to advance the water to the end of the field as quickly as possible to improve application uniformity, and then a reduction in the inflow rate to reduce runoff losses.
- Surge irrigation is the intermittent application of water down a furrow or border in a series of on and off time periods. Surge irrigation is used as a management tool to improve efficiency and uniformity, although the mechanisms contributing to these improvements are not fully understood. It is known however that the soil's infiltration rate is reduced during surge irrigation leading to a quickening of the advance in subsequent surges (Hoffman and Martin 1993). This is thought to result principally from surface sealing effects resulting in a reduction in deep percolation losses. Surge irrigation is largely confined to the United States with the practice not yet adopted by farmers in Australia.

2.2.3 Phases of the irrigation cycle

There are four distinguishable phases that occur during a typical surface irrigation event; namely the advance, storage, depletion, and recession phases:

- The *advance* phase occurs when water is introduced into the furrow or bay and flows downstream on initially dry soil.
- *Storage* occurs once the advance reaches the end of the field. There is a continuous volume of water on the surface of the furrow or bay.
- Once the inflow rate is cutoff, the *depletion* phase begins where the surface water level at the top end of the furrow or bay begins to fall. Some refer to this as the vertical recession phase.
- The *recession* phase exists when a zero surface water depth is encountered at the top end of the furrow or bay. A distinguishable dry/wet boundary front

then propagates along the furrow in the direction of flow. Sometimes this is known as the horizontal recession phase.

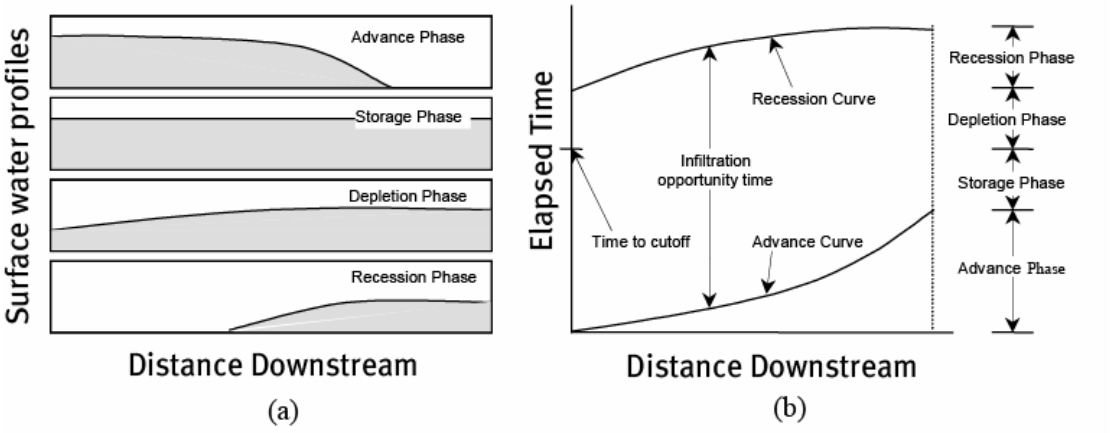


Figure 2.1: (a) Water surface profiles and (b) advance and recession characteristic curves for different phases of the irrigation cycle.

Combinations of these phases can sometimes exist such as simultaneous "advance and depletion" and "advance and recession". These occur when the inflow is terminated before the advance reaches the end of the field culminating in a pulse of water moving down the furrow.

Furrow end conditions designate further categorisation. "Ponding" at the downstream end occurs when the end of the furrow or border has been purposely blocked to prevent runoff. Otherwise, "free-draining" conditions exist.

2.2.4 Design and management practices

Surface irrigation design and management are separate practices requiring different considerations and treatments. For example, irrigation design involves selecting field parameters (such as application method, drainage method, furrow cross-section, field-length and slope) prior to the first irrigation occurring with the multiple goals of simplifying management, maximising performance, and minimising costs. Irrigation management involves controlling the distribution and amount of water for an individual irrigation event (typically through selection of flow rate, time-to-cutoff and application depth) to maximise plant uptake and minimise costs and water losses.

2.3 Surface irrigation decision support

2.3.1 What is a decision support system for surface irrigation?

A decision support system for surface irrigation is computer-based software for aiding the design and management of surface irrigation. Wikipedia (www.wikipedia.org) provides a generalised definition for these systems: *"Decision support systems are a class of computerised information systems or knowledge based systems that support decision making activities."*

The fundamental component of these systems in the context of surface irrigation is the simulation model. The model consists of a series of mathematical equations that simulate the physical hydraulic processes of the irrigation event. These equations are linked to an interface through which the user can manipulate the simulation by entering data representing the physical properties of the irrigation system. The equations are usually solved at discrete time and distance intervals using a suitable finite difference or finite element technique. Operation of the model produces a series of graphical or textural outputs representing the simulated performance of the event.

Other components of a surface irrigation decision support system could include; a database for storage and retrieval of input data, and to serve as a repository for processed results; an optimisation algorithm for calibration and optimisation of model outputs; and analyses to pre-process and post-process input data and model outputs. The system is typically encapsulated by a graphical user interface and contains graphical and textural reporting facilities to communicate information.

2.3.2 Uses of the decision support system

A principal role of a surface irrigation decision support system is to evaluate the performance of an irrigation event. Given a set of measured (time-independent) inputs, the irrigation model is used to simulate an event, so as to predict quantities that are time-dependent and difficult or impractical to measure. During a typical simulation, the flow rate and cross-sectional area of flow are predicted at various locations along the furrow length for each time interval. As well, the advance is predicted during the initial phases of the event. From this, the performance of the irrigation can be evaluated in terms of efficiency, uniformity, and volumes.

However, if the advance is known, the model can be used to obtain estimates of other parameters, which are normally measured; for instance, infiltration. The solution of the infiltration parameters from a measured advance is known as a solution to the “inverse problem”. This is an important technique, as infiltration is a property that is very difficult to physically measure due to the temporal and spatial variability of the soil.

Lastly, repeated simulations of the model using different combinations of design variables are fundamental for optimising design and management strategies. These variables are usually those that the irrigator has direct control over such as time-to-cutoff and inflow rate. Optimal strategies can be determined through a manual trial and error approach of running the model; or systematically simulating all combinations of the variables to generate design curves; or through a structured optimisation strategy. The addition of an optimisation algorithm to automatically determine these variables is a higher objective for a decision support system.

2.3.3 The need for decision support systems

Extensive research has been undertaken towards improving decision making involved in the design and management of surface irrigation, yet irrigation practices have changed little over the past decades. While recent technology has introduced a wide range of tools and techniques to draw upon, their adoption has been slow and generally poorly coordinated. Decisions are more often influenced by local policy, neighbouring practices, consultants' experiences and preferences, and availability and cost of materials, rather than by evaluation and comparison of methods for the site of interest.

Field trials and computer modelling are becoming more commonplace in developed nations and have set the standard for surface irrigation decision making, although both are limited in their current form. Computer models offer the greatest potential but so far have really only been used as research tools, indicating that there are problems with the current technology/software.

Therefore, an opportunity exists to develop a decision support tool for surface irrigation that will overcome these problems and place this technology in more hands. It needs to simplify the multidimensional nature of the decision process by simultaneously comparing and demonstrating different techniques and methods over a range of conditions.

2.3.4 Research for decision support systems

Research into decision support systems has been directed in three principal areas including:

- simulation of furrow and border irrigation;
- solving the "inverse problem"; and
- optimising design and management practices.

Simulation models have been the main focus of surface irrigation research over the last fifty years. There is a considerable body of literature on simulation modelling, which unfortunately is not all encompassing. While the methodologies presented are typically sound and often novel, the depth of the subject area has meant that many topics have been poorly treated.

Solution of the "inverse problem" has been the second most researched area of furrow and border irrigation decision support. The "inverse problem" relates to using the simulation technology in "reverse" to estimate field parameters (such as the infiltration or hydraulic roughness parameters) through a calibration of the simulated outputs against measured field data (such as the advance, or runoff hydrograph). Simple methods have been developed which have proven useful, but the research is yet to take full advantage of current technology.

The least treated research area is that of using simulation technology to optimise design and management practices. Different methodologies have been presented to optimise these practices including generation of design charts, computer optimisation, and real-time control of irrigation systems. The relative lack of depth in this subject area is indicative of both the infancy of the technology, and problems associated with the basic simulation capabilities.

Background modelling theory and a review of the literature in each of these three areas will now be presented.

2.4 Background to simulation modelling

The simulation model is the primary component of a surface irrigation decision support system, and the success of secondary tools is dependent upon the effectiveness of the simulation model. Therefore, before a literature review of surface irrigation decision support technologies can be undertaken, an understanding of the background theory of simulation modelling is required. This theory will now be discussed in terms of the underlying model equations; simplifications to the model; infiltration model; numerical solution techniques; and dimensionless formulations of the equations.

2.4.1 Model equations

Computer simulation of water flowing and infiltrating along the furrow or border during surface irrigation is governed by the laws of mass and momentum conservation. The mathematical equations describing these laws are generally known as the *de Saint-Venant Equations* (often the "de" is omitted). They consist of two separate hyperbolic partial differential equations[1]; one representing continuity (Eqn. 2.1); and one representing momentum (Eqn.2.2).

$$\frac{\partial q}{\partial x}+\frac{\partial y}{\partial t}=r-z \qquad (2.1)$$

$$\frac{\partial v}{\partial t}+v\frac{\partial v}{\partial x}+g\left(\frac{\partial y}{\partial x}\right)=g\left(S_o-S_f\right)+\frac{v(z-r)}{y} \qquad (2.2)$$

Where q is the inflow rate per unit width, x is the distance along the furrow, z is the rate of infiltration per unit length, y is the depth of flow, t is the irrigation time, g is the gravitational constant, S_0 is the furrow slope, S_f is the energy slope, v is the velocity of the surface flow, and r is the lateral inflow (rainfall) rate.

For completeness, a derivation of these equations is presented in Appendix 2.1 as many texts fail to state the assumptions underlying the derivation; are not thorough in the derivation; and neglect to include lateral inflows and outflows.

2.4.2 Simplification of the model

The solution of these equations is complex, and as yet, no analytical solution to the complete equations has been found. Therefore researchers have typically used a combination of numerical methods and simplifying assumptions to obtain a solution. The simplification of these equations can be categorised into four main types based on the level of modification. These include the solution of the full set of hydrodynamic equations, the zero-inertia approximation, the kinematic-

[1] Note that this form of the equations is most suited to flow in borders. A different form of the equations, that better represents flow in furrows (in terms of Q and A, rather than q and y), will be presented in Chapter 3, and used in the developed software.

wave approximation; and the volume-balance method. These will now be discussed in turn.

Solutions based on the full set of hydrodynamic equations

The complete form of the Saint Venant Equations (Equations 2.1 & 2.2) represents the most accurate description of the water flow over the ground surface. However, complex numerical methods are required for their solution, being costly both in programming complexity and execution times.

Solutions based on the zero-inertia approximation

The first level of simplification of the Saint Venant Equations involves removing the inertial, or acceleration, terms in the equations. The acceleration terms are a source of fragility, especially if the flow is near critical. Their removal from the momentum equation leads to a more robust solution, as the equations are now parabolic, rather than hyperbolic (Strelkoff and Falvey 1993). The momentum equation then becomes:

$$\frac{\partial y}{\partial x} = S_0 - S_f \quad (2.3)$$

Numerous studies (Katopodes and Strelkoff 1977b; Clemmens 1979; Fangmeier and Strelkoff 1979; Elliot *et al*. 1982; Schwankl and Wallender 1987) have shown that this zero-inertial approach can accurately simulate flow in borders and furrows. Surface irrigation is typically characterised by sub-critical flow with Froude numbers close to 0.3, meaning that the zero-inertia assumptions are seldom violated.

Application of this approximation is relatively simple and computer execution times small even though it is usually solved numerically. Several researchers have formulated analytical and quasi-analytical solutions in recent years.

Solutions based on the kinematic-wave approximation

The kinematic-wave approximation is the next level of simplification to the Saint-Venant equations. This approximation further simplifies the zero-inertia equations by assuming that the water surface slope is relatively small compared to the other terms of the equations. The momentum equation then becomes:

$$S_0 = S_f \quad (2.4)$$

The model assumes uniform flow conditions at the furrow inlet and outlet, but not along the furrow reach due to lateral outflows (infiltration). Effectively, the assumption is that the bed slope can approximate the frication slope, and that flow is “uniform” only over discrete distances. The absence of the depth gradient term ($\partial y / \partial x$) from the equation implies that the depth at the top end of the furrow will instantaneously reach the value of normal depth as soon as flow commences, rather than rising gradually as you would expect in practice.

Many analytical solutions to these equations have been found including explicit and implicit “time of advance” equations. Numerical solutions are still used as well, and are simple to code and operate very efficiency.

The limitation of the method is that it is only applicable to special conditions that don't void the underlying assumptions. For example, the method cannot handle backwater effects due to the uniform flow boundary condition at the furrow outlet, so it cannot be used for blocked, or partially blocked furrow ends.

Solutions based on the volume-balance approximation

The volume-balance model represents the simplest approximation of the Saint Venant equations where the momentum equation is usually replaced with an assumption of average depth of flow or a predetermined water-surface profile. Shape factors are often used along with empirical "power" relationships to define advance trajectories. The depth of flow at the top end of the field is assumed to be the normal depth for the applied flowrate. Only the continuity equation from the Saint-Venant equations is used and is usually solved algebraically.

2.4.3 Infiltration model

The most commonly used infiltration functions in surface irrigation modelling include the Kostiakov-Lewis, Modified-Kostiakov-Lewis, Phillip, and Horton equations (Table 2.1). These are empirical equations and are primarily designed for non-cracking soils, neglecting to consider instantaneous crack filling (Maihol and Gonzalez 1993). However, it is not uncommon for these equations to be modified through the addition of a crack-fill parameter. Physically based models such as the Green and Ampt equation (Eqn. 2.9) may be better suited to the cracking situation, but are difficult to incorporate in simulation models and apply to field situations (Evans *et al*. 1990).

Table 2.1: List of commonly used infiltration equations in surface irrigation modelling

Source	Equation	
Kostiakov-Lewis (Kostiakov 1932; Lewis 1937)	$Z = kt_{op}^{a}$	(2.5)
Modified-Kostiakov-Lewis	$Z = kt_{op}^{a} + f_o t_{op}$	(2.6)
Phillip (1957a)	$Z = kt_{op}^{\frac{1}{2}} + t_{op} f_o$	(2.7)
Horton (1940)	$Z = \gamma(1 - e^{-rt_{op}}) + f_o t_{op}$	(2.8)
Green and Ampt (1911)	$I = K\left[1 + \frac{(\theta_s - \theta_i)h'}{Z}\right]$	(2.9)
SCS, USDA, also Evans *et al*. (1990)	$Z = kt_{op}^{a} + C$	(2.10)
Maheshwari-Jayawardane (1992) also Maihol-Gonzalez (1993)	$Z = Z_c + i_l t_{op}$	(2.11)
Wallender-Rayej (1985)	$Z = kt_{op}^{a} + f\ t_{op} + S$	(2.12)

where Z is the cumulative infiltration ; t_{op} is the opportunity time for infiltration to occur; a, k and r are fitted empirical parameters; f_o is the steady state infiltration rate; I is the infiltration rate; θ_s is the saturated moisture content; θ_i is the initial moisture content; K is the hydraulic conductivity in the wetted zone; h' is the suction at the wetting front, γ is cumulative infiltration at t_{op}=0 if steady state had been reached at t_{op}=0; C is deferred infiltration volume; i_l is the saturated hydraulic conductivity of the restricting layer; Z_c is the initial depth of infiltration required for crack filling and absorption; and S is the volume of cracks per unit area.

With the exception of the Green and Ampt equation (Eqn. 2.9), all of the equations are of a similar form. That is, they are composed of variations of a non-linear component, a steady state component, and sometimes an instantaneous crack-fill component. Also, the actual role of each equation-part is not necessarily fixed, and is dependant upon the parameter values. For example, the Kostiakov-Lewis Equation (Eqn. 2.5) could be used to represent any one of the three components. In general this equation represents non-linear infiltration. However, if the parameter a is set to 0, it would represent instantaneous crack-fill; and if a is fixed at 1, then it would represent steady-state infiltration.

2.4.4 Numerical solution techniques

While dozens of attempts have been made to develop surface irrigation models over the last century, only a relatively small range of solution techniques has been employed. Early work involved the solution of volume-balance models involving only simple algebraic equations. The limitations of these models were recognized early and with the advent of modern computers, attempts were made to solve variations of the more complex hydrodynamic models, which have proved impossible to solve algebraically in the purest form. Therefore, iterative numerical solution techniques are required to solve the models.

Conceptually, the modelling consists of two levels of approximation: *firstly*, the governing equations represent an approximation to reality; and *secondly*, the numerical method chosen is an approximation to the analytical solution of the governing (differential) equations. Numerical methods used include the method of characteristics, finite differencing, finite element analysis, and finite volume analysis. Finite differencing is the most commonly used method in surface irrigation modelling.

Method of Characteristics

This technique simplifies the problem of solving the two *partial* differential equations (continuity and momentum) by transforming them into four *ordinary* differential equations (Stephenson and Meadows 1986). This technique was used long before the advent of modern computers through graphical solution methods. These days, numerical techniques are used through explicitly solving the transformed equations on an irregular space-time grid formed by the intersection of disturbance-trajectories (characteristics curves). A limitation of the method, due to the irregular grid, is that inputs to the characteristic equations must constantly be interpolated from the previous calculations. This is both computationally intensive and a potential source of compounding error. For an extensive treatment of this theory, see Courant and Hilbert (1962).

Finite differencing numerical methods

The majority of surface irrigation models, including the latest state-of-the-art models such as *SIRMOD* (Walker 2003) and *SRFR* (Strelkoff *et al.* 1998), utilise finite differencing techniques to approximate the differential equations. These methods involve replacing the partial derivative terms in the governing equations with discrete approximation terms (since it is the derivative terms which prohibit the formulation of an analytical solution). This is the most common solution

technique used in surface irrigation modelling, as it is in most problems involving one-dimensional, steady or time-dependant flow (Chaudhry 1993).

Finite difference techniques can be categorised into implicit or explicit techniques. Implicit techniques solve "simultaneously" for all of the unknown variables at a time-step, where as explicit techniques will solve for the unknowns in a "marching" cell-by-cell fashion. Explicit techniques are generally simpler to program and easier to debug, but they are more susceptible to instability problems (Chaudhry and Mays 1993). Stability checks such as the "Courant condition" (Courant *et al*. 1928; 1948; and 1956) must be performed to ensure that the nodes at which each solution is sought, lie within the zone of dependence of the neighbouring nodes that are used in the derivative approximations. Also, in surface irrigation modelling (though not necessarily in general) explicit methods are assumed to be an order of magnitude less computationally efficient than the implicit techniques (although no evidence could be found of actual comparisons).

Other numerical methods

Other possible methods for solution of the model equations for furrow and border irrigation include finite element and finite volume analysis. Finite element methods have been used for modelling furrow irrigation conditions (Shayya *et al.* 1993), modelling sediment and chemical transport (Katopodes 1994), and two dimensional analysis of furrow infiltration (Vogal and Hopmans 1992), but has not found widespread application in modelling general open-channel and irrigation flow conditions. Singh and Bhallamudi (1997) developed a finite volume method to solve the two-dimensional governing equations of basin irrigation.

2.4.5 Dimensionless solution formulations

Dimensionless formations of the finite difference forms of the continuity and energy equations have been used by many researchers for modelling surface irrigation (e.g. Strelkoff 1985; Rayej and Wallender 1985). The main advantage of a dimensionless solution of the model equations is that it allows significant reductions in the amount of data generated and presented, without loss of generality (Strelkoff and Clemmens 1994). These dimensionless solutions are independent of the systems of units used. Dimensionless variables appear in the form of ratios of the normalised quantities used. Dimensionless solutions have been favoured for creating design charts due to the reduced number of variables present (Ram and Singh 1985).

2.5 Simulation model development

The mathematical modelling of surface irrigation is not new. Volume-balance models had been developed as early as the 1930's, and the advent of modern personal computers in the 1970's saw more complex forms of model being introduced with energy and momentum components. A review of the evolution of these models will now be undertaken focussing on the four separate types of

simulation model available including: volume-balance models; kinematic-wave models; zero-inertia models, and hydrodynamic models.

2.5.1 The evolution of volume-balance models

The first attempts at simulating the advance-phase of surface irrigation used volume-balance models that apply the continuity equation (Eqn. 2.1) over the entire flow profile and use simplifying assumptions to replace energy and momentum effects. They are still of interest today because of their simple form which is easy to program into a computer, and couple with optimising and batch-processing algorithms. Both analytical and iterative solutions have been developed, with key differences between methods lying in the underlying assumptions for surface and sub-surface geometry.

The first attempt at an analytical solution was by Lewis and Milne (1938) who derived an integral equation for advance time in terms of inflow, mean surface water depth and cumulative infiltration. Philip and Farrell (1964) later used Laplace transforms to simplify the integral equation for different infiltration equations.

The first iterative numerical scheme for solving the volume-balance equations was developed by Hall (1956) in what has been described as a "landmark contribution" (Al-Azba and Strelkoff 1994) to the research. This method for solving the advance laid the foundation for physically based numerical irrigation models and has regularly been referenced in journal articles and books ever since. Hall's approach used an iterative numerical scheme over a sequence of time-steps where normal depth at the upstream end was assumed along with a power-law shape factor for the surface stream-depth profile yielding a "reasonable" approximation of the actual advance.

Another algebraic volume-balance model was later developed by Strelkoff (1977) for generating approximate advance and recession curves for border irrigation. He approximated the surface volume by using a simple surface shape profile, and assuming normal depth at key points along the surface. Results were compared to other more complex mathematical models, and to laboratory and field experiments with results of "useful quality".

This technique was later modified for furrow irrigation by Levien and Souza (1987), using a power function to represent furrow geometry and a simple Kostiakov function to represent infiltration. Having validated the model against field data as well as other models, the authors claimed "reasonable results" but believed that it was more accurate than many of the more complex models.

At the same time, Rayej and Wallender (1987) also produced a volume-balance model for open furrow irrigation. They used the same assumptions made by Strelkoff (1977) along with a modified Kostiakov infiltration equation, and a power equation to represent furrow geometry. During the recession phase, the flow-area at the downstream end was assumed proportional to the distance of the recession from the downstream end. This was questioned later by Xu and Singh (1990) who thought it more reasonable to assume that flow "depth" instead of flow "area" should be proportional to the recession distance. Low

computational efficiency and programming complexity were also acknowledged as a limitation of the method.

Xu and Singh (1989, 1990) developed an analytical volume-balance model to simulate all phases of the irrigation cycle for borders and furrows. A parabolic shape was assumed for surface and subsurface profiles during the advance phase; the coefficients of which were determined in the region of gradually varied flow away from the advance front. Recession was simulated using a simple iterative technique by Strelkoff (1977) that assumes that the surface profile is linear during this phase and that "*the downstream end flow is normal and declines proportionally to the receding tail position measured from the downstream end*". The furrow irrigation version of the model can be applied to any form of furrow shape by transformation into an equivalent semicircular shape.

Another volume-balance approach used by several authors involves replacing the energy equation with a Muskingum storage-discharge relationship (Singh and He 1988; Singh *et al.* 1988; Wilson and Elliot 1988). Singh demonstrated this method for both furrows and borders (with a modified Kostiakov infiltration equation) and found that it "satisfactorily" predicts water movement and infiltration distribution for all phases of the irrigation cycle, with errors comparable with other models. However, the model contains twelve input coefficients making it difficult to calibrate for a volume-balance model. The main advantage of the model is that it is simple to program and executes rapidly.

Wilson and Elliot (1988) used a modified version of the Muskingum flood-routing approach to predict the advance, along with a second routing method that assumes that the flow is approximately steady within each furrow reach. Each method used a power function to describe the surface and subsurface profiles. Both models predicted advance times accurately on soils that quickly reach steady state infiltration and had relatively high infiltration losses. However, the second method was found to over-predict the advance on low infiltration soils.

Al-Azaba and Strelkoff (1994) revisited the work of Hall (1957) showing that the original technique had problems associated with volume-balance errors occurring at the infiltration wetting front, and accumulating over each successive time-step. They found that errors could be as high as 10% for small advance times, but the error decreased with increasing time. This is also not uncommon with today's more complicated hydrodynamic models. The authors neglected to point out that the higher volume-balance errors correspond with smaller volumes. However, they did present a more accurate model by redeveloping the technique and calculating volumes at each time-step rather than computing incremental changes. They also modified the model to account for the sharp curvature of the infiltration profile at the advance front.

In recent work, Greek researcher John Valiantzas has undertaken considerable research with volume-balance models. In 1993, he developed a simple model to predict advance in borders using a volume-balance equation with an adjusted surface shape factor and the zero-inertia motion equation evaluated at the head of the border (Valiantzas 1993). A system of two equations containing the two unknowns of advance position and upstream depth were solved at each time-

step. The method performed well against the zero-inertial solution and four well-documented field irrigations. It also proved accurate on borders with small slope where volume-balance models traditionally fare poorly.

Four years later, he developed a new algebraic form of volume-balance model (Valiantzas 1997a,b) describing the time variation of the subsurface water profile. This equation was developed by solving the modified Hall technique (Al-Azaba and Strelkoff 1994) for different values of the exponent of the Kostiakov equation. By "systematic" analysis of the results, a relationship was derived relating the subsurface water profile to time for infiltration represented by the Kostiakov Equation. This analysis was then extended to other infiltration equations. However, a limitation of the resulting equations is that the advance time in the model is given implicitly. Therefore, its calculation requires the use of an iterative numerical technique such as the Newton-Raphson method. Nevertheless, the resulting equations were found to be substantially more accurate than previous algebraic volume-balance equations utilising the power advance assumption.

Valiantzas (1999a,b) readdressed the problem of the iterative nature of the method two years later, when he analysed the behaviour of dimensionless advance curves for various combinations of infiltration parameters. He found that by expressing the advance problem in terms of conveniently selected dimensionless variables, the various advance curves (obtained using the modified Hall technique), could be described by a single advance curve independent of the infiltration parameters. A "simple" explicit time-of-advance equation was then derived to explain this curve. The proposed formula was compared against the kinematic-wave numerical results with good agreement.

The next year, Valiantzas (2000a,b) admitted that this technique was not as simple as he had initially suggested. A disadvantage of the method is that the advance relationship described by the model is presented as three equations of different mathematical form, lacking the simplicity of the popular SCS empirical equation (U.S. Department of Agriculture 1983), with the extra calculations required deemed significant. Therefore he derived another simpler equation based on the systematic analysis of dimensionless zero-inertia numerical solutions, yielding good agreement with the zero-inertia model.

Once again in the following year, Valiantzas (2001) simplified the previous model further, proposing a new time-of-advance equation equivalent in simplicity to the SCS equation. This equation is derived as an extension of the two small-time large-time explicit advance time solutions. He evaluated this against the SCS equation, which was found to perform poorly against his new design. He also tested the equation against observed furrow data, and a zero-inertia model and found good agreement with both.

2.5.2 The evolution of kinematic wave models

First proposed by Lighthill and Whitman (1955) for modelling overland flow, the kinematic-wave approximation was widely utilized soon after in catchment hydrology and for predicting flood movement in rivers (Henderson and Wooding 1964; Wooding 19665a,b). This hydrologic application of the model was

extended to sloping, free draining borders by Chen (1970) using the method of characteristics. Smith (1972) made improvements to Chen's work, while also introducing his own solution technique based on finite difference approximations to the partial differential equations.

Walker and Humpherys (1983) highlighted that the kinematic-wave approximation developed for borders, could easily be modified for furrows by including a description of furrow cross-section along with a wetted perimeter dependant infiltration function. This was later modified by Ross (1986) in a model called "KIM" using a variable time-step (to keep the space-step approximately constant) and a volume-balance approach to calculate recession times.

Rayej and Wallender (1988) developed a kinematic-wave model to solve for the time-of-advance to specified locations down the field using the implicit double-sweep technique of Liggett and Cunge (1975). However, at this stage of model development, the coding requirements were found to be extensive and computational times long, even for this simple model type. Wallender and Yokokura (1991) followed on from this work, using an explicit Newton Raphson solution method that iteratively adjusts the time-step to ensure that the nodes fall at fixed locations, rather then using fixed node locations and solving implicitly for the time-step. This proved to be more computationally efficient than Rayej and Wallender's method, while providing identical results.

Shayya, *et al.* (1993) developed a unique model for all phases of furrow irrigation based upon application of the one-dimensional Galerkin formulation of the finite-element method to the numerical solution of the kinematic-wave equations. The Kostiakov-Lewis infiltration equation was used without wetted-perimeter compensation. Results showed that the method produced "excellent" predictions of the advance and recession trajectories for "almost" all simulations of the available field tests. Simulation times of thirty seconds on a "386" computer suggests that the method is an order of magnitude slower than the implicit finite difference method.

One researcher who has contributed greatly towards developing the kinematic wave models is Vijay P. Singh of the USA. For over twenty years from the late 1970's, he provided a comprehensive mathematical treatment of kinematic wave modelling of surface irrigation. In his early work with Sherman (Sherman and Singh 1978, 1982; Singh and Sherman 1983), and later with Ram (Singh and Ram 1983, 1984) he refined a model using a variety of numerical solution techniques (one being the Kinematic Wave Train method of Ram *et al.* 1983) for different phases of the irrigation cycle. These solution techniques were simplified in later work becoming part numerical, part analytical (Singh and Ram 1985).

Singh and Ram (1983) tested the model against data from thirty-one experimental borders. They concluded that while it was quite good at simulating advance and recession, it struggled to handle the depletion phase adequately. This agrees with an earlier study that Singh was involved in (Chen *et al.* 1981) and also that of Smith (1972). In the following year, Singh and Ram (1984) further developed the solution for the transient infiltration case in the "Kinematic

Wave Train" formulation. This method can be used with many infiltration models and uses a semi-analytical recursive solution technique.

They then focused on providing quantitative estimates regarding the accuracy of the model through the development of dimensionless advance and recession curves for border irrigation (Ram and Singh 1985). In the majority of cases, they found the model to be "sufficiently accurate".

In 1986, in a series of papers with Ram and Prasad, Singh presented a quasi-steady state integral model for both closed and free draining border irrigation (Ram *et al.* 1986a,b). They used a semi-analytical method for solving the governing equations for all phases of the irrigation cycle. The continuity and the *quasi-steady state approximation* of the momentum equation were integrated for the depth and velocity distributions of the surface water while a Kostiakov infiltration function predicted the subsurface distribution. Results were reported as "satisfactory" with prediction errors between twenty and thirty percent. The following year, Singh published several more papers (Singh and Yu 1987a,b,c) on this model.

This work was continued in 1989 with a model for free draining borders simulating all phases of the cycle (Jain and Singh 1989). This was based upon the integral formulation with a Newton-Raphson iterative scheme and could accommodate any infiltration function. Very low volume-balance errors were encountered while testing against field data. However, Singh later reported that this technique might converge slowly under some situations (Reddy and Singh, 1994).

To address this problem, Reddy and Singh (1994) reinvestigated the linearisation scheme that Strelkoff and Kotopodes (1977) had used to directly solve the non-linear equations. They rederived explicit algebraic expressions for the linearised form of the kinematic-wave equations, for computation of advance and runoff rate in furrow irrigation. A modified Kostiakov-Lewis infiltration equation was used in the derivation. They used a deforming control volume approach during the advance phase while resorting to a fixed control grid for the storage and runoff phases. The authors didn't consider the depletion and recession phases suggesting that in general, these phases are insignificant in furrow irrigation. They compared their results with field data, and also a zero-inertia model and found close agreement between them all. They also derived a differential equation to estimate the error between the kinematic-wave and zero-inertia forms of the model by assuming a constant infiltration rate. This equation could then be used to define the limits of usability for the kinematic-wave model in furrow irrigation.

Turbak and Morel-Seytoux (1988) also assumed a constant infiltration rate when they developed an analytical solution to the kinematic wave model for all phases of the irrigation cycle. The authors acknowledge that the assumption of a constant infiltration rate would introduce a significant prediction error. Nevertheless, this modelling attempt highlighted that an analytical solution could be found through the use of a simplified linear infiltration assumption.

This method was revisited by Maihol and Gonzalez (1993) for the recession phase when developing an analytical furrow irrigation model targeted at real-time applications on cracking soils. The advance phase of the model was solved using a Laplace transform solution of the Lewis-Milne (volume-balance) equation containing a linear crack-fill infiltration function, to produce an "implicit" time-of-advance equation. As with Turbak's method, the simplified infiltration equation limits the utility of the method, especially during later stages of the irrigation, and on cracking soils that may have a "zero" final infiltration rate. The authors also suggested that the linear form of the infiltration equation facilitates the need for statistical analysis to overcome the problem of spatial variability.

These attempts at an analytical solution were not lost on Australian researchers Austin and Prendergast (1997), who were more successful in their approach to develop a simple analytical irrigation model based upon the kinematic wave approximation. The model was specifically targeted at cracking clay soils, and included a simple linear infiltration function deemed suitable for describing infiltration into soils exhibiting shrinkage and cracking upon drying. The two parameters used in the function have physical significance allowing pre-irrigation estimation of these parameters in the field. It is because of the linear nature of this function (that is, a constant rate of change of infiltration over time) that an analytical solution to the kinematic wave model was able to be derived for all phases. The method is simple enough that it can be undertaken using hand calculations without the use of a personal computer. Initial results were good enough to warrant future research.

2.5.3 The evolution of zero-inertia models

In a "ground-breaking" paper, Strelkoff and Katopodes (1977) developed the first zero-inertia model for border irrigation. They borrowed the idea of using the zero-inertia approximation from Brakensiek *et al.* (1966), and from Harder and Armacost (1966) who utilised the technique in river-flood routing. Problems in adapting these methods to border irrigation were identified in the region of the advance front where the depth is zero, and with convergence problems towards the end of the depletion phase. They dismissed the use of an explicit solution technique because of these convergence problems and therefore presented an implicit double sweep technique solving the integrated form of the governing equations in the x-t plane, paving the way for the state-of-the-art models of today. For each time-step, a series of non-linear equations were first linearised and then solved using the double-sweep algorithm. Solution techniques were presented for each individual phase with "satisfactory" results. This was later verified by Fangmeier and Strelkoff (1979), while the method was later modified for closed-end borders by Clemmens (1979).

Five years later and following on from the work of Strelkoff and Katopodes, Elliott *et al.* (1982) presented the first zero-inertia model for furrow irrigation advance. The model equations were linearised and then solved on a moving Langrangian solution grid using a double-sweep algorithm. They related both depth and wetted-perimeter to cross-sectional flow area using a power relationship, something that again has become popular. They presented only the advance phase of the simulation, deeming it of greatest interest as it is most responsible for the uniformity in the final distribution of infiltrated water (the other phases of

the irrigation were later presented in a book by Walker and Skogerboe, 1987). Field data were used to validate the model concluding that significant errors were not introduced by neglecting the acceleration terms, nor by linearising the series of algebraic equations.

The authors followed up this work the following year by non-dimensionalising the zero-inertia equations so as to be able to present graphically, a series of dimensionless advance curves for irrigation design (Elliot *et al.* 1983a). It is interesting to note that this team abandoned this dimensionless approach (which is still used by researchers such as Strelkoff and Katopodes), when developing their popular *SIRMOD* model.

Schwankl and Wallender (1987, 1988) presented a zero-inertia furrow model with variable infiltration and hydraulic characteristics. The method was novel in that the model equations were solved at specified space increments rather than specified time increments using an explicit finite differencing technique. This allowed them to vary infiltration and hydraulic properties at locations along the furrow corresponding to measurements of infiltration, roughness and geometry. They found great benefit in this, stressing the importance of wetted perimeter effects on infiltration.

Australian researcher Ross developed a zero-inertia model for furrow and border irrigation in 1987 (Ross, 1987). Aptly named "ZIM" (Zero-Inertia Model), the model followed on from the work of Strelkoff and Katopodes (1977) and Elliot *et al.* (1982) using a Newton-Raphson technique in the solution process. This model managed to earn some early recognition with Australian primary industry researchers, although later models (such as *SIRMOD* and *SRFR*) soon overshadowed it.

Schmitz and Seus (1989, 1990, 1992) developed a zero-inertia model for irrigation in borders and furrows. The model is unique in that an analytic solution replaced the finite difference approximation to the derivative terms in the model. Up until this time, analytical solutions could only be achieved for volume-balance approaches. However, it is not an explicit time-of-advance model, and does require iteration over successive time-steps. The model can be used with any infiltration equation and a range of furrow geometries can easily be accommodated. The authors claimed that the model is very accurate and numerically economic avoiding the normal errors introduced by numerical discretisation. They compared the model against a full hydrodynamic model and also field data showing "excellent agreement". Unfortunately, along with many other models discussed, it doesn't seem like any further development has taken place in the time since it was first published.

Oweis and Walker (1990) modified the method of Elliot *at al.* (1982) for the situation of surge flow. The model simulates all phases of the irrigation cycle including simultaneous advance and recession, which is regularly overlooked by researchers. Unfortunately, although this phase combination is crucial in surge flow modelling, the authors still only offered a brief coverage of the topic passing up an opportunity to publish in an important but little understood area. Nevertheless, the model appears to be successful, and would later on contribute to the development of *SIRMOD*. Neglecting the inertial terms was thought to

have minimal impact on the accuracy of the simulation despite the relatively high flow velocities associated with surge irrigation.

Katopodes (1994) developed a unique and powerful model for simulating the surface irrigation advance that also calculates the velocity profile of the surface-water wetting front. This complex technique makes possible the analysis of particle suspension and chemical transport, which is in fact the reason for its development. A two dimensional finite element approach in the vertical plane is used to solve the Navier-Stokes equations in the region of the wave front. Zero-inertia theory is then used a short distance upstream where a fully developed vertical structure is encountered. The model is not suggested as a tool to analyse or design border irrigation flow and the author admits that it could not compete with any of the recent hydrodynamic models with results differing considerably from a zero-inertia model. It presents as a first attempt to model surface irrigation based on the turbulent Navier-Strokes equations addressing the problem of point-source contamination from irrigation runoff.

2.5.4 The evolution of hydrodynamic models

Nearly all furrow irrigation models up until the late 1970's employed volume-balance methodologies. While hydrodynamic modelling had commenced a decade earlier, it had met with little success. Since then, most interest in hydrodynamic modelling has been directed towards using explicit solution techniques to solve the equations, with stability problems and slow computation times reported. The most successful methodology (and benchmark standard) appears to be the implicit double-sweep technique of Walker and Skogerboe (1987).

Wilke (1968) presented one of the first attempts at using a hydrodynamic model using the method of characteristics, but computational difficulties near the advance front prevented accurate predictions of the advance trajectory. The first "successful" attempts at simulating all phases of the irrigation cycle using the full hydrodynamic model were made by Basset (1976) and Katopodes and Strelkoff (1977b) using finite differencing techniques based on the method of characteristics. Long simulation times and the high cost of producing runs on shared computing systems stirred much interest in simplifying the model by neglecting accelerations terms in the momentum equation.

Fonken *et al.* (1980) presented a complete hydrodynamic model of all phases of border irrigation. The authors employed a Newton-Raphson-based numerical integration technique applied to control volumes in replacing the method of characteristics (which was popular throughout the previous decade), with great improvements in numerical efficiency. The numerical costs associated with running this new model were comparable to that of the simpler zero-inertia models of the time. Interestingly, the model was presented as a useable tool, rather than academic concept, which tended to add some credibility to the science at that time.

The underlying model for the *SIRMOD* software was presented by Walker and Skogerboe (1987) in their textbook for surface irrigation theory and practice. The method uses a Eulerian integration approach to approximating the Saint-Venant

equations and solving with the implicit double-sweep technique of Liggett and Cunge (1975). The authors have presented what is probably the most comprehensive treatment of hydrodynamic simulation modelling for furrow irrigation to date, and includes treatments for initial conditions and downstream boundary conditions. Given the success of the *SIRMOD* software (see Section 2.9), this model has set the standard for surface irrigation models.

During the late 1980's there was considerable interest in simulating irrigations using fixed node locations and solving for the time-of-advance using volume-balance, kinematic wave, and zero inertia models (Wallender 1986, Rayej and Wallender 1988; Schwankl and Wallender 1988). A benefit of specifying node locations is that different infiltration functions (uniform or stochastic) can be used along the furrow. Wallender and Rayej (1990) presented the first attempt at developing a hydrodynamic furrow irrigation advance model using this approach with an explicit shooting algorithm. This explicit solution technique was used to solve for flow-area and flowrate on a cell-by-cell basis in the upstream direction, given the location of the wave front and an initial value of the time-step.

The authors contend that since flow in surface irrigation is typically subcritical, downstream conditions can propagate upstream and this type of solution technique was ideally suited to this situation. They contrasted this against the more common two-point boundary value solution using the double sweep technique, implying (without supporting evidence) that this latter approach would not handle this situation very well. They added that the explicit shooting algorithm simplified the coding by decoupling the model equations allowing greater flexibility in resolving instability problems at locations on the solution grid where hydraulic conditions change. They criticised the double sweep technique that at that time was solved using fixed time-steps without the ability to fix node locations. However, many of the authors' underlying hypotheses for developing the model were later shown to be unfounded.

Bautista and Wallender (1992) later expanded on this research to include the storage, depletion, and recession phases of the irrigation, and to simulate simultaneous advance and recession. The authors presented a numerical analysis showing that the specified space solution is computationally more efficient than the traditional specified time solutions. However, computation times were in the order of minutes rather than seconds, which is an order of magnitude larger than the implicit double sweep methodologies. Convergence issues were also identified and the authors suggested different research options to improve this. While results suggested that the model was successful, no evidence could be found that the model was developed further, possibly due to the problems associated with long computation times.

Greek researchers Sakkas, Bellos and Klonaraki (1994) developed a model based upon the complete hydrodynamic equations for all phases of the irrigation cycle. The explicit two-step numerical solution technique of MacCormack and Warming (1973) was used to solve the equations, proving to be the main "new" information presented. As usual, the model provided "satisfactory" results. This explicit technique allowed the decoupling of surface and subsurface flow models, which for simplicity can then be operated in sequence. A simple Kostiakov-like infiltration equation was used in the presented example but reference was made

of some preliminary work which used the physically based Philip equation; a benefit of the decoupling process.

This benefit was highlighted two years later by Indian researchers Singh and Bhallamudi (1996), who identified the flexibility of the explicit finite difference method over the implicit technique for handling different infiltration equations. They developed yet another hydrodynamic model which was solved using the explicit MacCormack method. Interestingly, no reference was made to the work of their Greek colleagues. However, they did use a different infiltration model, that of the Parlange equation (Haverkamp *et al.* 1990). The Kostiakov equation was also employed in a separate variation of the model. A simple sub-grid technique was introduced to implement a small grid size at the location of the wetting front to avoid the occurrence of negative depths, while maintaining a courser grid at other locations to improve computational efficiency. The amount of effort by the researchers to improve solution speed highlights a deficiency in the model; that of long computation times. While there is no doubt that today's modern computers would handle the explicit techniques more quickly, they tend to be an order of magnitude slower than the implicit double-sweep techniques for the same level of accuracy.

A double-sweep technique was used by Tabuada *et al.* (1995), who coupled a two-dimensional infiltration model with the Saint Venant equations to simulate the interaction of surface water depth and soil water movement during all phases of the irrigation cycle. Richards' equation was used for infiltration as it takes into account the initial soil water conditions before the irrigation and the surface water depth during the irrigation. The model aimed to provide insight into the infiltration phenomenon involved in furrow irrigation, tracking the position of the soil-water wetting front through time. This allows greater investigation of different inflow rates, furrow spacings and furrow shapes to improve irrigation efficiency. However, computational intensity was deemed as a practical limitation of the model, which had very long computation times on powerful super-computers of that era.

The MacCormack solution method was also used by Dholakia *et al.* (1998) to simulate all phases of border irrigation using the hydrodynamic (HDFD), zero-inertia (ZIFD), and kinematic wave (KWFD) models. A key feature of the methodologies is that consistent discretisation in implementing the boundary conditions and pseudo viscosity has eliminated the need for special grid treatments (e.g. moving grid, deformable gird, subgrid techniques) for the advance and recession fronts. Computation times for the hydrodynamic model ranged from 75 sec to 300 sec on a 486 computer, which is nearly twice that of the kinematic wave model. Simulated outputs from the models were compared against measured data with good agreement found, although the authors suggest that the hydrodynamic model is the most suitable model for simulating all irrigation phases.

2.6 "Inverse" methodologies

Optimal design and management of furrow irrigation practices using simulation models requires accurate identification of soil infiltration and hydraulic roughness properties. These soil characteristics are one of the dominant factors

in determining the performance (efficiency and uniformity) of furrow irrigation applications and exert their influence by controlling the rate of advance of the irrigation water down the furrow or bay. Knowledge of the spatial average values is required for the optimisation of furrow irrigation management while the temporal range of values is required for field design.

Substantial recent work has been directed towards developing methods to measure the infiltration properties of the soil. A volume balance at the end of the event (through measuring inflow and outflow) such as that presented by Merriam and Keller (1978) is a useful check on infiltration parameters estimated during the event. Strelkoff *et al.* (1999) proposed that surface volumes measured over time could also be used in the estimation. Devices have also been specifically developed for measuring infiltration (e.g. Turral & Malano 1996).

However, it is the "inverse solution" methodology for determining infiltration parameter values that has generated most interest in surface irrigation; that is, determining the infiltration parameter values from the measured irrigation advance, and/or surface depth profile, and/or runoff hydrograph. An advantage of this is that parameters can be estimated in "real-time", before the irrigation has been completed. Over the last 40 years, many methods have been developed to solve the inverse problem differing in their data requirements, assumptions, ease of analysis and accuracy (Strelkoff and Clemmens 2001).

The inverse methods can be broadly sorted into three main categories. The first two categories involve those methods that use a direct application of the volume-balance (or sometimes kinematic-wave) equations, which are manipulated in some way in order to determine the infiltration parameters. These first two categories differ in the way in which the infiltration parameters are extracted, with the first including methods employing a graphically based procedure, while the second includes those that use a numerical solution technique. The third category involves those methods that require repeated simulations using an optimisation technique to minimise the error between the measured and predicted advance and/or surface profile measurements.

Note that several attempts have also been made to derive generalised infiltration relationships for different soil types. These include efforts by the US Department of Agriculture (1975), Merriam and Clemmens (1985), and Walker (1989). Soil roughness characteristics (Manning *n*) are also typically categorised for different hydraulic situations. While these generalisations may have some credit in field design, they are less suitable for irrigation management where spatial and temporal variability effects must be considered.

2.6.1 Graphical solution techniques for the "inverse problem"

Several methods for solving the inverse problem were developed in the 1950s and 60s which required the use of manual curve fitting techniques on graph paper, and were usually quite time consuming and labour intensive. Finkle and Nir (1960) developed a simple graphical procedure to solve a volume-balance model to calculate infiltration in borders. This method is based upon the inverse procedure of Hall (1956) and was aimed at improving on earlier work by Bauwer

(1957), which required a minimum of three sets of irrigation measurements (with different flow rates, volumes, and cutoff times) on adjacent borders. However, Finkle and Nir's method also required intensive measurements at many points inside the test border. It was later found that this method places much emphasis on the advance during the first few minutes of the irrigation, which is subject to field measurement errors (Turner and Clift 1984).

Philip and Farrell (1964) developed an analytical solution for the irrigation advance based upon a Laplace transformation of the Lewis and Milne equations (1938). This work provided the foundation for much of the later research in time-of-advance solutions (e.g. Or and Silva 1996) and numerically based inverse methodologies. Infiltration is estimated through plotting on graph paper, the Laplace-transformed volume-balance equation for large irrigation times, and calculating the slope and intercept to extract the sorptivity and saturated hydraulic conductivity of the Philip infiltration equation (Eqn. 2.7). A limiting assumption of constant cross-sectional flow area in time and space is applied. This method was later shown to be valid only for short times, failing to predict the correct long-term infiltration behaviour (Knight 1980), and was also criticized for its time-consuming procedure (Shepard *et al.* 1993).

This method was revisited by Norum and Gray (1970) who instead utilised dimensionless curves superimposed on nomographs to determine the infiltration coefficients through curve-matching, with the advantage that the complete advance curve could be used, instead of just points at large times. However, both methods are restricted to certain forms of infiltration equation and therefore have limited field application (Maheshwari *et al*. 1988).

An advantage of the previous two methods (associated with using the Laplace transformation methodology) is that no assumptions about the functional form of the advance equation are necessary to determine infiltration from the advance (Smerdon and Blair 1988). In earlier work, Gray and Ahmed (1965) had developed a different volume-balance methodology for calculating infiltration in border dyke systems. Power functions were used to approximate both advance and infiltration relationships with a least squares methodology used to calculate power-coefficients. The power function approximations proved to be a disadvantage of the method, along with the physical measurement and mathematical representation of surface water storage with time (Norum and Gray 1970). Nevertheless, the power approximations used in this methodology have been reused in many of the later techniques, including the industry-standard "two-point" method (Elliot and Walker 1982).

Another Laplace model was developed by Wilke and Smerdon (1965) which, like the method of Norum and Gray (1970), uses a dimensionless form of the Laplace transformed system equations and graphical curve matching to estimate the infiltration parameters. However, this method multiplies a surface profile shape factor by the measured upstream cross-sectional area of flow to calculate surface storage.

Christiansen *et al.* (1966) developed a graphical technique, which requires the plotting of advance data on log-log paper to obtain the coefficients of a power advance equation. Infiltration volumes (calculated as the difference between

inflow and surface volumes) are then plotted against time from which the Kostiakov infiltration parameters are then estimated. In a later study, Elliot and Eisenhauer (1983) found that errors in surface flow estimates could be as high as 46% using this method.

Detar (1989) presented a modification of Christiansen's work producing essentially the same results, but with a more direct approach. The concept of average opportunity time was introduced to plot the infiltration function directly from tabulated data, hence simplifying the methodology.

Cahoon (1998) used a kinematic wave model iteratively to systematically vary the Kostiakov infiltration parameters over a grid of input values. For each pair of a and k parameters, the predicted and measured advance and runoff hydrographs were compared. Solution of the field average combination could be interpolated from the charts. He observed that a wide range of infiltration parameters can lead to an acceptable coincidence between the measured and simulated advance and runoff hydrographs. Simultaneously fitting both advance and runoff data provided a smaller solution space than fitting each singularly.

2.6.2 Numerical approximation techniques for the "inverse problem"

Numerical solution techniques to the inverse problems became popular in the 1980's when computers became more accessible. Some methods relied on numerical solutions to the previous graphical procedures, while others employed particular assumptions to simplify the equations so that an analytical solution to the infiltration parameters could be found. These numerical techniques were typically only designed for determining the infiltration parameters, and not the hydraulic roughness parameter.

Lal and Pandya (1972) developed a simple volume-balance technique to estimate the Kostiakov-Lewis infiltration parameters. The method was designed to be programmed into a computer and uses least squares fitting to determine the coefficients to an exponential form of advance equation, and also the infiltration parameters. Extensive field measurements of advance and surface storage were required (suggested at 20m intervals) over time. The accuracy of the solution is determined by the extent and accuracy of field measurements taken. Maheshwari *et al*. (1988) found that this technique places much emphasis on the beginning of the irrigation, when accurate measurements are hard to obtain.

Burt *et al*. (1982) developed a similar volume-balance approach to the previous method using numerical integration. The method is more numerically intensive than the "two-point" method and it requires the surface profile to be measured at several locations along the furrow. However, it still requires only two advance measurements, taken at the middle and end of the field. Elliot and Eisenhauer (1983) found that errors in estimated surface volume were around 3%.

Elliott and Walker (1982) and Elliott and Eisenhauer (1983) developed the "two-point" method, which has become the most popular procedure for solving the inverse problem. Its simple numerical solution technique makes it suitable for hand calculations, as well as implementing into computer code. This method

incorporates the modified Kostiakov-Lewis equation into a volume-balance model to solve for the infiltration parameters *a* and *k*. The final infiltration-rate f_o must be estimated or measured separately. Input data includes the cross-sectional area of the flow at the upstream end of the furrow or bay. Only two irrigation advance points are required. These are used to generate two non-linear volume-balance equations that are solved for the two unknown infiltration parameters. In the process of generating these equations, a simple power equation is used to represent the advance. A logarithmic transformation is used to linearise the volume-balance equations giving two linear algebraic equations in two unknowns.

Smerdon *et al*. (1988) and Blair and Smerdon (1988) expanded on the work of Elliott and Eisenhauer (1983). Six forms of "two-point" volume-balance methods (including three infiltration equations and two advance equations) were evaluated before suggesting a simple and direct method based upon the Kostiakov infiltration equation and the power advance equation.

Clemmens (1991) used a modification of the double-sweep technique commonly used in solving the continuity and momentum equations to determine one global parameter at each time step. The modification derived double-sweep coefficients for the Kostiakov infiltration parameter *k*, the Manning *n*, and global time-step. Another procedure allowed for determining Kostiakov *a* and *k* during the advance. However, it was subsequently found that these attempts were subject to errors caused by assumptions at the advance tip with regards to surface storage volumes. Nevertheless, this lead to further research in real-time control application where advance measurements were used to estimate the trade-off between infiltration and roughness using Bayesian statistical method (Clemmens and Keats 1992).

Renault and Wallender presented a series of papers (1991, 1992 and 1994) on determining infiltration using advance rates (rather than times) in a methodology they called "ALIVE" (Advance Linear Velocity). They developed a "time of advance-rate" equation using the Laplace transform of a flow-rate-balance equation (instead of a volume-balance equation) using a methodology borrowed from Philip and Farrell's (1964) solution of the flow-volume equation. They derived a function for advance-rate with two exponential terms, using a Horton (1940) law to represent infiltration. When using this to solve the inverse problem, four characteristics of the measured advance-velocity diagram were used to calculate the two Horton-infiltration parameters and one surface storage parameter. This process involves fitting two linear equations to the advance-velocity diagram: firstly for the initial rapid advance that occurs along the top end of the furrow, and then to steadier advance for the remainder of the furrow. In practice, the rapid advance phase could not be accurately measured. The authors promoted this method for its ability to determine the steady state infiltration term without having to measure runoff. In later work (1994), they demonstrated how this technique could be used to detect and evaluate heterogeneous soil properties at different sections of the field.

Shepard *et al*. (1993) developed a simple one-point method using a volume-balance equation and over-conditioning the advance and infiltration functions. The method uses the Philip infiltration and power advance relationships to derive

a simple algebraic expression to determine the Philip infiltration coefficients using only one measured advance point (preferably at the end of the field). To derive this expression, a constant value of 0.5 was used for the exponent term in the power advance equation. This assumption would have very limited applicability on most soils. Khatri and Smith (2005) found that the method continually failed to provide a reasonable prediction of cumulative infiltration, and it also failed to match the measured advance curves. It was found to under-predict infiltration at all times up until the final advance time.

Scaloppi *et al*. (1995) developed a volume-balance methodology to determine the Kostiakov or modified Kostiakov infiltration parameters. This approach requires advance and/or runoff data, resulting in three different procedures to determine infiltration. Some mathematical approximations are suggested to simplify the amount of field measurement and numerical computation. The results from the three procedures were found to vary considerably, with most variability found with smaller advance times.

Valiantzas *et al*. (2001) developed their own one-point method using a power advance equation and the USDA infiltration function (Eqn. 2.10) with a constant value for the parameter c. A differential algebraic relationship between the infiltration and irrigation parameters was derived requiring a Newton-Raphson iterative procedure to solve for two infiltration parameters. Khatri and Smith (2005) found that like the previous one-point method, this method under-predicts infiltration up until the final advance time, and poorly represent the measured advance. It was found that this method cannot describe initial high infiltration rates (including crack-fill) because of the fixed value of the infiltration parameter C. Allowing parameter C to vary resulted in improved performance of the method.

In earlier work, Valiantzas (1994) was able to determine both roughness and infiltration parameters in border irrigation through a simple iterative analysis using both the full advance, and surface depth measurements at a single station. Approximate estimates of the infiltration parameters were obtained using simple algebraic equations and are successively corrected using a zero-inertia model.

Other techniques have been developed by Singh and Chauhan (1973), Reddell (1981), Clemens (1982), Ottoni and Warrick (1983), and Scaloppi (1984), Izadi *et al*. (1988), which have received relatively little recognition from other authors in the literature.

2.6.3 Optimisation-based techniques for the "inverse problem"

The most promising method for solving the inverse problem is through repeated simulations using an optimisation technique to match predicted and measured quantities. Many methods have been developed since the 1980's making use of the rapid rise in computing power, and optimisation technologies. These methods can be categorised into those that employ time-of-advance equations, and those that utilise full irrigation simulations.

While the time-of-advance methods usually provide rapid solutions, they are often limited (by their analytical structure) by the forms of infiltration equations that they can employ, and also by which objective-functions (and hence measured data) can be used. Typically, they are based upon a volume-balance or kinematic-wave approximation to the full hydrodynamic model, which can lead to errors under certain field situations.

Using optimisation with complete simulations to solve the inverse problem has the advantage that there are no restrictions on the objective-function and solution parameters that are used. However, the main limitation is in speed and radius of convergence in the optimisation. Complete simulations inherently have some amount of noise in their results resulting from the discretisation process in the solution procedure. This can lead to convergence problems using sensitive optimisation tools. Most of the tools developed have demonstrated these problems.

Maheshwari *et al.* (1988) adopted a Hooke-Jeeves pattern search optimisation algorithm to solve a volume-balance model. The objective-function was the minimisation of the difference between the measured and estimated infiltrated volumes. The model allowed for the adoption of any time dependant infiltration equation and also any form of advance equation. Data requirements included measurements of the advance, surface storage depths, runoff, channel geometry, and inflow. They concluded that the method showed promise.

Conjugate gradient and variable metric optimisation techniques were used by Katopodes *et al.* (1990) to determine three parameters from a zero-inertia model. Two of the parameters were a and k from the Kostiakov infiltration equation while the third parameter was the Manning n. The objective-function used was the minimisation of the error between the measured and estimated depths of flow on the surface. The method is limited in that it requires the measurement of the advance, surface storage depths, field slope, inflow, and channel width. The optimisation process required good initial estimates with convergence problems identified when solving for the three parameters. In another paper, Katopodes (1990) suggested that only one parameter can be identified from advance data only, and two to three parameters can be determined from surface depth profile measurements.

In later work, Yost and Katopodes (1998) readdressed the convergence problems of this method, implementing a “fixed-point” algorithm that permitted unconditional global convergence on the solution. This is a slow-converging but reliable optimisation process which can be switched to a localised gradient technique after several iterations to speed convergence. Both infiltration and hydraulic resistance parameters could be determined reliably using a zero-inertia model and calibrating against surface depth measurements. Parameter scaling, gradient modification and switching optimisation algorithms were required to make this method robust and efficient. This method was suggested to overcome what was seen as a general problem of limited radius of convergence associated with optimisation based inverse methods.

Walker and Busman (1990) used a Simplex optimisation algorithm to determine the modified Kostiakov-Lewis infiltration parameters using a kinematic-wave

model and measured advance data. Having demonstrated that the technique works, they applied it to mimic the situation of real-time control, in which the infiltration parameters are continuously recalculated as more and more advance data became available during the irrigation. The results showed that the parameters could be determined with sufficient accuracy from early advance data for situations of slow to linear advance rate. They neglected to discuss the efficiency and reliability of the Simplex method and any difficulties involved in the optimisation.

The real-time solution of the inverse problem was also investigated by Azevedo (1992) with applications for variable-inflow irrigations with feedback control systems. A kinematic-wave model was combined with a non-linear search optimisation algorithm for constant inflow irrigation that was later applied to the variable inflow situation. It was found that some non-uniqueness of the inverse furrow advance problem can exist, although this had little effect on computations of application efficiencies and runoff hydrographs.

A Marquardt optimisation algorithm was used by Bautista and Wallender (1993a) to solve a hydrodynamic model for the modified Kostiakov-Lewis infiltration parameters. The parameters were found by minimising the error between the measured and predicted advance times or velocities, the latter of which was the more successful. They concluded that solving for three infiltration parameters was too difficult, the result being overly influenced by noisy data. They only had confidence in their results when solving only for the Kostiakov a and k parameters.

Smith (1993) developed a method utilising the volume-balance model from the two-point method of Elliott and Walker (1982). In this method, the Kostiakov-Lewis parameters were found by minimising the volume-balance error using a Steepest Descent optimisation procedure. His results, although initially appearing to be considerably different from the results of the two-point method, produced a cumulative infiltration curve that was almost identical. However, unlike the two-point method, the steady state infiltration rate did not need to be measured as it was determined in the optimisation. Data required were the cross-sectional area of water at the upstream end of the furrow, inflow rate and three or more points on the irrigation advance. In practice, the method had a limited range of convergence, and slow computation times.

Hume (1993) presented a solution to the infiltration characteristic of a cracking clay soil using a regression approach to the volume-balance technique, and utilising automatic data gathering techniques. This technique enables the fitting of any form of infiltration function through a least squares regression approach.

A 'flexible tolerance' algorithm was developed to work with the $SRFR$ model by Calejo and de Sousa (1996) to estimate the Kostiakov Lewis infiltration parameters and hydraulic roughness parameter using advance and/or recession data. The flexible tolerance algorithm was chosen because of its global convergence ability. However, convergence problems were identified when optimising on two or more parameters. The most accurate results were achieved when using both advance and recession data, while significant errors were

recorded when calibrating on recession data alone. No evidence could be found that this method was developed futher.

Camacho *et al.* (1997) developed an infiltration parameter estimation (IPE) methodology for management and control of furrow irrigation in real time. The method is based upon a kinematic-wave model and a downhill simplex optimisation method optimising on measured and predicted advance. The method is unique in that the model can compute the spatial and temporal variability of infiltration due to variations in the wetted perimeter. The same model can then be used to simulate the irrigation and suggest a cutback rate and time-to-cutoff. The simulated results were compared against *SIRMOD* output with some discrepancies. Nevertheless, this represents one of the few instances where the same model is used for both calibration and simulation.

Walker (2005) presented a stepwise multi-level optimisation scheme to calculate infiltration and roughness parameters, for any form of simulation model. In an effort to simplify field requirements, the procedure requires inflow and outflow hydrographs, but does not require individual advance measurements. Through an understanding of the parameter verses response sensitivities, a "decomposed" or stepwise multilevel approach to estimating the parameters is used, as opposed to searching automatically and simultaneously for all parameters within a feasible parameter range. Potentially, this offers the advantages of simplicity, stability and ease of implementation at the cost of long computation times. It could be implemented manually using the existing *SIRMOD* and *SRFR* software, although it could easily be automated. The author suggests, "*the most important advantage of the multilevel approach is that it is easier to manage and control periodic convergence failures within the simulation model*" (Walker, 2005, p131).

However, the piece-wise nature of the optimization has potential problems. For example, as each parameter is optimised/calibrated in turn, the previously adjusted parameter no long longer represents the optimal condition. Given this, it is questionable whether the final calibrated figures are optimal, and it is concerning that the author infers that the inherent accuracy is due to estimating different parameters from different parts of the irrigation. Probably, the greatest influence on accuracy is that the method includes a greater proportion of the irrigation response on which to calibrate, so that the calibrated parameters will naturally be more accurate than simpler "advance-only" methods.

Both advance and runoff measurements were included in the inverse solution of Gillies and Smith (2005). The method employs a volume-balance model using the optimisation algorithm developed and presented in Section 4.4.2 of this dissertation. Objective-functions are developed for advance-only, and then combined advance-runoff situations. In the advance-runoff example, the objective-function uses runoff-volumes rather than runoff-rates, and weighting parameters are introduced to change the sensitivity of the individual objective-function components. Results of their study suggest that infiltration can be calculated more accurately when both advance and runoff data are collected. In effect, this enables an extrapolation of the infiltration curve to greater times. The dual methodology is limited to advance and storage phases of the irrigation, and cannot be employed during recession phases. In later work, Gillies *et al*. (2006)

modified the technique to account for variable inflow irrigations by introducing an accumulated inflow term, instead of the previous average inflow assumption. They found that this provided accurate results under variable inflow irrigation so long as it is not applied where inflow changes rapidly. It cannot be applied to traditional cutback inflow irrigations.

2.7 Optimisation of furrow and border irrigation design and management

Optimisation of surface irrigation design and management practices has historically been undertaken through trial and error over many seasons, and through extensive field experimentation. Simulation models have provided a new tool to optimise practices, although this is not a straightforward process, with considerable user input required to run multiple simulations. Relatively little research has been undertaken to simplify, and/or automate this process. However, other options are also available.

In general, there are four types of methods available to determine optimum design and management parameters:

- Human-based learning (or "action learning");
- design charts;
- simulation models; and
- automated feedback control systems.

Research into these methods is reviewed below.

2.7.1 Human based learning for optimising design and management

As the transfer of computer-based technology is still in its infancy stage, the trial and error approach to improving design and management practices is still encouraged at the farm level with an emphasis on measurement and quantification of performance. A survey by Maheshwari and Patto (1990) showed that most Australian irrigators "guess" the design variables (flowrate, length, and slope) that dominate surface irrigation performance.

A "technology-gap" exists between farming and research organizations, which extension officers worldwide are trying to bridge through demonstration field trials and participatory action groups. While this aspect of improving design and management practices is not a focus of this dissertation, it does warrant a mention because it is a useful medium to communicate practical findings from decision support system outputs. Also the decision support software provides a means to measure changes in practice.

Field research in Australia (e.g. Raine and Bakker 1996; Raine & Shannon, 1996) has revealed a range of simple inexpensive measures, which can improve performance and are attractive to farmers. A range of methods to improve application efficiencies were identified and grouped according to whether they modified the soil, water or design parameters. Results showed that water use could be reduced by 50% through modification of field-length, time-to-cutoff, water inflow rate, furrow shape and cultivation practices. While the value of these results should not be underestimated, the techniques employed in this sort

of research are expensive and time consuming and are limited to a narrow range of conditions

2.7.2 Design charts for optimising design and management

Design charts (or field design and management guidelines) are a paper-based design and/or management tool, and were one of the first support tools for surface irrigation decision-making. Historically they have been developed from field trials, empirical relationships, and simple analytical functions. Recently, simulation models provide a more convenient means of developing these charts. They are presented as contours or three-dimensional surfaces of performance plotted against the decision variables.

Before personal computers were readily available, design charts developed from field experimentation provided a simple means to design irrigation fields. Hall (1960) developed a simple graphical method using the advance function to design border checks to achieve maximum application efficiency.

Strelkoff and Shatanawi (1985) produced a series of dimensionless normalised graphs (based upon generalised "ultimate outcome solutions") for wide, sloping, plane, free draining borders. With the use of a calculator, these curves allow the determination of the final distribution of infiltration water, runoff volume and efficiency for any combination of management (required depth of infiltration, flowrate, time-to-cutoff) and field parameters (bottom slope, length and roughness, infiltration parameters).

Zerihun *et al.* (1993) developed design-management "nomographs" for free draining graded furrows. This represents plots of efficiency, time-to-cutoff and uniformity coefficient contours in a length-flowrate space for a given set of field parameters. The nomograph can be used to determine the combinations of length, flowrate, and time-to-cutoff for an optimum combination of efficiency and uniformity.

One of the most successful attempts at guideline generation was through the development of the *BORDER* software application (Strelkoff *et al.* 1996). *BORDER* was originally released as a *DOS*-based design and management tool for border irrigation with tailwater runoff. It consists of a stored database of pre-run irrigation simulations with an algorithm for retrieving and displaying the results for a range of design and operating parameters. The outputs are presented in the form of contour plots of selected irrigation performance measures for different combinations of design and management parameters. It has recently been incorporated into the *WinSRFR* decision support system.

In more recent research, Hornbuckle *et al.* (2003) used the *SIRMOD* simulation model to develop design charts to demonstrate a potential application of the software. These charts were designed to present application efficiencies, distribution uniformities, infiltration volumes and runoff volumes for different combinations of inflow and time-to-cutoff. The authors contend that by recording irrigation properties for a previous irrigation season, these charts can be created and used in the following season to improve irrigation practices. This assumes that infiltration characteristics remain constant over the season and is

acknowledged as a limitation of the method. Previous work by Hornbuckle (1999) demonstrated that usually only small differences in the infiltration characteristics occur after the first irrigation of the season. Nevertheless, they recommend that infiltration characteristics for both the first irrigation and the later irrigations be used to represent infiltration over the season.

2.7.3 Computer optimised practices for design and management

Computer simulation models offer the greatest potential to optimise design and management practices. Optimum parameter combinations can be determined through repeated simulations using existing simulation models, although the quality of these results is largely dependant on the operator's skill in using the model. Acknowledged reliability problems and high complexity of existing models have hindered efforts to accommodate an optimisation algorithm to remove the dependence of a skilled operator.

The few existing self-optimising models are limited in their range of objective-functions and optimisable parameters and have failed to reach their potential. Self-optimising models have long been considered the last stage in model development by irrigation researchers, but the decisions derived from such tools are valid only as long as conditions remain constant in the field. Variations in field parameters such as soil infiltration must be considered when optimising.

Before 1990, the optimisation of irrigation design and management practices using computer simulation software was limited by the need to apply a trial and error approach. Since then, very few attempts have been made to develop automated optimising capabilities into a surface irrigation decision support system. None have succeeded to achieve practical use. To understand the difficulties involved in developing such a system, one should consider that even the most popular simulation tools require considerable user input and guidance.

Geometric programming techniques were used by Reddy and Clyma (1981) to optimise the design of free draining borders while considering net economic benefit. Similar methods were used by Holzapfel *et al*. (1986) who used a linear programming economic model to optimise the design of free draining borders. They used a log-transformed objective-function to maximise the profit of the crop. A year later Holzapfel and Marino (1987) resolved the problem using a non-linear optimisation technique. In these last two examples, the soil was considered homogeneous, while relationships between irrigation performance and design variables (inflow, cutoff time, and field-length) were derived through regression analysis.

Smerdon and Blair (1987) made the first serious attempt at combining an optimisation algorithm with a hydraulic model to optimise irrigation efficiency. They combined a kinematic wave model with the golden section optimisation method to determine the time-to-cutoff.

Singh *et al*. (1987) developed a model for optimisation of inflow rate and time-to-cutoff in closed end borders using the Strelkoff zero-inertia model (1985) and a quasi-Newton optimisation algorithm. They developed an objective-function based upon maximising the "deficit/excess efficiency" term by Blair and

Smerdon (1988). In effect, this function attempts to minimise runoff and deep-drainage components of the irrigation. While their results would suggest that the optimum design parameters were identified successfully, the authors neglect to mention any difficulties associated with the optimisation process.

Wallender *et al*. (1990) used a volume-balance simulation model (including a runoff water recovery component) to provide input into a profit calculation model to maximise profit as a function of inflow rate and irrigation time. Objective-function response-surfaces were generated to determine the optimum design values, instead of using an automated optimisation algorithm.

Another alternative to automated optimisation was in the form of a regression-based model for border irrigation called $BICADM$ (Maheshwari and McMahon 1991; Maheshwari 1994). Multiple regression analyses on the input and output data from $SRFR$ simulation model were carried out resulting in an approximation to the parent model, free from its associated reliability and time problems. The accuracy of the new model was comparable to the original, demonstrating that a mathematically complex model such as the Strelkoff simulation could be used to develop a simpler model for specific field conditions. It was suggested that this type of model would be ideal for optimisation purposes, relieving the computational load of the original model while maintaining its accuracy

The optimal management of a cutback furrow irrigation system was analysed by Bautista and Wallender (1993) using a cost minimisation objective-function subject to achieving a specified proportion of the irrigation requirement. A kinematic-wave simulation model with wetted-perimeter dependant infiltration was combined with an economic model to formulate the objective-function and a Box optimisation algorithm used to undertake the search. Response contours of the objective-function were plotted against the decision variables to analyse different cutback strategies and economic settings. Results showed the response-surface to be insensitive to changes in the decision variables around the optimal solution and the authors acknowledged some convergence problems. Both discrete and continuous cutback functions were investigated with little difference in performance resulting between the two.

Ito *et al.* (1999) used a kinematic-wave simulation model in conjunction with a Box and constrained grid search optimisation algorithm to maximise economic "return to water". This research was aimed at investigating the effect of a lack of infiltration and furrow geometry data on the design and economic return to water for furrow irrigation systems. Optimisations were carried out for both actual and partial information cases and monetary loss due to lack of infiltration data was calculated. Monetary loss was found to be lower for systems with high inflow rates.

An optimisation algorithm was not used by Valiantzas (2001) who addressed the furrow irrigation design problem through an analytical time-of-advance solution. For a specified length of furrow, the inflow rate (and time-to-cutoff) could be found using a simple algebraic equation to minimise the cost of the furrow system (independent of water and labour costs) in terms of the inflow volume. The results were validated against the optimum values obtained from a zero-

inertia model and were in close agreement. The method was also extended to solve for the optimal furrow length.

2.7.4 Real time automated control

Recent technology advances have provided numerous mechanical devices to aid irrigation design and management. These include measurement devices used to monitor soil moisture, flow rates, depths and irrigation advance, and laser-grading machines to accurately level the field. Variable rate inflow valves exist for use in cutback and surge irrigation while efficient delivery systems exist to transfer the water into the furrows. Of particular interest is the recent development of automated control systems for furrow irrigation.

Several real-time automated control systems have been developed for irrigation management. There is no doubt that these systems will dominate surface irrigation in the future, but at present they are virtually untried and far too expensive to implement into existing systems. The advantage of these systems would be in achieving high performance results while reducing labour requirements. However, the use of simulation software to manage irrigations may produce similar performance results without the high capital and maintenance costs of automated systems.

Researchers Reddell and Latimer (Reddell 1981; Reddell and Latimer 1987; Latimer and Reddell 1989, 1990) developed the "Advance Rate Feedback Irrigation System" (ARFIS) that senses the advance of water at two stations down the field. These measurements are relayed to a computer through a telemetry system and used to calculate an infiltration function for input into a volume-balance model to calculate suitable management parameters. Once calculated, these results are sent to a flow control system that regulates the inflow to match the existing infiltration rate in a cutback procedure. Upon shutdown of the inflow valve, performance figures are calculated.

Katopodes and Tang (1991) developed a self-adaptive control system for surface irrigation advance. The goal of their system is to use sensors, a zero-inertia simulation model and an algorithm to adjust inflow rate to obtain an ideal distribution of water. An objective-function is constructed based upon minimising discrepancies between actual and desired advance rate. In their system, field sensors measure the water surface profile as water advances down the field. Infiltration and roughness parameters are estimated using this data, before the optimum advance trajectory for these conditions is retrieved from a database. An optimisation algorithm is then used to determine an adjusted inflow rate to improve the distribution of water. This continues for a few minutes before the process is restarted over again. The authors report "satisfactory" performance of the system given that most of the steps of the system rely on ideal conditions for data collection and implementing the control actions.

Hibbs *et al.* (1992) also used an adaptive control algorithm to automate furrow irrigation management (called FAAC). Using a flume and runoff depth sensor, infiltration is estimated in real-time using a volume-balance model before the computer adjusts inflow to control outflow at a desired rate. Tests showed that the adaptive control algorithm was accurate enough to restrict furrow outflow at

a desired rate. The results were compared to a constant inflow system, and the FAAC system was found to substantially decrease discharge and tailwater losses with a small decrease in cumulative infiltration. Application efficiencies were increased when the furrow was initially dry.

2.8 Decision support software for furrow and border irrigation

Very little of the research into simulation modelling, inverse methodologies, and design and management tools has been developed into user-friendly software applications. While a number of tools and ad-hoc constructions have been built for specific applications, very few have developed into serious decision support packages for surface irrigation design and management. Some early packages that were developed include *BASCAD* (Boonstra and Jurriens 1988), *BICADM* (Maheshwari and McMahon 1991), *FISDEV* (Zerihun and Feyen 1992) through to *SURDEV* (Jurriëns 2001), *BASIN* (Clemmens *et al.* 1995) and *BORDER* (Strelkoff *et al.* 1996). However the must successful software developments are the *SIRMOD* (Walker 1997), and *SRFR/WinSRFR* (Strelkoff *et al*. 1998; AARC 2006) software packages, which have both seen long-term and widespread use amongst research groups, and have had some practical application.

SIRMOD was developed at Utah State University to simulate both border and furrow irrigation for continuous flow irrigations as well as surge flow and cutback methodologies. It employs a full hydrodynamic model, as well as zero-inertia and kinematic-wave approximations. The ability of the software to accurately predict advance and recession in relatively short furrows has been verified by many authors including the developers of the model (Walker and Humphries 1983). *SIRMOD* was originally developed for research and teaching purposes and has been successfully used at both Utah State University and the University of Southern Queensland since 1987 (Raine and Walker 1998). The tool is continually being further developed.

SRFR is software from the United State Department of Agriculture (USDA) that has existed as a *DOS* program for over fifteen years, although it was recently redeveloped as a Windows program. *SRFR* employs a zero-inertia model and can accommodate a range of spatially and temporally varying input parameters including slope, furrow cross-sections, infiltration and hydraulic roughness. It is also currently serving as a platform for simulating constituent transport (Strelkoff *et al.* 2001). It has also been used in many research projects in Australia (e.g. Maheshwari *et al.* 1993a,b; Wood, *et al.* 1998; Hardie *et al.* 2002; Victorian DPI 2004).

Both of these tools are primarily simulation engines; that is they are specifically designed for simulating surface irrigation to determine irrigation performance. They can both be used manually to optimise practices but do not contain automatic optimisation capabilities. Also, they do not allow the solution of the "inverse problem" using the simulation model. However, both packages contain a built-in database of typical soil-infiltration properties, although the reliability of these for Australian conditions is doubtful.

In terms of accuracy, both packages have now been evaluated by several researchers (Maheshwari and McMahon 1993a; Raine and Walker 1998; Hornbuckle *et al*. 2003) who all rated the performance as satisfactory to good. For example, Maheshwari and McMahon (1993a) investigated the performance of *SIRMOD* and *SRFR* models along with four other border irrigation models. Over sixty irrigations were monitored and the models applied. It was concluded that the Walker model was the best for predicting advance times and the Strelkoff best for the recession. More generally, it was found that the models employing the hydrodynamic and zero-inertia approaches were the most appropriate. There was no difference in the results between the hydrodynamic and zero-inertia approaches of the Walker model. This supports the assumption that other authors have made regarding negligible effects of inertia terms in border irrigation. Maheshwari and McMahon (1993a) found that the kinematic-wave models had a tendency to underpredict the recession.

During the research for this dissertation, considerable time was spent using and evaluating these tools. The initial impression of both packages was that they are very powerful and impressive looking, at the expense of being overly complex and difficult to use. The interfaces were limited and relatively user-unfriendly. Both packages often crashed when unrealistic data was entered, while *SIRMOD* often required adjustment of the time-step parameter to achieve convergence. Considerable practice and experience with the tools was required to use them successfully.

While some people have found *SIRMOD* and *SRFR* to be stable, robust and fast (Garcia-Navarro *et al.* 2004), evidence of the angst experienced by other users is found in a paper by Maheshwari (1994). Having previously tested several of the leading models in "real" field conditions, he remarked that "*the use of <the model> for design purposes on cracking soils was found to be tedious and time consuming, particularly if the number of simulation runs required are excessive (say >10)*".

New *Windows* versions of both *SIRMOD* and *SRFR* software packages were released in 2006 (USDA 2006; AARC 2006), but not in time to be properly evaluated as part of this research. Both have gone to great lengths to improve flexibility and reliability of the software with many advanced features, including simple inverse-solutions, and design capabilities. For example, *WinSRFR* now combines the functionality of the individual *SRFR*, *BORDER* and *BASIN* software. Recent dialog with Professor Wynn Walker has provided some comment on these software programs stating, "*robustness <problems> has been improved but not eliminated… software packages tend to feel like the research model of old. Some model parameters remain and need to be hidden. The inverse solution tends to be simple, and, while the design algorithms are good, they do not optimise – optimal design still depends on trial and error. Both use a lot of code remnants from earlier versions for some components*." (Walker pers.comm. 2007).

2.9 General discussion

Several gaps in the research can be identified from the reviews on simulating surface irrigation, solution of the inverse problem, and optimisation of design and management practices. These will now be discussed in turn.

The literature review of simulation modelling methodologies reveals that not all areas of research have been given thorough treatment. For example, relatively few surface irrigation models have been created to simulate all phases of the irrigation cycle with most research directed towards modelling the advance (Xu and Singh 1990). Ironically, it is the advance phase that has been most successfully modelled and needs the least treatment. It is not uncommon for papers claiming to model a surface irrigation event, to only describe the procedure for the advance phase. To find an adequate coverage of the other phases, one must start to look in Ph.D. dissertations (e.g. Kafshgiri 1984) and in books such as Walker and Skogerboe (1987).

Problems associated with solution techniques have rarely been discussed. The computer algorithms and numerical solution techniques for solving the model equations are the most fragile part of the technology, especially with the hydrodynamic and zero-inertia models that are difficult to solve. Yet relatively few authors mention any performance problems associated with their methodologies. Accuracy is often discussed, but robustness is hardly ever mentioned. Also, the research typically focuses upon the mathematical treatment of the methodologies, but very few authors present algorithms for transforming the mathematics into its computer code equivalent.

It was also found that some authors have often benefited by segregating models for border and furrow irrigation and publishing separate papers for each. However, the model and solution techniques for the two irrigation practices are very similar with the differences occurring in the treatment of infiltration and wetted perimeter effects.

The review of the inverse methodologies has highlighted that the “two-point” method has been very popular with researchers, with several variations of the technique being presented. It has also probably been the most commonly used method in practice, and has regularly been used to calculate infiltration parameters for input into *SIRMOD* and *SRFR* (it is inbuilt in the most recent versions). However, the difference in model structures between the “two-point” method (volume-balance) and target simulation tools (hydrodynamic and zero-inertia) is a potential source of error. This was also identified by Walker (2005). Unfortunately, the literature review showed that very few of the suggested inverse techniques employ the full hydrodynamic model.

The literature review of optimisation of practices revealed little information about the nature of objective-function response that was involved with the methodologies. Investigation of response-surfaces is crucial in evaluating the performance of automatic optimisation capabilities. Instead, none of the automatic optimisation methodologies provided any clear evidence of successful solution. Once again, it appears authors have failed to report the limitations of their techniques.

In current practice, probably the most accessible way of optimising irrigation is still through a trial and error approach of repeatedly running the simulation. Manual optimisation requires a degree of skill on behalf of the operator, and can lead to problems caused by entering unrealistic parameter values. For example, during the case study presented in Appendix 2.2, *SIRMOD* was found to be sensitive to the range of input parameter values and the program often "crashed" with the input of unrealistic data.

2.10 Direction for developing a new decision support system for furrow and border irrigation.

Having performed a comprehensive literature review and an evaluation of the *SIRMOD* software, it was decided to base the new decision support system on a hydrodynamic simulation model using a methodology similar to that employed by *SIRMOD* and *SRFR*. However, the focus will be on automating this simulation engine for optimisation and system response evaluation. An inverse technique will need to be developed and incorporated into the system using the same hydrodynamic model, to avoid calibration using a different type of model. There is also a need to develop an optimisation algorithm to facilitate the calibration process and automatically determine the optimum design and management parameters. Response-surface generation facilities should be added to enable evaluation of the optimisation and calibration performance. This feature can then be used to generate design charts for irrigation management and design. The new system will need to refine the existing computational methodologies to improve system robustness, while maintaining accuracy. Data management facilities will also be required given the range of analyses that is potentially available to the system. This will deliver a decision support system that provides tools for each of the major decision support requirements.

2.11 Conclusions

This chapter has presented the concept of decision support systems for furrow and border irrigation. A literature review was undertaken into the three main surface irrigation research areas of simulation modelling, solution of the "inverse problem" and optimisation of design and management practices. It was found that gaps exist in the literature, especially with: simulating the later stages of the irrigation cycle; converting the mathematical model into computer code form; ensuring simulation robustness; calibrating using the complete hydrodynamic simulation model; parameter analysis of system responses; and automating the optimisation process. Functionality requirements for the development of a new decision support system were identified as simulation, calibration, optimisation, parameter-analysis and data management.

The remainder of this dissertation presents research into these "gap" areas while developing the functionality for a new decision support system for furrow irrigation.

Chapter 3 Development of a simulation engine for furrow and border irrigation decision support

3.1 Introduction

Review of the literature of furrow and border irrigation modelling in Chapter 2 has highlighted that existing surface irrigation models are accurate for many practical applications, but suffer from reliability and usability problems. The goal of this chapter is to overcome these limitations by developing a new simulation engine (henceforth known as the *FIDO* simulation engine) based upon the modification and refinement of existing techniques, which can be incorporated into a decision support system for furrow and border irrigation.

This research presented in this chapter has six main objectives: *(1)* it will outline design criteria considered when developing the *FIDO* simulation engine; *(2)* it will present the model and solution technique formulation, including the redevelopment of Preismann double-sweep solution technique for furrow and border irrigation into a simpler form; *(3)* it will develop an object-oriented algorithm capable of transforming the mathematics of this model and solution technique into a tool capable of being implemented into a modern user friendly decision support system; *(4)* observations about the simulation behaviour will be discussed; *(5)* four treatments to the simulation are presented to achieve simulation robustness; and *(6)* the simulation engine will be validated against the existing *SIRMOD* simulation tool.

This chapter is accompanied by two appendices containing the source-code (Appendix 3.1) and validation results for the simulation engine (Appendix 3.2).

3.2 Background to simulation engine design

To design a simulation engine for furrow and border irrigation, one must first understand what a simulation engine is, what it is composed of, and what the primary design-objectives are. Two main aspects of the design must be considered: *firstly* concerning the model and solution technique formulation; and *secondly* regarding the software algorithm design. Finally, an understanding of the complexity of the design task is fundamental towards overcoming barriers faced by developers in the past.

3.2.1 What is a simulation engine?

A simulation engine (for furrow and border irrigation) is a computer-based model, which mathematically predicts the physical processes of water flowing down a furrow and infiltrating into the soil over time. The simulation engine is not a "stand-alone" computer program built around a user-interface; rather it is only a sub-module of a computer program or decision support system for surface irrigation. Forming the central core of this decision support system, it is crucial that it be as accurate, reliable and robust as possible. The useability and

flexibility of all of the other components of the system depend upon the simulation engine being able to repeatedly and reliably deliver an accurate set of results without user intervention.

Simulation involves solving the hydrodynamic equations for flow rate and flow area during each irrigation phase either on a predefined grid, or on a grid defined during the advance phase. Combinations of different irrigation phases and grid design along with the presence of two possible furrow-end conditions leads to many simulation configurations. Robust solution of all configurations is required for inclusion of the engine in computer-managed processes (such as optimisation) where physically unrealistic input data may be encountered.

3.2.2 Elements of the simulation engine

The simulation engine is composed of three conceptual elements (Figure 3.1):

- differential (model) equations;
- a numerical solution technique; and
- a computer algorithm for managing the simulation.

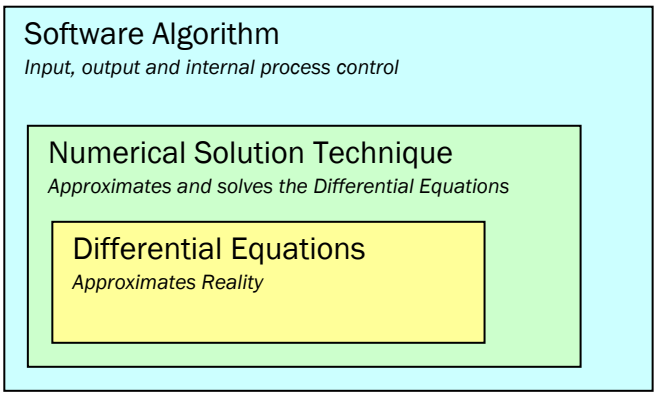


Figure 3.1: Fundamental Components of the Simulation Engine.

The *differential equations* (model equations) are a mathematical representation of the physical laws and processes of water flowing down a furrow and infiltrating into the soil. These processes are very complicated to describe mathematically, and the equations take on a differential form making them very difficult to solve. Therefore a *numerical solution technique* is required to solve the equations iteratively by approximating the differential terms, linearising, and solving the resulting set of simplified equations. The computer or *software algorithm* forms the controller or manager of the simulation. Its purpose is to oversee the operation of the solution technique and provide input/output functionality.

Past surface irrigation research has focussed mainly on the first two of these elements, with less emphasis on the logistics of converting the mathematical equations, simulation options and constraints into their computer language equivalent.

3.2.3 Objectives of simulation engine development

The primary goal of the work in this chapter is to develop a simulation engine capable of being implemented into a decision support system for furrow and border irrigation. This involves (a) the refinement and modification of existing techniques in dealing with the model equations and solution techniques, and (b) the development of a new object-oriented computer algorithm for controlling the simulation.

The target decision support system is required to support operations such as optimisation, calibration and response-surface generation for a range of management operations. For this to occur, three primary objectives of the simulation engine must be achieved:

- It must be accurate, robust and reliable;
- It must be flexible in handling a range of physical scenarios and conditions; and
- It must be reusable in a variety of applications within a decision support system.

These objectives will now be discussed in more detail.

Accuracy, robustness and reliability.

Accuracy, robustness and reliability are three criteria used to assess the performance of the simulation engine. These are related more to the underlying mathematical model and numerical solution techniques rather than the engine's computer algorithm.

Accuracy relates to how well the simulation model replicates the physical processes of surface irrigation. Factors that can influence the accuracy of the simulation engine include:

- Limitations in the mathematical model: For example, the full hydrodynamic model is considered more accurate than the volume-balance model for simulating more conditions in surface irrigation;
- Errors resulting from the discretisation process in the solution technique;
- Numerical approximations: For example, an approximation is used during the recession phases of the simulation whereby upstream cells are removed when the depth of flow is less that 5% of the normal depth. Although the flow is still finite, an approximation of zero depth and zero flow is then used for the following time-step; and
- Poor grid refinement. Simulation detail may be missed if the time-step or distance-step is too large.

Robustness is a somewhat ambiguous term with different meaning to different people and contexts. According to Strelkoff and Falvey (1993), robustness implies an absence of sawtooth fluctuations in the numerical results, even under severe flow conditions. The ASCE Task Committee on Irrigation Canal System Hydraulic Modelling (1994) suggested that a robust model was simply one that could continue the numerical solution through potentially troublesome circumstances without encountering errors, which could cause the program to terminate.

Reliability combines the attributes of accuracy and robustness and implies a proven ability to consistently deliver accurate simulation results over a period of time and conditions.

Flexibility

Flexibility defines the ability of the simulation engine to successfully model a range of scenarios. This could include different management techniques such variable inflow methods, blocked furrow irrigation, or it could mean handling spatially variable input parameters.

Reusability

Reusability is an attribute of the computer algorithm in relation to the simulation engine's ability to be used in a variety of contexts. For example, the same simulation engine should be able to be used in simulation, calibration, optimisation and analysis roles, or even be used as a sub-model in a larger modelling toolkit.

3.2.4 Model and solution technique considerations

As outlined in Chapter 2, a variety of numerical techniques and treatments exist to solve the hydrodynamic equations for surface irrigation simulation. Therefore many decisions need to be made when formulating the underlying model and solution techniques used in the simulation engine:

- *Which form of the hydrodynamic equations should be used as the basis of the model: full-form, zero-inertia, kinematic-wave or volume-balance?*
- *Should a dimensionless form of the equations be used?*
- *What are the irrigation phases that need to be modelled?*
- *Should the simulation be one-dimensional or two-dimensional?*
- *What sort of approximation to the partial differential equations should be used: finite differencing, finite elements or something else (like neural networks)?*
- *What sort of coordinate system should be used for the discretisation process: regular grid, moving grid, or method of characteristics?*
- *What type of solution technique should be used: an implicit technique or explicit technique?*
- *During the advance phases, do we use a fixed time-step and solve for the advance distance, or do we use a fixed distance-step (predefined grid) and solve for time?*
- *During the recession phases, do we try and solve for the recession trajectory, or use an approximation methodology?*
- *How do we accurately model rapidly changing conditions (inflow on, inflow off, transition to runoff) without violating stability constraints (such as "Courant")?*
- *What boundary conditions need to be applied?*
- *How should infiltration be treated?*
- *How do we set parameter constraints without compromising numerical-stability?*

These questions (and those to follow) will be addressed throughout the remainder of this chapter.

3.2.5 Software algorithm design considerations

Very little information has been published by irrigation researchers on how to transform the mathematical hydrodynamic equations and iterative solution techniques into their equivalent computer code (some have presented programming algorithms, including Schmitz and Seus 1991 and; Tabuada *et al*. 1994). One could then wonder how closely modern programming practices have been adhered to in these transformations given that the latest generation of surface irrigation software has been criticised as being unreliable.

These days, almost all software-engineering (with graphical user interfaces) is undertaken using object-oriented programming (OOP) techniques, given the power and flexibility that this methodology offers (the reader is referred to texts such Riel 1996 and Coad *et al*. 1993 for more information). OOP was chosen over procedural techniques for the development of the simulation engine and *FIDO* decision support system.

Good OOP design is a difficult skill with many complex decisions required during the initial design stage. Key questions arise such as:

- *How should the software components be modularised? Should the objects be data-centred (categorised based upon data considerations), or model-centred (categorised based upon irrigation characteristics such as phases)?*
- *How should the data objects be formulated: what information should they store, and how much processing ability should they encapsulate?*
- *How should memory be managed given that many megabytes of data could be generated?*
- *How should the engine cope with any errors that arise?*
- *How can switching between irrigation phases be handled: using memory pointers, conditional statements, OOP "virtual methods" (a powerful technique utilising OOP's "polymorphism[2]" capabilities)?*
- *Which parameters should be made available to the program interface?*
- *How much of the design should be accessible (using OOP "scoping" techniques) to future developers so that modification and expansion can be easily undertaken?*

3.2.6 Programming complexity issues

A single mathematical equation is usually quite simple to translate into its programming language equivalent form. Consider however, if there are dozens of equations that need only slight changes under certain conditions. Then the computer algorithm will need to be modified to account for these variations using logical and conditional statements. Note that it would be very dangerous to have multiple copies of the main algorithm with slight variations. As well as increasing the size of the code, this can easily introduce errors if part of the main algorithm needs to be modified since the changes would have to be made in more than one place in the code. Good programming practices dictate that each equation should be written only once and that conditional programming statements should be used to apply these equations in the appropriate order. The effect of all of this

[2] See www.wikipedia.org for further information.

is that the computer code quickly becomes more complex than the initial set of equations. The complexity of the computer code grows proportionally (and sometimes exponentially) with the number of variations in the model.

Considering this, the problem of simulating water flowing down a furrow is a very complex programming exercise. It is a very dynamic problem with the programmer not knowing beforehand the number of time-steps, distance-steps and iterations that will occur, and even which irrigation phases will need to be simulated. The number of input variables will also vary for different users with the requirement that a field could be broken up into several reaches of different slope, geometry, roughness, and infiltration characteristics. Furthermore, there are also a range of management techniques that must be considered such as different inflow methods (constant, variable, cutback, or surge) and furrow-end treatments (blocked or free draining).

Despite all these variations, researchers have typically presented very little information on how to replicate these conditions in the software algorithms. In terms of the research community, it could be assumed that this is seen as programming problem, rather than part of the hydraulic engineering task. Only those who have taken the research past the academic level can appreciate the effort required, especially given the immature software engineering technologies that were available until recently. Nevertheless, the lack of professional software engineering expertise into the development of the computer algorithms remains a major reason why surface irrigation models have had reliability and flexibility problems.

3.3 Model and solution technique formulation

The first step in developing a simulation engine for furrow and border irrigation involves defining the underlying model and solution techniques. This includes identifying model inputs and outputs as well as describing the coordinate system for the discretisation process. Key model and solution technique equations must then be derived for main body of the simulation, for the starting calculations, for different furrow end treatments, and for the simulation termination. Boundary conditions, initial parameter estimates, and parameter ranges must also be identified.

3.3.1 Choosing the underlying model

The full form of the one-dimensional hydrodynamic equations (*Saint Venant Equations*) for open channel flow was chosen as the basis of the underlying model in the *FIDO* simulation engine. The inclusion of these equations instead of the simpler and easier-to-program *zero-inertia* form of equations was justified based upon:

- The findings of the literature review conducted in Chapter 2 that identified the advantages of the more complex full hydrodynamic models in terms of accurately simulating a wide range of conditions;
- The proven ability of the Walker model (which also uses the full form of these equations) which has been shown to be accurate under many conditions (Maheshwari *et al.* 1993a and 1993b; Hornbuckle *et al.* 2003); and

- Increased computing power (leading to faster simulations) and modern software engineering techniques and diagnostic tools, which have overcome many of the obstacles faced by researchers in the past when using this type of model (many of who typically defaulted to the simpler equation forms).

The full form of the hydrodynamic equations is essentially a set of hyperbolic partial differential equations (the most difficult type of partial differential equations to solve) with no known analytical solution. Therefore, a numerical procedure is needed to approximate the differential terms in the equations in order to extract the simulation information. The furrow reach is broken up into finite cells, and the solution for each cell is then computed over incremental time-steps. This is the most complex process in the system, and the success of the simulation engine is heavily dependant on the power and efficiency of the solution technique used.

Chapter 2 highlighted that much of the early surface irrigation hydrodynamic modelling research (and continuing today) utilised a dimensionless form of the underlying equations in order to reduce the number of independent parameters that they contain. This was deemed important in the early days as computing power was very limited, and many different techniques were employed to reduce the computational load. However, the trade-off from this was that extra processes were involved to dimensionalise and non-dimensionalise the inputs and outputs, and the solution parameters on their own, had no physical meaning. The true benefit of this is questionable. The main issue raised was that the added complexity of dimensionalising/non-dimensionaling negates the (possible) benefit of improved computational performance. Also, there is little evidence (other than anecdotal evidence) to prove that there is any real performance benefit. Elliot *et al*. (1982) experimented with the dimensionless form of equations in early research, however, Walker's *SIRMOD* model (which has roots from this early work) did not utilise this form of the equations. Therefore, the dimensionless form of the equations was not used in the simulation engine developed in this chapter.

3.3.2 Choosing a numerical solution technique

The implicit Preismann double-sweep technique (Liggett and Cunge 1975) is redeveloped in this chapter into a more generalised form. The review of the literature in Chapter 2 has highlighted the potential of the double-sweep method in terms of simulation speed and success. The method uses finite differencing to approximate the differential terms in conjunction with a Newton-Raphson procedure to linearise and solve the equations at each time-step. The resulting matrix is banded and solved using an efficient Gaussian solution technique that bears the name of the method – the "double sweep" technique. The method results in coefficients being calculated in a forward sweep starting from the upstream cell before the model parameters can be calculated in a backward sweep.

Use of an implicit solution technique such as this allows conditions at nodes to be determined simultaneously rather than on a node by node basis. This "supposedly" removes the burden of conditional stability that is inherent in the

explicit form of solution techniques (although this will later be shown in Section 3.6.4 to be unfounded). Many explicit techniques exist which have the benefit of being simpler to program, but have been found to be slow in operation and dependant on stability restrictions. For example, an explicit shooting method technique (based on the work of Wallender and Rayej 1990) was trailed during the early stages of this research, but was found to be an order of magnitude slower than the implicit double-sweep methods[3]. This finding is supported by other researchers publishing long execution times for their explicit solution-technique based models (Singh and Bhallamudi 1997).

Several variations of the double-sweep method have proven successful in surface irrigation modelling. However, the solution equations that have been published, such as those by Walker and Skogerboe (1987) with fixed-time-steps, and those by Strelkoff (1992) with fixed-distance-steps, are not in their simplest form. The simpler and more generalised form developed here is easier to convert into computer code, and allows either fixed time-step or fixed distance steps to be used, depending on the simulation requirements.

The formulation of the method developed in this chapter borrows heavily from both forms of the published equations. Both forms of the equations are similar even though one is “time-step-based” and the other is “distance-step-based”. Strelkoff’s equations have the addition complexity of a global unknown time-step parameter allowing for solution on a predefined grid. However, by removing this term from the Strelkoff algorithm, the equations are effectively the same.

The new equations developed in this work also include the global unknown time-step parameter, giving the option of solution at fixed node locations. However, in practice, the *FIDO* simulation engine rarely makes use of this feature, normally using fixed time-steps and solving for the advance distance.

At first glance, it may seem more sensible to have a predefined solution grid and solve for the unknown advance time to each of the nodes. This would allow nodes to be located at points of interest down the furrow. For example, nodes could be placed at locations where field measurements were captured such as advance time, flow-rate or flow-area. However, problems with the technique could arise once the advance rate becomes very small and the time-step becomes larger. Firstly, the time-step variable could become so large that it introduces convergence problems and volume-balance errors. Secondly, the advance may never reach the target node. In either case, new nodes may have to be added during the time-step. This requires both some judgement as to where to place the new node, and also some interpolation of the new node parameter values at the previous time-step. This process may have to be repeated many times during the course of a simulation. In practice, the fixed-time-step method is the simpler (mathematically, and programmatically) and more robust method of the two.

Having the global unknown time-step parameter in the generalised equations is still useful. If it is known that runoff is just about to occur, this feature can be used during the time-step to fix a node at the end of a furrow before performing

[3] These results have not been included in this dissertation.

the calculations. Otherwise, a trial and error method is required to try and match up the last node location with the position of the furrow end.

3.3.3 Solution grid formation

Two coordinate systems are commonly employed when using the finite difference forms of solution technique such as the double-sweep method. One is the Eulerian integration approach (Figure 3.2a) which uses a stationary rectangular grid structure; and the second is a "deformable control volume" method (Figure 3.2b) which uses a deforming cell (trapezoidal) grid structure where the cells have a forward velocity, and is often incorrectly called the "Lagrangian" system in the literature.

The Eulerian system has been chosen as the default coordinate system in the *FIDO* simulation engine due to its inclusion in the Walker model. However, the engine has been designed to accommodate either coordinate system, in keeping with its general open structure geared towards future development. The differences in computer code materialise in both the model equations and derivative terms, and also the parameter referencing. The "deformable control volume" form of the finite difference equations (see Eqns. 3.7 and 3.8) contain extra terms that disappear when the cells become rectangular. The model parameter-objects (see section 3.5.3) contain a switch that can swap cell coordinate systems (using "pointer" convention). At present, switching between systems is internal and changeable only by the developer.

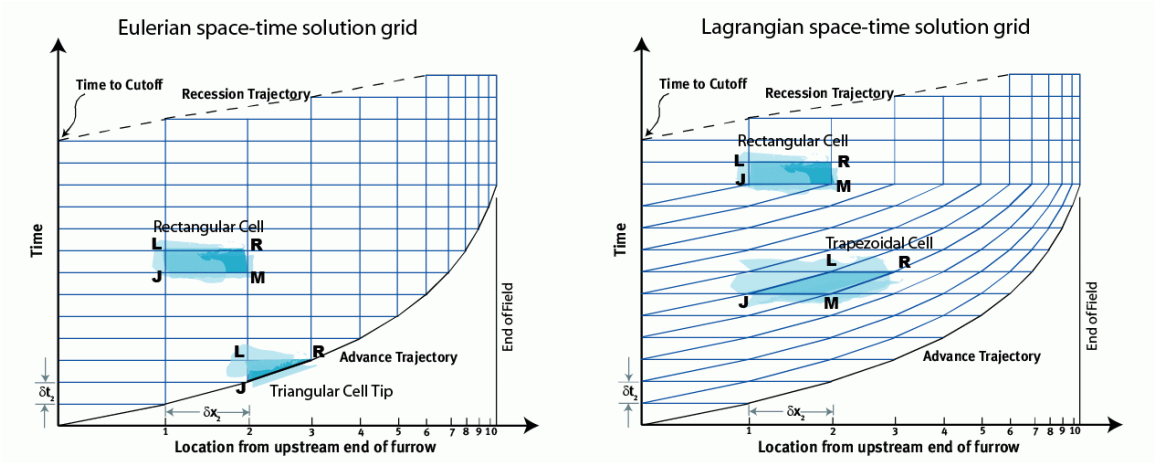


Figure 3.2: Eulerian (a) and "deformable control volume (Lagrangian)" (b) grid structures.

The benefit of one system over the other is questionable as there is no evidence in the literature of comparative performance. Many of the older models (such as Souza 1981; Rayej and Wallender 1985; Wallender and Rayej 1990) used the "deformable control volume" coordinate system while the advance was moving down the field and then switched to the Eulerian system once runoff occurred. However, the Eulerian form was adopted throughout for the *FIDO* simulation engine as fewer calculations are required.

3.3.4 Input requirements

There are thirteen input variables required by the simulation engine in order to perform a simulation (Table 3.1). These parameter values are passed into the engine through an input-object (model record), stored in a database and are editable through the user interface (see Chapter 7). SI units are used for each parameter in the calculations, even though a different set of units may have been entered into the interface by the user.

Table 3.1: Input variables required by the simulation engine.

Management Variables	Field Variables	Soil Parameters	Furrow Parameters
Flow rate, Q_{in}	field-length, L	Kostiakov a	σ_1
time-to-cutoff, t_c	field slope, So	Kostiakov k	σ_2
Z-Required, Z_{req}	Manning n	Kostiakov fo	ρ_1
			ρ_2

The field, soil and furrow variables are all vector types[4] to store a range of values to **account for spatially varying properties** of a furrow. The flowrate variable is also a vector to account for **variable inflow**. The validation of the simulation engine presented in this dissertation is based upon spatially and temporally uniform conditions so the default size of these vectors is "1".

Each variable is tested before being loaded into the simulation engine. Simulation will not take place if any of the parameter values are undefined or outside of predefined limits. The furrow variables σ_1, σ_2, ρ_1 and ρ_2 (empirical shape factors representing the furrow geometry) are also calculated at this time. These parameters are generated from the top-width, mid-width, bottom-width and maximum depth parameters located in the input record, although they can also be explicitly defined through the user interface.

3.3.5 Simulation engine outputs

The simulation engine is designed to be the central core of the decision support system supplying simulation output data for external analysis. Two types of outputs are generated: *solution node* information representing the physical properties of the flow and infiltration profiles at each time-step; and *summary* values describing irrigation performance. The second of these is calculated on a need-to-know basis for any time-step so as to minimise the demand on computer system resources.

3.3.6 Solution node outputs

During the simulation, the values of a number of variables are calculated and stored in the output object at each grid point in time and space. This includes the cross-sectional area of flow ($A_{t,x}$), flowrate ($Q_{t,x}$), cumulative infiltration depth ($Z_{t,x}$)

[4] The vector form of these variables is disabled in the presented version of the simulation engine. Future versions of the software will see these feature enabled. The validation presented is for uniform conditions.

and the node location ($X_{t,x}$). Note that $X_{t,x}$ needs to be stored at each time-step given the possibility of cells being removed or repositioned during the simulation. Other parameters that are stored at each time-step include total-time, delta-time (dt_t), the downstream cell-index, and upstream cell-index. Finally, maximum flow depth, maximum cumulative infiltration volume, number of time-steps, and total number of iterations are also recorded.

3.3.7 Summary outputs

The traditional measures of irrigation performance of "Application Efficiency" and "Storage Efficiency", along with "Application Uniformity" (as opposed to the more commonly used "Distribution Uniformity") and the percentage "Volume-balance Error" are calculated and stored in the output-object for any time-step. These performance measures are defined based upon volume-balance principles and may differ from what other authors define as standards. The main reason for choosing these indicators is that *firstly* they can be calculated at any time during the simulation[5]; and *secondly*, they are simple to calculate and can easily be explained (to farmers) through simple diagrams.

Application efficiency (AE): is the ratio of the amount of water that is stored in the root zone (below Z_{req}) to the total amount of water applied to the field (Figure 3.3). Most researchers fail to include the surface water volume in the calculation because they only need to calculate application efficiency once the simulation is completed. By adding this term, we are able to monitor application efficiency throughout the irrigation. In this case, the surface water storage is classed as a temporary "loss", reducing the magnitude of the application efficiency.

$$AE = \frac{InflowVol - DrainageVol - RunoffVol - SurfaceVol}{InflowVol} \times 100\% \quad \text{(3.1)}$$

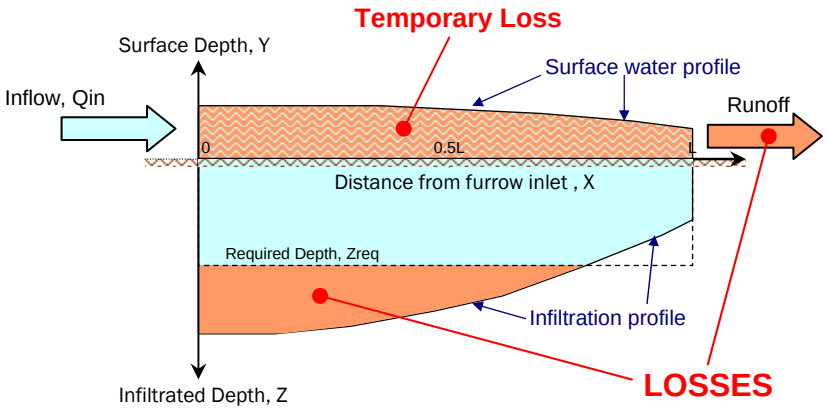


Figure 3.3: Components used in calculating Application Efficiency.

[5] The traditional measure of distribution uniformity can only be calculated once the advance has been completed.

Storage efficiency (SE): is the ratio of the volume of water stored in the root-zone (defined by Z_{req}) to the total storage capacity of the root-zone (Figure 3.4).

$$SE = \frac{StoredVol}{Z_{req} \times FieldLength \times 1000} \times 100\% \quad \text{.....(3.2)}$$

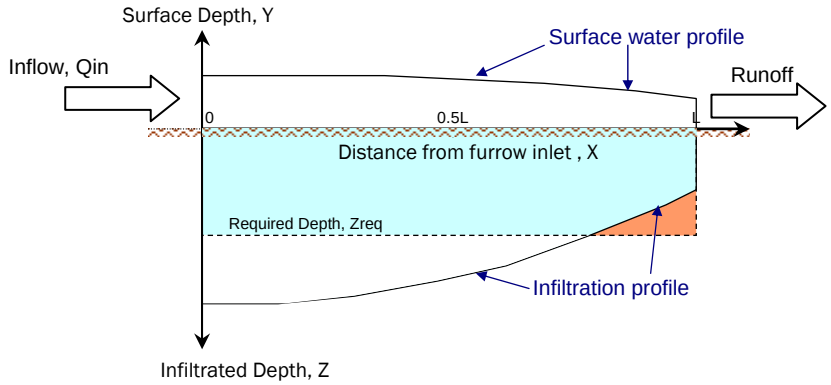


Figure 3.4: Components used in calculating Storage Efficiency.

Application uniformity (AU): is described as how evenly the water is applied along the furrow (Figure 3.5).

$$AU = \frac{StoredVol + DrainageVol}{Z_{t,0} \times FieldLength \times 1000} \times 100\% \quad \text{.....(3.3)}$$

where $Z_{t,0}$ is the infiltration depth at the top end of the furrow. This normally corresponds with the maximum infiltrated depth.

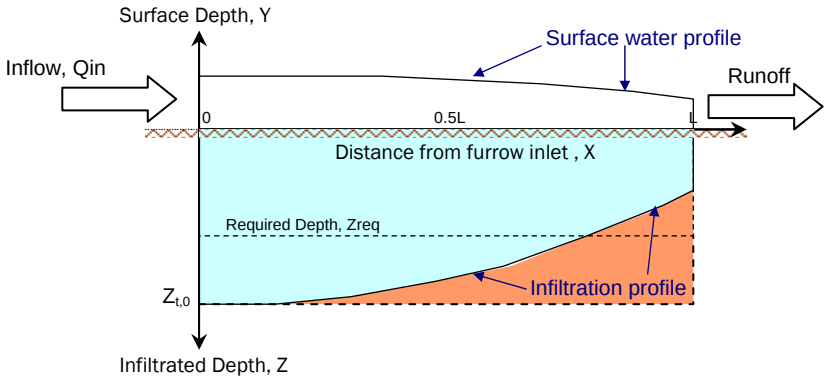


Figure 3.5: Components used in calculating Application Uniformity.

Volume-balance error (VBE): is not an irrigation performance measure but a measure of model accuracy.

$$VBE = \left(1 - \frac{(SurfaceVol + StoredVol + DrainageVol + RunoffVol)}{InflowVol}\right) \times 100\% \quad \text{.....(3.4)}$$

Other summary values that are calculated and stored in the output-object include: inflow volume; surface volume; stored volume; drainage volume; runoff volume; and error volume. Other performance indicators such as "Tail Water Ratio" and "Deep Percolation Ratio" are not generated but are easily calculated from the data provided in the output object.

3.4 Refinement of the numerical method

Translating and embedding the numerical solution technique into a computer algorithm is a complex task. Robustness, flexibility and reusability of the model are highly dependent on the effectiveness of this procedure. Reducing the solution equations to their simplest form before translation is a prerequisite for efficient, flexible, and robust programming. However, neither Walker's nor Strelkoff's published solution equations have been reduced into their simplest form. Therefore, these equations have been rederived from first principals in this chapter to achieve this objective.

The new equations differ from that published by Walker and Strelkoff through the calculation of "intermediate values", which significantly simplifies the algebra. The solution from these equations should be identical to that of those published, and will probably have little effect on the efficiency of the solution. However, the benefit is a much easier-to-understand implementation of the solution leading to both simpler translations into computer code, and a better basis for future modification and enhancement.

3.4.1 Principal formulation

The derivation of the solution technique that follows is a generalised form of the solution equations, which includes the global unknown time-step parameter as a solution variable. Modifications to the technique are required to tailor it to specific irrigation conditions such as different irrigation phases and furrow end conditions. Later sections in this chapter will cover these adjustments.

This generalised numerical technique (to solve to the continuity and momentum equations) represents the solution to the finite difference approximations to the following equations, first described in Chapter 2 (Eqns. 2.1 and 2.2). Rewritten here in terms of Q and A (instead of q and y), they are:

$$\frac{\partial A}{\partial t}+\frac{\partial Q}{\partial x}+\frac{\partial Z}{\partial t}=0 \quad (3.5)$$

and,

$$\frac{1}{g}\frac{\partial Q}{\partial t}+\frac{\partial}{\partial x}\left(\frac{Q^2}{gA}\right)+\frac{\partial P}{\partial x}+\left(D-AS_0\right)=0 \quad (3.6)$$

where, A is the cross-sectional area of flow (m^2), Z is the infiltrated volume/unit length (m^3/m), Q is the flowrate (m^3/sec), P is the hydrostatic pressure (N/m^2), D is the drag force (friction force), and S_0 is the field slope.

By multiplying through by ∂t and ∂x and replacing the partial differential terms with their finite difference approximations, Eqn. 2.1 then becomes:

$$\begin{aligned}
&[\theta(Q_L - Q_R) + (1-\theta)(Q_J - Q_M)]dt \\
&-[\theta(A_L + Z_L) + (1-\theta)(A_J + Z_J)](X_L - X_J) \\
&+[\theta(A_R + Z_R) + (1-\theta)(A_M + Z_M)](X_R - X_M) \\
&+[\phi(A_J + Z_J) + (1-\phi)(A_M + Z_M)](X_M - X_J) \\
&-[\phi(A_L + Z_L) + (1-\phi)(A_R + Z_R)](X_R - X_L) = R_C
\end{aligned} \quad \text{.....(3.7)}$$

Lagrangian Components:
These terms reduce to "0" when cells are rectangular (stationary).

and, Eqn. 2.2 becomes:

$$\begin{aligned}
&\frac{1}{g}[\phi Q_J + (1-\phi)Q_M](X_M - X_J) - \frac{1}{g}[\phi Q_L + (1-\phi)Q_R](X_R - X_L) \\
&+\frac{1}{g}[\theta Q_R + (1-\theta)Q_M](X_R - X_M) - \frac{1}{g}[\theta Q_L + (1-\theta)Q_J](X_L - X_J) \\
&+\frac{1}{g}\left[\theta\left(\frac{Q_L^2}{A_L} - \frac{Q_R^2}{A_R}\right) + (1-\theta)\left(\frac{Q_J^2}{A_J} - \frac{Q_M^2}{A_M}\right)\right]dt \\
&+[\theta P_L + (1-\theta)P_J]dt - [\theta P_R + (1-\theta)P_M]dt \\
&+\left(\theta[\phi(D_L - S_0 A_L) + (1-\phi)(D_R - S_0 A_R)]\right)(X_R - X_L)dt \\
&+\left((1-\theta)[\phi(D_J - S_0 A_J) + (1-\phi)(D_M - S_0 A_M)]\right)(X_M - X_J)dt = R_M
\end{aligned} \quad \text{.....(3.8)}$$

where ϕ is a space-averaging coefficient and θ is a time-averaging coefficient. These values are typically equal to 0.6. The terms R_C and R_M are the residuals of continuity and momentum respectively. These residual values will be approximately equal to zero, and any deviation from zero will be attributed to a volume-balance error resulting from the finite differencing approximation used. The subscript referencing J, M, L and R represent the variable's Eulerian grid cell representation (Figure 3.6) on the space-time solution grid. Each individual cell on the solution grid is represented by a pair of these equations.

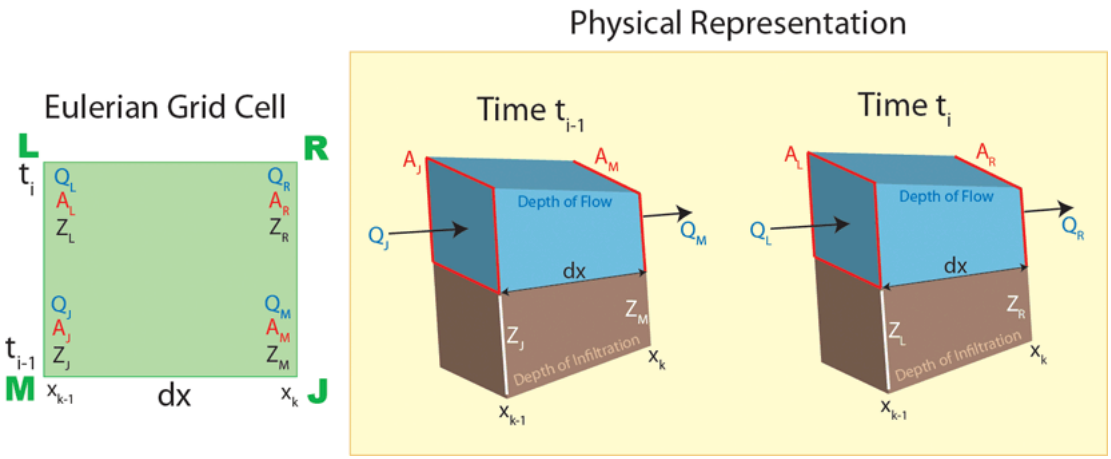


Figure 3.6: Eulerian Grid Cells and time-dependant (physical) representation.

The hydrostatic pressure P, can be related in terms of the cross-sectional area of flow A:

$$P = \frac{\sigma_1^{\frac{-1}{\sigma_2}}}{(1+\sigma_2)A^{1+\frac{1}{\sigma_2}}} \quad (3.9)$$

Where σ_1 and σ_2 are furrow coefficients defined by:

$$A^2 R^{1.33} = \rho_1 A^{\rho_2} \quad (3.10)$$

The wetted perimeter can also be defined in terms of ρ_1 and ρ_2:

$$WP = \left(\frac{1}{\rho_1}\right)^{\frac{3}{4}} A^{\left(\frac{5}{2}-\frac{3}{4}\rho_2\right)} \quad (3.11)$$

The drag force (friction force) D is related by the following equation:

$$D = \frac{n^2}{\rho_1} Q^2 A^{1-\rho_2} \quad (3.12)$$

We can simplify Eqns. 3.7 and 3.8 by introducing the term dx, which represents the distance between cell node positions. Eqn. 3.7 then becomes:

$$\begin{aligned}
&\left[\theta(Q_L - Q_R) + (1-\theta)(Q_J - Q_M)\right]dt \\
&-\left[\theta(A_L + Z_L) + (1-\theta)(A_J + Z_J)\right]dx_{LJ} \\
&+\left[\theta(A_R + Z_R) + (1-\theta)(A_M + Z_M)\right]dx_{RM} \\
&+\left[\phi(A_J + Z_J) + (1-\phi)(A_M + Z_M)\right]dx_{JM} \\
&-\left[\phi(A_L + Z_L) + (1-\phi)(A_R + Z_R)\right]dx_{RL} = R_C
\end{aligned} \quad (3.13)$$

Eqn. 3.8 then becomes:

$$\begin{aligned}
&\left(\left[\phi Q_J + (1-\phi)Q_M\right]dx_{MJ} - \left[\phi Q_L + (1-\phi)Q_R\right]dx_{RL}\right. \\
&\left.+\left[\theta Q_R + (1-\theta)Q_M\right]dx_{RM} - \left[\theta Q_L + (1-\theta)Q_J\right]dx_{LJ}\right)\frac{1}{g}\frac{\partial R_M}{\partial dt}dt = R_M
\end{aligned} \quad (3.14)$$

where for convenience, $\frac{\partial R_M}{\partial dt}$ is defined by:

$$\begin{aligned}
&\frac{\partial R_M}{\partial dt} = \theta\left(\frac{Q_L^2}{gA_L + P_L} - \frac{Q_R^2}{gA_R + P_R}\right) + (1-\theta)\left(\frac{Q_J^2}{gA_J + P_J} - \frac{Q_M^2}{gA_M + P_M}\right) \\
&-\theta\left[\phi(D_L - S_0 A_L) + (1-\phi)(D_R - S_0 A_R)\right]dx_{RL} \\
&-(1-\theta)\left[\phi(D_J - S_0 A_J) + (1-\phi)(D_M - S_0 A_M)\right]dx_{MJ}
\end{aligned} \quad (3.15)$$

This simplification is useful when transforming the mathematics into its computer-code form, as $\partial R_M / \partial dt$ already needs to be calculated as part of the double-sweep technique. Using the Eulerian coordinate system, these equations can be further simplified down to:

$$\left[\theta(Q_L - Q_R) + (1-\theta)(Q_J - Q_M)\right]\delta t + \left[\phi(A_J + Z_J) + (1-\phi)(A_M + Z_M)\right]dx_{JM} - \left[\phi(A_L + Z_L) + (1-\phi)(A_R + Z_R)\right]dx_{RL} = R_C \quad (3.16)$$

and,

$$\left(\left[\phi Q_J + (1-\phi)Q_M\right]dx_{MJ} - \left[\phi Q_L + (1-\phi)Q_R\right]dx_{RL} \right)\frac{1}{g}\frac{\partial R_M}{\partial dt}dt = R_M \quad (3.17)$$

Now that the discretisation of the underlying model has been defined, we can begin to derive its solution. The double-sweep method that we are developing uses a Newton-Raphson procedure for solving these equations for unknowns A, Q (and possibly dx and/or dt,) for each cell on the solution grid. This methodology effectively linearises the equations, whose solution is then determined iteratively.

To implement this methodology, consider a single cell element (Figure 3.6) of the solution grid. The residuals R_C and R_M for this cell are first written in terms of a Taylor Series expansion, which linearises the expressions:

$$R_{C1}^{\,n+1} = R_{C1}^{\,n} + \left(\nabla R_{C1}^{\,n}\right)\Delta R_{C1}^{\,n} \quad (3.18)$$

$$R_{M1}^{\,n+1} = R_{M1}^{\,n} + \left(\nabla R_{M1}^{\,n}\right)\Delta R_{M1}^{\,n} \quad (3.19)$$

where n is the iteration number. The derivation begins to differ from Walker's solution at this point through the inclusion of the global value of the time-step parameter (dt) pertaining to all n, following the methodology of Strelkoff. Given starting conditions (starting values of the unknowns), the solution will continue to be improved over successive iterations. Therefore the residuals at the improved solution are functions of current solution:

$$R_{C1}^{\,n} = R_C\left(A_L^n, Q_L^n, A_R^n, Q_R^n, dt^n\right) \quad (3.20)$$

$$R_{M1}^{\,n} = R_M\left(A_L^n, Q_L^n, A_R^n, Q_R^n, dt^n\right) \quad (3.21)$$

The gradient terms, $\nabla R_{C1}^{\,n}$ and $\nabla R_{M1}^{\,n}$ are the Jacobian matrices:

$$\nabla Rc_1^n = \left(\frac{\partial R_{C1}}{\partial A_L}, \frac{\partial R_{C1}}{\partial Q_L}, \frac{\partial R_{C1}}{\partial A_R}, \frac{\partial R_{C1}}{\partial Q_R}, \frac{\partial R_{C1}}{\partial dt}\right)^n \quad (3.22)$$

$$\nabla R_{M1}^{\,n} = \left(\frac{\partial R_{M1}}{\partial A_L}, \frac{\partial R_{M1}}{\partial Q_L}, \frac{\partial R_{M1}}{\partial A_R}, \frac{\partial R_{M1}}{\partial Q_R}, \frac{\partial R_{M1}}{\partial dt}\right)^n \quad (3.23)$$

The difference terms $\Delta R_{C1}^{\,n}$ and $\Delta R_{M1}^{\,n}$ are:

$$\Delta R_{Cn}^{\,1} = \left(A_L^{n+1} - A_L^n, Q_L^{n+1} - Q_L^n, A_R^{n+1} - A_R^n, Q_R^{n+1} - Q_R^n, dt^{n+1} - dt^n\right) = \left(\delta A_L, \delta Q_L, \delta A_R, \delta Q_R, \delta dt\right) \quad (3.24)$$

$$\Delta R_{Mn}^{\,1} = \left(A_L^{n+1} - A_L^n, Q_L^{n+1} - Q_L^n, A_R^{n+1} - A_R^n, Q_R^{n+1} - Q_R^n, dt^{n+1} - dt^n\right) = \left(\delta A_L, \delta Q_L, \delta A_R, \delta Q_R, \delta dt\right) \quad (3.25)$$

Now rewriting Eqns. 3.18 and 3.19 in expanded form, we get:

$$R_{C1}^{\,n+1} = R_{C1}^{\,n} + \left(\frac{\partial R_{C1}}{\partial A_L}\right)^n \delta A_L + \left(\frac{\partial R_{C1}}{\partial Q_L}\right)^n \delta Q_L + \left(\frac{\partial R_{C1}}{\partial A_R}\right)^n \delta A_R + \left(\frac{\partial R_{C1}}{\partial Q_R}\right)^n \delta Q_R + \left(\frac{\partial R_{C1}}{\partial dt}\right)^n \delta dt \quad (3.26)$$

$$R_{M1}^{\,n+1} = R_{M1}^{\,n} + \left(\frac{\partial R_{M1}}{\partial A_L}\right)^n \delta A_L + \left(\frac{\partial R_{M1}}{\partial Q_L}\right)^n \delta Q_L + \left(\frac{\partial R_{M1}}{\partial A_R}\right)^n \delta A_R + \left(\frac{\partial R_{M1}}{\partial Q_R}\right)^n \delta Q_R + \left(\frac{\partial R_{M1}}{\partial dt}\right)^n \delta dt \quad (3.27)$$

Because the residuals $R_{C1}^{\,n+1}$ and $R_{M1}^{\,n+1}$ will eventually tend towards zero, they are set to zero at each iteration leading to two linearised equations with four unknowns $\delta A_L, \delta Q_L, \delta A_R, \delta Q_R$. Therefore, by setting $R_{C1}^{\,n+1} = 0$ and $R_{M1}^{\,n+1} = 0$, Eqns. 3.26 and 3. become:

$$-R_{C1}^{\,n} = \left(\frac{\partial R_{C1}}{\partial A_L}\right)^n \delta A_L + \left(\frac{\partial R_{C1}}{\partial Q_L}\right)^n \delta Q_L + \left(\frac{\partial R_{C1}}{\partial A_R}\right)^n \delta A_R + \left(\frac{\partial R_{C1}}{\partial Q_R}\right)^n \delta Q_R + \left(\frac{\partial R_{C1}}{\partial dt}\right)^n \delta dt \quad (3.28)$$

$$-R_{M1}^{\,n} = \left(\frac{\partial R_{M1}}{\partial A_L}\right)^n \delta A_L + \left(\frac{\partial R_{M1}}{\partial Q_L}\right)^n \delta Q_L + \left(\frac{\partial R_{M1}}{\partial A_R}\right)^n \delta A_R + \left(\frac{\partial R_{M1}}{\partial Q_R}\right)^n \delta Q_R + \left(\frac{\partial R_{M1}}{\partial dt}\right)^n \delta dt \quad (3.29)$$

To simplify the notation, we can substitute algebraic variables for the partial derivate terms as follows:

$$a_i \delta A_{i-1} + b_i \delta Q_{i-1} + c_i \delta A_i + d_i \delta Q_i + e_i \delta dt = -R_{C\,i} \quad (3.30)$$

$$p_i \delta A_{i-1} + q_i \delta Q_{i-1} + r_i \delta A_i + s_i \delta Q_i + u_i \delta dt = -R_{M\,i} \quad (3.31)$$

where i denote the cell index. This is slightly different to Strelkoff's formulation (but similar to Walker's) in that the indexing of the solution variables has purposely been shifted to the left (i becomes i-1) to simplify the explanation of the system. For example, the left hand side of the first cell is now indexed as 0, while Strelkoff has it indexed as 1. This has no effect on the results at all, but impacts directly on the readability of the mathematics and computer code. This subtle transformation has helped to derive a more simplified version of the solution technique.

These linearised equations form the basis of the double-sweep technique. Therefore, N pairs of these equations (where N represents the number of cells for the current time-step) can be put into matrix form and solved iteratively until the change in the solution variables (between iterations) becomes negligible.

Unfortunately, the matrix algebra required for solving such a (potentially) large set of equations is complex. However, the matrix is banded which leads to a particularly efficient solution using the double-sweep methodology. This solution technique is formulated by assuming that a linear combination of the flowrate, flow-area and time-step increment variables exist for each node on the solution grid:

$$\delta Q_i = E_i \delta A_i + H_i \delta dt + F_i \quad (3.32)$$

For the case of only two cells in the solution grid, the previous three equations can be put into matrix form:

$$\begin{bmatrix} -E_0 & 1 & & & & & -H_0 \\ a_1 & b_1 & c_1 & d_1 & & & e_1 \\ p_1 & q_1 & r_1 & s_1 & & & u_1 \\ & & a_2 & b_2 & c_2 & d_2 & e_2 \\ & & p_2 & q_2 & r_2 & s_2 & u_2 \\ & & & & -E_2 & 1 & -H_2 \end{bmatrix} \begin{bmatrix} \delta A_0 \\ \delta Q_0 \\ \delta A_1 \\ \delta Q_1 \\ \delta A_2 \\ \delta Q_3 \\ \delta dt \end{bmatrix} = \begin{bmatrix} F_0 \\ -R_{C1} \\ -R_{M1} \\ -R_{C2} \\ -R_{M2} \\ F_2 \end{bmatrix} \quad \text{(3.33)}$$

Defining the matrix in this way allows the solution sweep to start from the upstream cell and iterate through to the downstream cell. We begin the sweep by determining auxiliary coefficients for each cell. These coefficients are derived by combining Eqns. 3.30 to 3.32. From this point onwards, the derivation follows closely to Strelkoff's methodology, with the structural form of the equations appearing very different to that derived by Walker. Beginning the derivation, Eqn. 3.32 written for the first cell is:

$$\delta Q_0 = E_0 \delta A_0 + H_0 \delta dt + F_0 \quad \text{(3.34)}$$

Then Eqn. 3.30 written for the last cell is:

$$a_1 \delta A_0 + b_1 \delta Q_0 + c_1 \delta A_1 + d_1 \delta Q_1 + e_1 \delta dt = -R_{C1} \quad \text{(3.35)}$$

Substituting Eqn. 3.34 into Eqn. 3.35, we get:

$$a_1 \delta A_0 + (b_1 E_0 \delta A_0 + b_1 H_0 \delta dt + b_1 F_0) + c_1 \delta A_1 + d_1 \delta Q_1 + e_1 \delta dt = -R_{C1} \quad \text{(3.36)}$$

Simplifying, this result, we now have:

$$\delta A_0 (a_1 + b_1 E_0) + \delta dt (e_1 + b_1 H_0) + c_1 \delta A_1 + d_1 \delta Q_1 = -R_{C1} - b_1 F_0 \quad \text{(3.37)}$$

Then ∂A_0 equals:

$$\delta A_0 = \frac{-R_{C1} - b_1 F_0 - \delta dt (e_1 + b_1 H_o) - c_1 \delta A_1 - d_1 \delta Q_1}{a_1 + b_1 E_0} \quad \text{(3.38)}$$

We can simplify this equation by introducing the new terms U,V, Z, and W:

$$\delta A_0 = U_0 \delta A_1 + V_0 \delta Q_1 + Z_0 \delta dt + W_0 \quad \text{(3.39)}$$

Where

$$U_0 = \frac{-c_1}{a_1 + b_1 E_0} \quad \text{(3.40)}$$

$$V_0 = \frac{-d_1 \delta Q_1}{a_1 + b_1 E_0} \quad \text{(3.41)}$$

$$Z_0 = \frac{-(e_1 + b_1 H_o)}{a_1 + b_1 E_0} \quad \text{(3.42)}$$

$$W_0 = \frac{-R_{C1} - b_1 F_0}{a_1 + b_1 E_0} \quad \text{(3.43)}$$

We now need to repeat this process for Eqn 3.31. Rewriting Eqn 3.31 for the first cell, we have:

$$p_1\delta A_0 + q_1\delta Q_0 + r_1\delta A_1 + s_1\delta Q_1 + u_1\delta dt = -R_{M1} \quad (3.44)$$

Then substituting Eqn 3.34 into Eqn 3.44:

$$p_1\delta A_0 + (q_1E_0\delta A_0 + q_1H_0\delta dt + q_1F_0) + r_1\delta A_1 + s_1\delta Q_1 + u_1\delta dt = -R_{M1} \quad (3.45)$$

Simplifying this we get:

$$\delta A_0(p_1 + q_1E_0) + \delta dt(u_1 + q_1H_0) + r_1\delta A_1 + s_1\delta Q_1 = -R_{M1} - q_1F_0 \quad (3.46)$$

Then substituting Eqn 3.38 into Eqn 3.46:

$$(p_1 + q_1E_0)\left(\frac{-R_{C1} - b_1F_0 - \delta dt(e_1 + b_1H_0) - c_1\delta A_1 - d_1\delta Q_1}{a_1 + b_1E_0}\right) + \delta dt(u_1 + q_1H_0) + r_1\delta A_1 + s_1\delta Q_1 = -R_{M1} - q_1F_0 \quad (3.47)$$

Expanding these terms we get:

$$\begin{aligned} &-(p_1 + q_1E_0)(R_{C1} + b_1F_0) - (p_1 + q_1E_0)(e_1 + b_1H_0)\delta dt - (p_1 + q_1E_0)c_1\delta A_1 \\ &-(p_1 + q_1E_0)d_1\delta Q + (u_1 + q_1H_0)(a_1 + b_1E_0)\delta dt + (a_1 + b_1E_0)r_1\delta A_1 + (a_1 + b_1E_0)s_1\delta Q_1 \\ &= -(R_{M1} + q_1F_0)(a_1 + b_1E_0) \end{aligned} \quad (3.48)$$

Finally, grouping the solution variables together, the equation becomes:

$$\begin{aligned} \delta Q_1 = &\frac{-(p_1 + q_1E_0)c_1 + (a_1 + b_1E_0)r_1}{(p_1 + q_1E_0)d_1 - (a_1 + b_1E_0)s_1}\delta A_1 \\ &+ \frac{-(p_1 + q_1E_0)(e_1 + b_1H_0) + (u_1 + q_1H_0)(a_1 + b_1E_0)}{(p_1 + q_1E_0)d_1 - (a_1 + b_1E_0)s_1}\delta dt \\ &+ \frac{-(p_1 + q_1E_0)(R_{C1} + b_1F_0) + (a_1 + b_1E_0)(R_{M1} + q_1F_0)}{(p_1 + q_1E_0)d_1 - (a_1 + b_1E_0)s_1} \end{aligned} \quad (3.49)$$

Notice that this equation now takes the form of Eqn. 3.32:

$$\delta Q_1 = E_1\delta A_1 + H_1\delta dt + F_1 \quad (3.50)$$

Therefore, we now have equations to represent the coefficients E, H and F:

$$E_1 = \frac{-(p_1 + q_1E_0)c_1 + (a_1 + b_1E_0)r_1}{(p_1 + q_1E_0)d_1 - (a_1 + b_1E_0)s_1} \quad (3.51)$$

$$H_1 = \frac{-(p_1 + q_1E_0)(e_1 + b_1H_0) + (u_1 + q_1H_0)(a_1 + b_1E_0)}{(p_1 + q_1E_0)d_1 - (a_1 + b_1E_0)s_1} \quad (3.52)$$

$$F_1 = \frac{-(p_1 + q_1E_0)(R_{C1} + b_1F_0) + (a_1 + b_1E_0)(R_{M1} + q_1F_0)}{(p_1 + q_1E_0)d_1 - (a_1 + b_1E_0)s_1} \quad (3.53)$$

These equations are very similar to those derived by Strelkoff (albeit with different indexing) although they are not yet in their simplest form. There is a recurring pattern within these equations, so if we now introduce new temporary parameters $T_{(1)}$ to $T_{(7)}$, we are able to reduce these equations even further:

$$T_{(1)} = a_1 + b_1 E_0 \quad (3.54)$$

$$T_{(2)} = p_1 + q_1 E_0 \quad (3.55)$$

$$T_{(3)} = e_1 + b_1 H_0 \quad (3.56)$$

$$T_{(4)} = u_1 + q_1 H_0 \quad (3.57)$$

$$T_{(5)} = R_{(C)_1} + b_1 F_0 \quad (3.58)$$

$$T_{(6)} = R_{(M)_1} + q_1 F_0 \quad (3.59)$$

$$T_{(7)} = d_1 T_{(2)} - s_1 T_{(1)} \quad (3.60)$$

Therefore, the coefficients E, H and F now become:

$$E_1 = \frac{\left(-c_1 T_{(2)} + r_1 T_{(1)}\right)}{T_{(7)}} \quad (3.61)$$

$$H_1 = \frac{\left(-T_{(3)} T_{(2)} + T_{(4)} T_{(1)}\right)}{T_{(7)}} \quad (3.62)$$

$$F_1 = \frac{\left(-T_{(5)} T_{(2)} + T_{(6)} T_{(1)}\right)}{T_{(7)}} \quad (3.63)$$

We can also rewrite Eqns. 3.40 to 3.43 using this formulation:

$$U_0 = -c_1 / T_{(1)} \quad (3.64)$$

$$V_0 = -d_1 / T_{(1)} \quad (3.65)$$

$$Z_0 = -T_{(3)} / T_{(1)} \quad (3.66)$$

$$W_0 = -T_{(5)} / T_{(1)} \quad (3.67)$$

Therefore, going back to a more generalised form with subscript i representing the cell-node index, the double-sweep solution equations can be summarised as:

$$\delta Q_i = E_i \delta A_i + H_i \delta dt + F_i \quad (3.68)$$

$$\delta A_{i-1} = U_{i-1} \delta A_i + V_{i-1} \delta Q_i + Z_{i-1} \delta dt + W_{i-1} \quad (3.69)$$

where

$$E_i = (r_i T_{(1)} - c_i T_{(2)}) / T_{(7)} \quad (3.70)$$

$$H_i = (T_{(4)} T_{(1)} - T_{(3)} T_{(2)}) / T_{(7)} \quad (3.71)$$

$$F_i = (T_{(6)} T_{(1)} - T_{(5)} T_{(2)}) / T_{(7)} \quad (3.72)$$

and

$$U_{i-1} = -c_i / T_{(1)} \quad (3.73)$$

$$V_{i-1} = -d_i / T_{(1)} \quad (3.74)$$

$$Z_{i-1} = -T_{(3)} / T_{(1)} \quad (3.75)$$

$$W_{i-1} = -T_{(5)} / T_{(1)} \quad (3.76)$$

Then, the temporary variables are expressed as:

$$T_{(1)} = a_i + b_i E_{i-1} \quad (3.77)$$

$$T_{(2)} = p_i + q_i E_{i-1} \quad (3.78)$$

$$T_{(3)} = e_i + b_i H_{i-1} \quad (3.79)$$

$$T_{(4)} = u_i + q_i H_{i-1} \quad (3.80)$$

$$T_{(5)} = R_{C\,i} + b_i F_{i-1} \quad (3.81)$$

$$T_{(6)} = R_{M\,i} + q_i F_{i-1} \quad (3.82)$$

$$T_{(7)} = d_i T_{(2)} - s_i T_{(1)} \quad (3.83)$$

To solve for the incremental changes to the solution parameters (δQ_i and δA_i), the temporary variables $T_{(1)}$ to $T_{(7)}$ and auxiliary coefficients E, H, F, U, V, Z and W are calculated for each cell (for the current time-step) using Eqns. 3.70 to 3.83, progressing in a forward sweep starting from the upstream cell and marching to the downstream cell. Once these parameters have been calculated, Eqns. 3.68 and 3.69 are used in a backward sweep to calculate the incremental changes to solution variables δQ_i and δA_i.

Once the incremental changes to solution parameters for all cells have been calculated, the actually solution parameter values can be updated using:

$$A_i^{n+1} = A_i^n + \delta A_i \quad (3.84)$$

$$Q_i^{n+1} = Q_i^n + \delta Q_i \quad (3.85)$$

$$dt_i^{n+1} = dt_i^n + \delta dt_i \quad (3.86)$$

The computer algorithm (developed in section 3.5 of this chapter) will be responsible for determining which of these parameters will form part of the

solution. During the advance phase, the simulation is solved for flowrate, flow-area and advance distance. In this case, the coefficients of the time-step parameter are disabled (equal 0) in the double sweep algorithm. More specific solution treatments are presented in the remainder of this chapter.

3.4.2 First cell calculations

During the first time-step, the solution grid is composed of a single triangular cell with the J, M and R subscripted variables equal to zero (Figure 3.7). The two unknowns in the system of equations are A_L and dx.

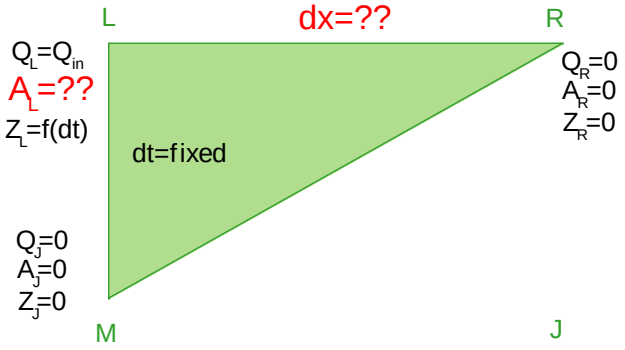


Figure 3.7: First cell representation.

For this cell, the continuity and momentum equations (Eqns. 3.7 and 3.8) therefore reduce to:

$$\theta Q_L \delta t - (A_L + Z_L)(X_R - X_L) = R_C \quad (3.87)$$

$$-\frac{1}{g}\phi Q_L \delta x + \frac{1}{g}\theta \frac{Q_L^2}{A_L}\delta t + \theta P_L \delta t + \theta\phi(D_L - S_0 A_L)\delta x = R_M \quad (3.88)$$

By substituting δdx for δQ_R in the residual of continuity and residual of momentum equations (Eqns. 3.30 and 3.31), we are able to solve for the incremental advance distance for this time-step. With this substitution, and removing the redundant terms for this time-step, Eqns. 3.30 and 3.31 reduce down to:

$$a_1 \delta A_L + d_1 \delta dx_1 = -R_C \quad (3.89)$$

$$p_1 \delta A_L + s_1 \delta dx_1 = -R_M \quad (3.90)$$

Therefore, we have two unknowns (δA_L and δdx) in two equations. We can isolate δA_L and δdx by substituting one equation into the other. To isolate δA_L, we substitute Eqn. 3.90 into Eqn. 3.89 for δdx_1 to get:

$$a_1 \delta A_L + d_i\left(\frac{-R_M - p_1 \delta A_L}{s_1}\right) = -R_C \quad (3.91)$$

Then simplifying, we have:

$$a_1 s_1 \delta A_L - R_M d_1 - p_1 d_1 \delta A_L = -R_C s_1 \quad (3.92)$$

Grouping like terms together:

$$\delta A_L \left(a_1 s_1 - p_1 d_1 \right) = R_M d_i - R_C s_1 \quad (3.93)$$

Finally, we can isolate the incremental change in the solution parameter:

$$\delta A_L = \frac{R_C s_1 - R_M d_1}{d_1 p_1 - s_1 a_1} \quad (3.94)$$

The same procedure can be following to isolate δdx_1, whereby this time, we substitute Eqn. 3.94 into Eqn. 3.90 for δA_L. After simplifying the equations and isolating δx_1, we get:

$$\delta dx_1 = \frac{R_M a_1 - R_C p_1}{d_1 p_1 - s_1 a_1} \quad (3.95)$$

These equations are used iteratively, as δA_L and δdx are only the incremental changes in the solution variables. The updated parameter value of δA_L can be calculated using Eqn. 3.84, and the updated value of dx can be calculated using:

$$dx_i^{n+1} = dx_i^n + \delta dx_i \quad (3.96)$$

Therefore, given starting values of A_L and dx (see Table 3.3) , then δA_L and δdx are calculated at each iteration and the new solution variables are recalculated until convergence is achieved (no further changes in the solution variables).

3.4.3 Advance phase calculations

As in the first-cell calculations, solving for the advance phase in the simulation involves substituting δdx for δQ_R in the residual of continuity and residual of momentum equations (Eqns. 3.30 and 3.31) for the downstream triangular cell. This assumes that a linear combination of the distance-step and flow-area exists at the downstream cell (similar to Eqn. 3.32) on the solution grid:

$$\delta dx_i = E_i \delta A_i + F_i \quad (3.97)$$

To solve for the advance, a fixed time-step size is used and the system of equations is solved for the incremental advance distance dx for each time-step. For the case of only two cells on the solution grid (Figure 3.8), the system of equations can be represented by:

$$\begin{bmatrix} -E_0 & 1 & & & & & \\ a_1 & b_1 & c_1 & d_1 & & & \\ p_1 & q_1 & r_1 & s_1 & & & \\ & & a_2 & b_2 & c_2 & d_2 & \\ & & p_2 & q_2 & r_2 & s_2 & \\ & & & & -E_2 & 1 & \end{bmatrix} \begin{bmatrix} \delta A_0 \\ \delta Q_0 \\ \delta A_1 \\ \delta Q_1 \\ \delta A_2 \\ \delta dx \end{bmatrix} = \begin{bmatrix} F_0 \\ -R_{C1} \\ -R_{M1} \\ -R_{C2} \\ -R_{M2} \\ F_2 \end{bmatrix} \quad (3.98)$$

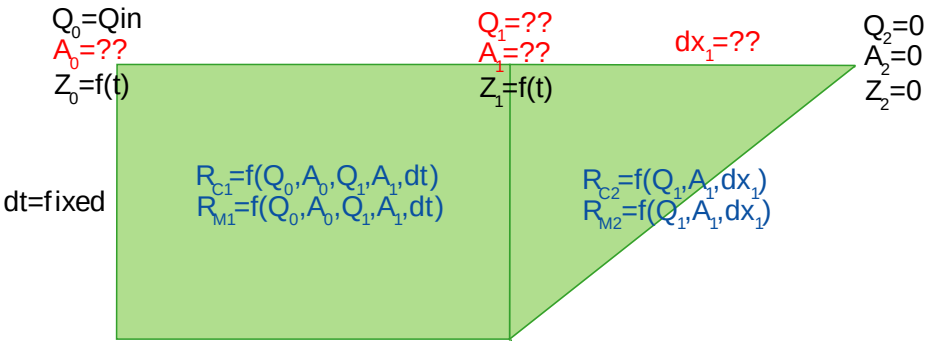


Figure 3.8: Two cell grid representation.

The assumptions of zero-flowrate and zero-flow-area exist at the front end of the triangular downstream cell. The unknowns in the two-cell example are A_0, Q_1, A_1 and dx. The solution begins by calculating the temporary variables $T_{(1)}$ to $T_{(7)}$ and auxiliary coefficients E, F, U, V and W for each cell using Eqns. 3.70 to 3.83, progressing in a forward sweep starting from the upstream cell and marching to the downstream cell. Time-step auxiliary coefficients H (Eqn. 3.71) and Z (Eqn. 3.75) no longer need calculating since dt is not a solution variable. Once these coefficients have been calculated, Eqns. 3.68, 3.69 and 3.97 are used in a backward sweep to calculate the incremental changes in the solution variables δQ_i, δA_i and for the last cell, δdx. Finally, the solution variables are updated using Eqns. 3.85, 3.84 and 3.96.

This algorithm can be used throughout the advance phase. However, at the end of the advance phase, it is very unlikely that a node will coincide with the exact location of the field end. Therefore, special treatment is required to match up the location of the last node with the end of the field. In this situation, the global unknown time-step parameter dt is included as a solution variable and the distance-step parameter dx is held constant.

To implement this, the simulation is monitored until the advance exceeds the field-length. Once this occurs, the last time-step is reset (including its solution), and a new node is positioned at the furrow outlet. The last cell is then tested to ensure that it is large enough to avoid convergence problems. From experimentation, it was found that the last cell should be at least a quarter of the width of the previous cell, otherwise the last two cells should be joined together. The generalised double sweep algorithm (Eqns. 3.70 to 3.83 and Eqns. 3.85 to 3.86) is then used to solve for dt instead of dx.

3.4.4 Runoff conditions

Special consideration must be taken once the advance reaches the end of the field and runoff occurs. Normal flow is assumed at the furrow outlet and is calculated using Manning's equation as a boundary condition. The double sweep technique is therefore modified to solve for these changed conditions. This modification occurs only in the last cell, whereby an explicit formulation of δA_R is presented that differs to what other researchers (e.g. Elliot *et al*. 1982) have

presented. Therefore, beginning the derivation, the boundary condition for the last cell is represented by the Manning Equation:

$$Q_{runoff} = \frac{(\rho_1 S_0)^{\frac{1}{2}}}{n} A_R^{\left(\frac{\rho_2}{2}\right)} \quad (3.99)$$

This can be simplified by introducing the coefficients α and ω:

$$Q_{runoff} = \alpha A_R^{\omega} \quad (3.100)$$

where

$$\omega = \frac{\rho_2}{2} \quad (3.101)$$

and

$$\alpha = \frac{(\rho_1 S_0)^{\frac{1}{2}}}{n} \quad (3.102)$$

By differentiating Eqn. 3.99 with respect to A_R, we have:

$$\frac{dQ_{runoff}}{dA_R} = \omega\alpha A_R^{(\omega-1)} = \frac{\delta Q_{runoff}}{\delta A_R} \quad (3.103)$$

Rearranging this we have:

$$\delta Q_{runoff} = \omega\alpha A_R^{(\omega-1)} \delta A_R \quad (3.104)$$

Now we can substitute this equation into Eqns. 3.28 and 3.29 to remove δQ_R from the equations:

$$-R_C = \frac{\partial R_C}{\partial A_L}\delta A_L + \frac{\partial R_C}{\partial Q_L}\delta Q_L + \left(\frac{\partial R_{C1}}{\partial A_R} + \omega\alpha A_R^{(\omega-1)}\frac{\partial R_C}{\partial Q_R}\right)\delta A_R \quad (3.105)$$

$$-R_M = \frac{\partial R_M}{\partial A_L}\delta A_L + \frac{\partial R_M}{\partial Q_L}\delta Q_L + \left(\frac{\partial R_M}{\partial A_R} + \omega\alpha A_R^{(\omega-1)}\frac{\partial R_M}{\partial Q_R}\right)\delta A_R \quad (3.106)$$

Once again, to simplify the notation, we can substitute algebraic variables (previously defined in Eqns. 3.30 and 3.31) for the partial derivative terms as follows:

$$-R_{C1}^{\ n} = a_1\delta A_L + b_1\delta Q_L + c_1^*\delta A_R \quad (3.107)$$

$$-R_{M1}^{\ n} = p_1\delta A_L + q_1\delta Q_L + r_1^*\delta A_R \quad (3.108)$$

In this example, c_1 and r_1 have changed to c_1^* and r_1^* which are defined by:

$$c_1^* = \frac{\partial R_C}{\partial A_R} + \omega\alpha A_R^{(\omega-1)}\frac{\partial R_C}{\partial Q_R} \quad (3.109)$$

And,

$$r_1^* = \frac{\partial R_M}{\partial A_R} + \omega\alpha A_R^{(\omega-1)}\frac{\partial R_M}{\partial Q_R} \quad (3.110)$$

Now, to generate an equation to calculate δA_R, we can rework Eqn. 3.48 with a few changes to reflect the updated boundary conditions. Therefore Eqn. 3.48 rewritten for the last cell, and including c_1^* and r_1^* is:

$$-(p_1+q_1E_0)(R_{C1}+b_1F_0)-(p_1+q_1E_0)c_1^*\delta A_1+(a_1+b_1E_0)r_1^*\delta A_1=-(R_{M1}+q_1F_0)(a_1+b_1E_0) \quad (3.111)$$

Rearranging the terms in the equation we get:

$$\delta A_1\left((a_1+b_1E_0)r_1^*-(p_1+q_1E_0)c_1^*\right)=(p_1+q_1E_0)(R_{C1}+b_1F_0)-(R_{M1}+q_1F_0)(a_1+b_1E_0) \quad (3.112)$$

Then δA_1 equals:

$$\delta A_1=\frac{(p_1+q_1E_0)(R_{C1}+b_1F_0)-(R_{M1}+q_1F_0)(a_1+b_1E_0)}{(a_1+b_1E_0)r_1^*-(p_1+q_1E_0)c_1^*} \quad (3.113)$$

We can simplify this equation by reintroducing the temporary variables that were previously defined in Eqns. 3.77, 3.78, 3.81 and 3.82:

$$\delta A_1=\frac{T_{(2)}T_{(5)}-T_{(1)}T_{(6)}}{T_{(1)}r_1^*-T_{(2)}c_1^*} \quad (3.114)$$

This equation is called immediately after the end of the forward sweep. After calculating δA_1, Eqn. 3.99 is called to calculate the corresponding normal flow associated with this cross-sectional area. The backward sweep calculations then follow as per usual.

3.4.5 Lateral flow conditions

A stage may be reached during the recession phases of the simulation whereby flowrates become very small, leading to very little downstream propagation of the flow profile. In the case of simultaneous advance and recession, this may result in the advance front of the surface profile receding back upstream, as infiltration dominates the volume-balance. At this stage, solution of the full momentum equation is prone to instability problems. In this case, the simulation engine can directly transform this surface water into the infiltrated volume over incremental time-steps, with little effect on the simulated performance figures.

Therefore, for each cell (i) with surface water present, and while the surface water volume is greater than the incremental infiltrated volume for the time-step $z(dt)$, the cumulative infiltration volume $Z_{(t,i)}$ is calculated by:

$$Z_{(t,i)}=Z_{(t-1,i)}+z(dt) \quad (3.115)$$

$$A_{(t,i)}=A_{(t-1,i)}-z(dt) \quad (3.116)$$

For the very last remaining portion of surface water, which is less than $z(dt)$:

$$Z_{(t,i)}=Z_{(t-1,i)}+A_{(t-1,i)} \quad (3.117)$$

$$A_{(t,i)}=0 \quad (3.118)$$

3.4.6 Boundary conditions

Boundary conditions can be segregated based upon the different phase-combinations of the simulation (Table 3.2).

Table 3.2: Boundary conditions for different phase combinations

Phase	Upstream conditions	Downstream conditions
Advance	$Q_{0,t} = Q_{in,t}$	$Q_{Lastcell,t} = 0$ $A_{Lastcell,t} = 0$
Storage with blocked-furrow	$Q_{0,t} = Q_{in,t}$	$Q_{Lastcell,t} = 0$
Storage with runoff	$Q_{0,t} = Q_{in,t}$	$A_{Lastcell,t} = \left(\frac{Q_{Lastcell,t}^{n}}{\sqrt{\rho_1 S_o}} \right)^{\frac{2}{\rho_2}}$
Depletion with block-furrow	$Q_{0,t} = 0$	$Q_{Lastcell,t} = 0$
Depletion with runoff	$Q_{0,t} = 0$	$A_{Lastcell,t} = \left(\frac{Q_{Lastcell,t}^{n}}{\sqrt{\rho_1 S_o}} \right)^{\frac{2}{\rho_2}}$
Recession with blocked-furrow	$Q_{Firstcell-1,t} = 0$ $A_{FirstCell-1,t} = 0$	$Q_{Lastcell,t} = 0$
Recession with runoff	$Q_{Firstcell-1,t} = 0$ $A_{FirstCell-1,t} = 0$	$A_{Lastcell,t} = \left(\frac{Q_{Lastcell,t}^{n}}{\sqrt{\rho_1 S_o}} \right)^{\frac{2}{\rho_2}}$
Advance and depletion	$Q_{0,t} = 0$	$Q_{Lastcell,t} = 0$ $A_{Lastcell,t} = 0$
Advance and recession	$Q_{Firstcell-1,t} = 0$ $A_{FirstCell-1,t} = 0$	$Q_{Lastcell,t} = 0$ $A_{Lastcell,t} = 0$
Advance (last cell)	$Q_{0,t} = Q_{in,t}$	$Q_{Lastcell,t} = 0$ $A_{Lastcell,t} = 0$ $dx^{t} = FieldLength - X_{lastcell-1}^{t-1}$
Advance and recession (last cell)	$Q_{Firstcell-1,t} = 0$ $A_{FirstCell-1,t} = 0$	$Q_{Lastcell,t} = 0$ $A_{Lastcell,t} = 0$ $dx^{t} = FieldLength - X_{lastcell-1}^{t-1}$

3.4.7 Initial parameter estimates

Unfortunately, the iterative double-sweep methodology has a limited range of convergence. Good initial starting estimates are crucial for convergence on the final solution. Fortunately, the starting information can usually be obtained from the final solution at the previous time-step. Table 3.3 presents formulae for the starting estimates for different irrigation phases.

Table 3.3: Initial parameter estimates for different irrigation phases.

Phase	Initial Estimates
Advance (first cell)	$Q_0^* = Q_{in} - \left(ak\left(dt^{a-1}\right) + fo\right),\quad A_0 = \left(\frac{\left(nQ_0^*\right)^2}{\rho_1 S_0}\right)^{\frac{1}{\rho_2}},\quad dx = \frac{Q_{in}dt}{A_0}$
Advance	For x>1, $Q_x^t = Q_{x-1}^{t-1},\quad A_x^t = A_{x-1}^{t-1},\quad dx^t = dx^{t-1}$
Storage with blocked-furrow	$Q_x^t = Q_x^{t-1},\quad A_x^t = A_x^{t-1}$
Storage with runoff	$Q_x^t = Q_x^{t-1},\quad A_x^t = A_x^{t-1}$
Depletion with block-furrow	$Q_x^t = Q_x^{t-1},\quad A_x^t = A_x^{t-1}$
Depletion with runoff	$Q_x^t = Q_x^{t-1},\quad A_x^t = A_x^{t-1}$
Recession with blocked-furrow	For x>1, $Q_x^t = Q_{x-1}^{t-1},\quad A_x^t = A_{x-1}^{t-1}$
Recession with runoff	For x>1, $Q_x^t = Q_{x-1}^{t-1},\quad A_x^t = A_{x-1}^{t-1}$
Advance and depletion	For x>1, $Q_x^t = Q_{x-1}^{t-1},\quad A_x^t = A_{x-1}^{t-1},\quad dx^t = dx^{t-1}$
Advance and recession	For x>1, $Q_x^t = Q_{x-1}^{t-1},\quad A_x^t = A_{x-1}^{t-1},\quad dx^t = dx^{t-1}$
Advance (last cell)	For x>1, $Q_x^t = Q_{x-1}^{t-1},\quad A_x^t = A_{x-1}^{t-1},$ $dx^t = FieldLength - X_{lastcell-1}^{t-1}$
Advance and recession (last cell)	For x>1, $Q_x^t = Q_{x-1}^{t-1},\quad A_x^t = A_{x-1}^{t-1},$ $dx^t = FieldLength - X_{lastcell-1}^{t-1}$

This table combined with the boundary conditions in Table 3.2 is sufficient to achieve convergence in almost all cases. However, sometimes, special treatment is required at times of rapid system change such as the transition between advance and runoff, and also when the inflow is terminated. These treatments will be discussed later in this chapter.

3.4.8 Parameter constraints

As the solution variables oscillate around their true values during convergence, they need to be constrained to physically realistic values to maintain system harmony. All of the solution variables must remain positive, while the inflow rate and corresponding normal depth of flow influence the upper-limit for flowrate and flow-area. Therefore, parameter limits include:

$$0 \le Q_{(x,t)} \le Q_{in} \quad (3.119)$$

$$0 \le A_{(x,t)} \le A_n \quad (3.120)$$

$$0 \le dx_{(x,t)} \le 100 \text{ (nominal value)} \quad (3.121)$$

The effect of these constraints on convergence will be discussed later in this chapter.

3.5 Computer algorithm development

Having derived the mathematical equations for the model and its solution, the next step in developing the simulation engine was to derive a computer algorithm to manage and control the simulation operation. In doing so, six key elements of the computer algorithm needed to be carefully defined including:

- the structure of the simulation engine;
- the simulation model algorithm;
- the input parameter objects;
- the output parameter objects;
- the phase switching mechanism; and
- exception handling facilities (error management).

3.5.1 Developing a structure

The *FIDO* simulation engine has been developed using an object-oriented structure using the C++ language. While simulation speed was a design issue, the object-oriented design approach was adopted despite the view that well written procedural code can potentially run faster. The benefits of the object orientated approach in this instance include:

- Simpler code structure;
- Easier readability of code; and
- Multiple input/output objects.

Designing an object-oriented algorithm involves breaking up the structure of a system into different modules and objects that can interact with each other and also with external elements. These objects should have some level of "intelligence" being able to perform tasks and follow instructions. They should also have a memory, being able to store information. Decisions need to be made at the design stage regarding the level of modularity required for the system. For the case being considered here, this level need not be high at all. The simulation engine could in fact operate as a simple linear system with one set of inputs and one set of outputs; a "black box" in effect (Figure 3.9).

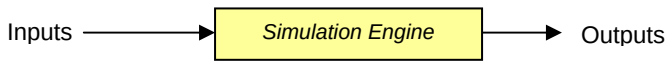


Figure 3.9: Basic linear or "Black Box" functionality of simulation engine.

The engine itself would be an object that encapsulates the model code, solution techniques, and internal parameters. The input data and output data would also be objects that contain parameters (which again are objects) and can be passed in and out of the engine. The simulation engine would also contain other internal parameter objects that are inaccessible to other parts of the program (otherwise, they should be part of the input object). Figure 3.10 shows the storage structure of these objects as used in *FIDO*. The output-object is located inside of the input-object for convenience; it is simpler to pass "one" object around the decision support system than it is for "two".

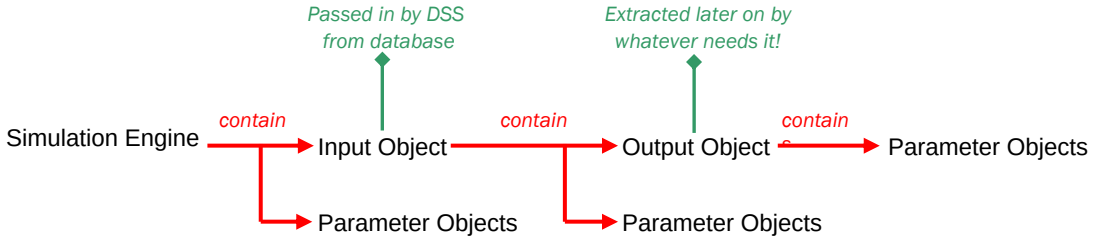


Figure 3.10: Simulation Engine Object Structure

In reality, the connections between the objects are varied. The algorithms require extensive use of memory-pointers that essentially give the objects "hands" by which they attach to other objects. This alleviates the continual copying of information (which is slow) when passing from one object to another. This is achieved through careful design of the parameter objects that make use of polymorphism to alleviate repetition of code between the different parameter types. In this case, the performance penalty (operation speed) associated with using polymorphic parameter objects was overshadowed by their benefits. The final design of these parameters is actually quite complex given the amount and variety of tasks that they are required to perform (this will be discussed later in this chapter). Figure 3.11 provides a more in-depth look at how all of these objects interact.

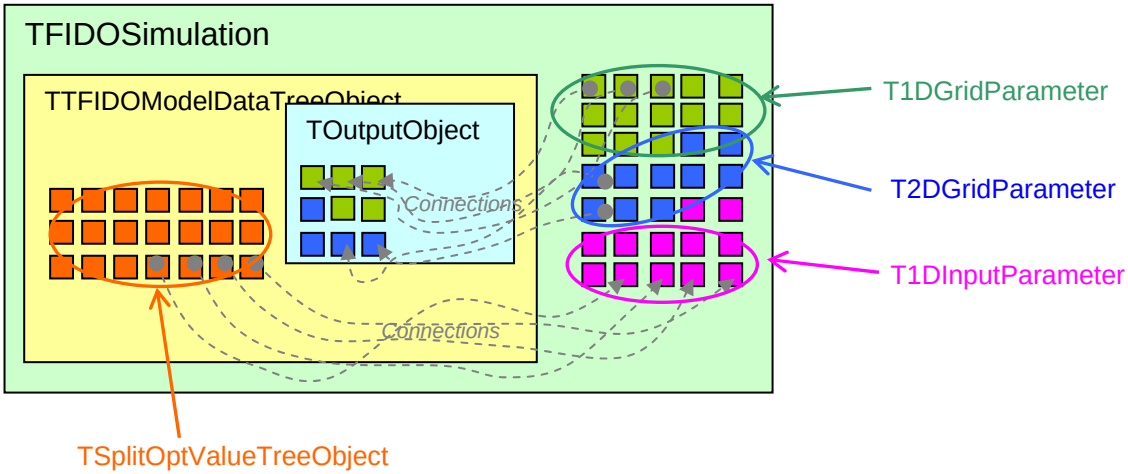


Figure 3.11: Parameter object and model-object interaction in the *FIDO* simulation engine.

In this example, the breaking down of the structure is biased towards the data components rather than the mathematical components. This can be justified since the decision support system is not required to interact with the simulation engine outside of this basic input/output functionality. For this reason there is no need to further modularise the model code. An added advantage of this type of structure is that it allows for many data-objects to exist in memory simultaneously (although processing by the simulation engine occurs one at a time) allowing for easy comparison of different results, and input data.

Other alternatives to this design were investigated during the development process. However, they were considered too complex with little or no extra benefit in meeting the design specification of the simulation engine.

One alternative was to use polymorphism to create a custom "simulation (engine) object" and derive child objects from this for different irrigation configurations. (e.g. "blocked" vs. "free-draining" furrow end conditions). However, the complexity involved in switching scenarios would have been the same whether it was performed inside or outside of the simulation engine. This would also have added the extra burden in managing memory, whereby the simulation engine would need to be created and deleted each time a different scenario was modelled. With the current design, the simulation engine is created at program start-up, and remains there until the program is closed down. Virtual methods associated with polymorphism could also have slowed down simulation times unnecessarily if they are called during each iteration. However, one possible advantage of this technique would be in future development work, whereby other developers could more easily create new variations of the simulation without altering the original code. This is still possible with the existing design, but it would possibly have been simpler and more flexible using polymorphic techniques.

Another alternative that was considered was to split up the simulation engine object at the "irrigation-phase" level. This would involve having a single "simulation (engine) object" and using polymorphism again by creating a custom phase object, and deriving child objects for each different phase of the irrigation (e.g. "advance", "storage" "recession", etc). The phase-objects would contain the solution techniques, boundary conditions and initial conditions for each phase. This could simplify the code by removing much of the conditional statements from the code. Each object would encapsulate functions specific for that phase, including error handling techniques. However, the problems outlined in the previous example could again surface here with polymorphism potentially slowing down simulation times. There would still be memory management issues; *do we create/destroy phase objects each time we change phases, or possibly create every single phase object in memory at once?* Another bottleneck could be the copying and transferring of information in and between phase objects.

Putting time-penalties through overuse of polymorphism aside, the main reason for going with the suggested model structure was because it was the simplest.

3.5.2 Model algorithm

Figure 3.12 outlines the model algorithm used by the simulation engine when running a simulation. The algorithm revolves around two main loops; one for incrementing the time-step, and one for incrementing the iteration step. When implementing this algorithm, modular coding is used to eliminate code repetition and simplify development. If a piece of code needs to be accessed from more than one place, then it is placed in its own unique method. Breaking the code down this way into its smallest elements naturally encourages modular programming, simplifying the algorithm and code maintenance and minimising

the likelihood of coding errors. Use of good naming conventions of these methods tends to self-document the code, especially when called from a controller method containing nothing more than method-calls and conditional/logic statements. This is demonstrated by comparing the `Simulate()` method in the simulation engine source code (Appendix 3.1) with the algorithm in Figure 3.12.

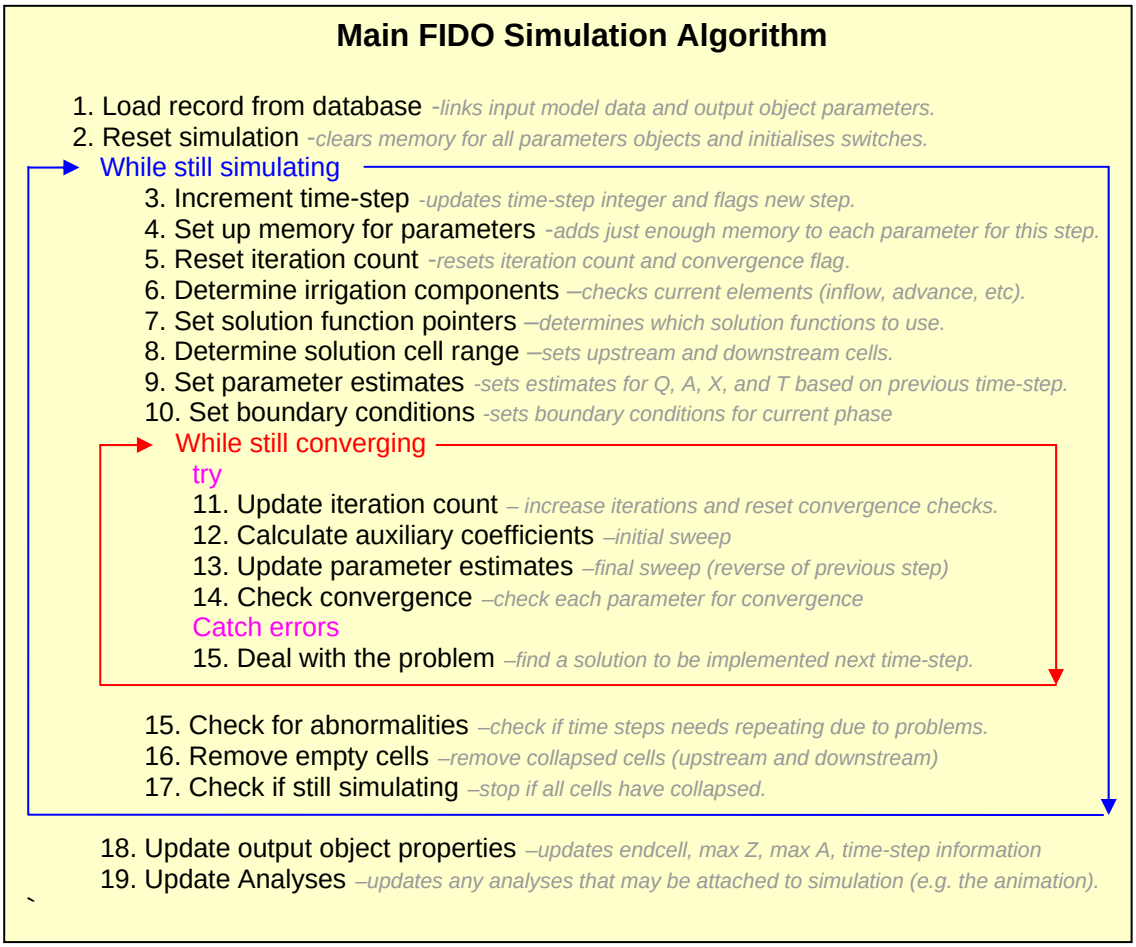


Figure 3.12: Algorithm used in the simulation engine for running simulations.

Another feature of the algorithm is that it allows output to be generated even in the case of simulation failure. Although the current version of the simulation engine has proved to be extremely robust, earlier development work had identified the need to provide sensible output in the event of such a failure. External analyses attached to the simulation engine behave unpredictably if unrealistic results are fed into them. Therefore the algorithm is designed to output the last “good” set of results from the simulation, which in the case of a terminated simulation, would be results generated from the time-step before failure occurred. In most cases, the final performance measures would be very close to the true values, since these types of failures occur predominately during the latter parts of the simulation after the advance has completed.

3.5.3 Input parameter objects

Object-orientated programming techniques were used to create input parameter objects to represent the solution variables such as Q, A, dx and dt (among others). This effectively adds a "brain" to the parameters so that they have both processing and memory management capabilities as well as the usual storage functionality. Traditionally, these parameters would have been an array of "floats" or "doubles". Now they contain information about upper and lower constraints, temporary storage vectors, solution tolerances, solution cell information (including differentiation between Eulerian and Lagrangian grids), and functions for setting initial values, updating new estimates and adjusting storage sizes.

Three different types of data storage objects have been used in the simulation engine, which have been called `T1DGridParameter`, `T1DInputParameter`, and `T2DGridParameter` objects. These are all ultimately derived from the "abstract base class" (which by definition, cannot be used to store data) called `TCustomGridParameter`, which implements the polymorphism functionality (by introducing "virtual" methods") and setting the structure of its children. Figure 3.13 demonstrates the relationships and key functionality of the parameters.

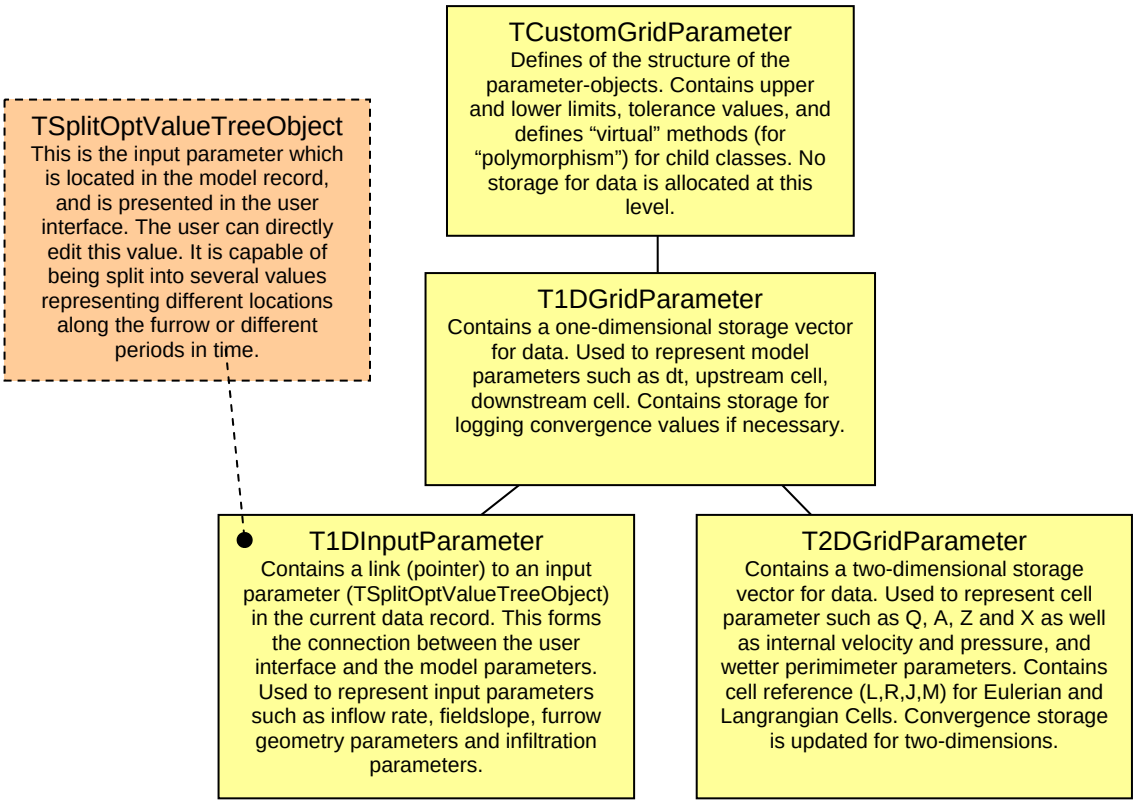


Figure 3.13: Structure of the object-oriented input parameter types used in *FIDO*. Note that the T1DInputParameter contains a pointer to an input parameter located in simulation input object.

A multi-value input parameter object called `TSplitOptValueTreeObject` has been created to house **spatially and temporally variable** input parameter values (such as for non-uniform infiltration, roughness, furrow geometry, field-slope and

inflow). This parameter object is linked to the input object through a pointer in a corresponding `T1DGridParameter` parameter object.

All of child parameter-object types have been designed to manage memory efficiently. At any one time, the storage containers inside of these objects have been sized exactly to that of the current memory demand. That is, bulk memory allocation is not employed, rather memory is allocated dynamically at each time-step according to the current requirements. For the `T2DGridParameter` types, this involves a triangular storage structure as the number of cells on the solution grid increases (during the advance phase of the irrigation) with each time-step (Figure 3.14).

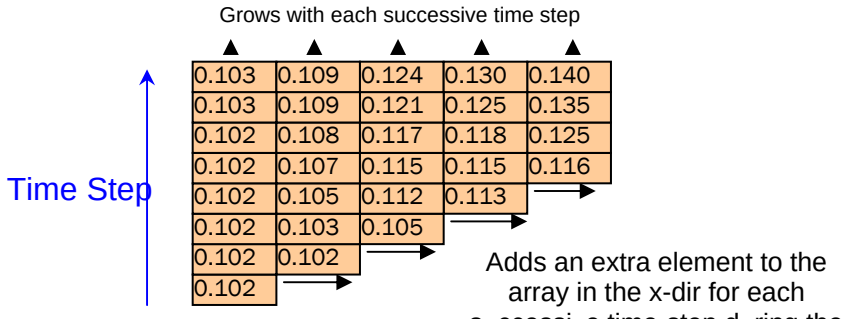


Figure 3.14: Memory allocation technique employed by theT2DGridParameter types.

3.5.4 Output objects

Storing the results of the simulation engine externally in a specially designed object allows the user to compare the results of many different simulations simultaneously. A single output object `TSimulationOutputObject` is used to store all of the simulation outputs described in Section 3.5.5. Outside of the basic storage functionality, the object contains the methodology to calculate the irrigation performance parameters for any time-step.

3.5.5 Phase switching

The simulation engine employs a special C++ container type called a *set* (an associative container that supports unique keys, allowing fast retrieval of the keys) to keep track of the physical processes occurring during an irrigation simulation. This *set* (called *`TPhaseComponents`*) can contain up to six different elements relating the state of the irrigation (although due to obvious physical limitations, they can not all occur simultaneously). This includes advance, recession, inflow, runoff, ponding, and lateral flow phases.

Figure 3.15 describes the logic in defining the contents of the *set*. At the start of the simulation, the *set* is initially empty. Then at the beginning of each time-step, this algorithm is called to see which of these elements are currently active. Any active elements are added to the *set*, and any inactive elements are removed. At any stage during the calculations, the *set* can quickly be examined to aid in the internal decision-making. Although a speed penalty is incurred through extra conditional operations during the calculations, the conditional checks are fast since no values or functions need to be examined.

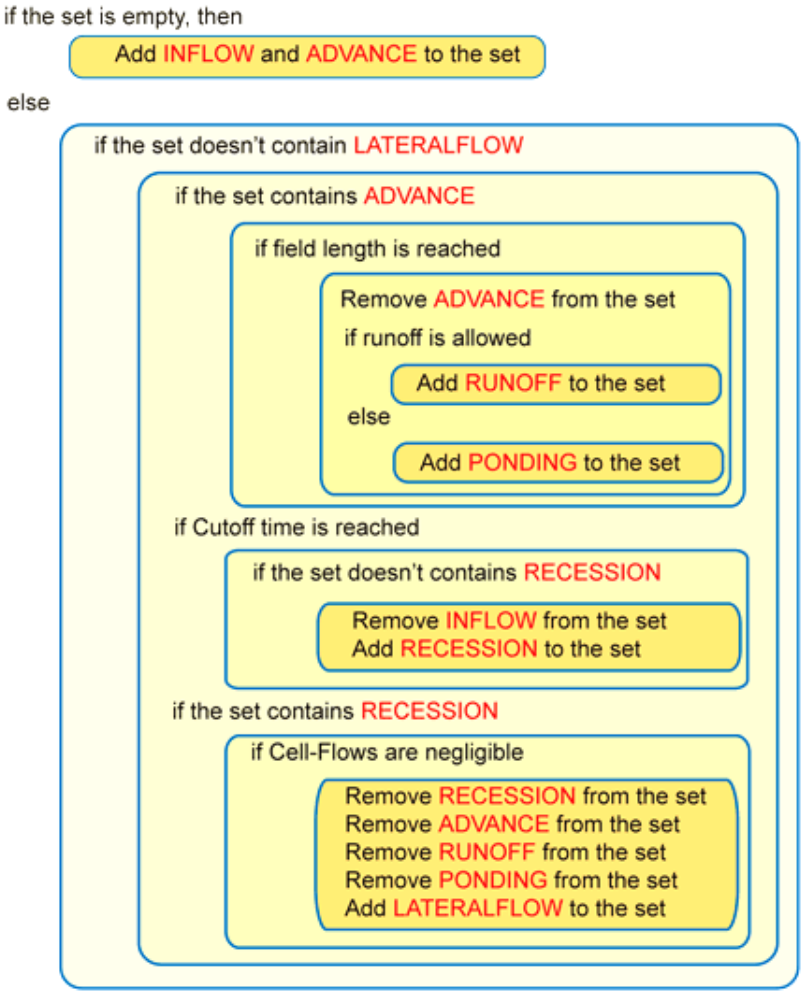


Figure 3.15: Algorithm for adding/remove phase component to the "set"

In early versions of the simulation engine, *sets* were not used. Instead, a series of switches were used for the individual phases and phase-combinations of the irrigation cycle. During each time-step, a "checking-function" would be called whereby the current state of irrigation phase was decided upon (from about fifteen different alternatives), and function-pointers were assigned according to this state. During simulation, these function pointers were called instead of the actual required function. This proved quite powerful, dramatically cutting down on the number of conditional statements required throughout the calculations, with all of the decision making being done in the initial "checking-function". However, the problem was that it resulted in an excessive amount of code with many functions repeated with only minor changes between them. This simulation engine became very difficult to work with from the developer's perspective. It was tremendously difficult to debug, and there was little confidence in any part of code because there was just too much to manage effectively. The use of *sets* in the latest version has reduced the total volume of code to a third of the original code amount.

3.5.6 Exception handling

Exception handling (error management) has been implemented at potential problem sources minimising the impact on the simulation progress. It aims to improve reliability allowing convergence problems to be identified and handled with a more appropriate grid structure or better initial parameter estimates. The try/catch statements implemented around Steps 11-14 in the algorithm (
Figure 3.12) aim to trap errors during the double-sweep solution process, where they are most likely to occur. If an error does occur at this level (normally a convergence error), the time-step can be adjusted or restarted without user-intervention through exception handling techniques. Sometimes erroneous results can occur despite apparent convergence success. These abnormalities are checked outside of this loop to verify that the simulation is behaving, as it should and that convergence wasn't achieved through unusual circumstances.

3.6 Observations on simulation characteristics

Having derived the simulation mathematics and program algorithms, the next step in the development process was to combine these by writing the computer program, which is presented in Appendix 3.1. From this point on, research focused upon refining, testing and debugging the program. A key part of this work was directed at studying and observing the characteristics of the simulation. Based upon these observations, measures were then implemented to improve robustness and reliability. Therefore, several key observations of the characteristics of the simulation will now be discussed including:

- cell sizes decrease towards the downstream end;
- volume-balance errors are greatest at furrow inlet;
- instability is greatest during rapidly changing conditions;
- grid structure influences simulation performance;
- recession approximations can lead to instability; and
- transition to runoff can cause oscillations in runoff hydrograph.

3.6.1 Cell sizes decrease downstream

During the advance phases of the simulation, nodes on the solution grid are created based upon the position of the advance front at each time-step. Therefore an extra node is created for each successive time-step. By the time the advance reaches the end of the field, the node spacings are much smaller then they are at the top end of the field. One advantage is that the smaller sized solution cells are located in positions where the later stages of the recession will predominate. This helps in the solution of a potential weak link in the simulation. Unfortunately, the larger cells are located at the top end of the field where a finer grid resolution is also required to capture rapidly changing inflow conditions.

3.6.2 Sources of volume-balance error

It was observed that volume-balance errors predominate at the upstream end of the furrow. The top end of the furrow is the location of the most dynamic and rapidly changing quantities of the simulation; that of instantaneous inflow and instantaneous cut-off. In the first instance, the system instantaneously switches between no-flow and high-flow conditions. Percentage-wise, the volume-balance

error during this time-step could be as high as 30% to 40%, but volumetrically it is still very small in terms of the whole irrigation water balance. In the second instance, the flow change is reversed from high-flow to no-flow, but it still occurs instantaneously. On both of these occasions, the effect will be most pronounced at the upstream end of the furrow. Therefore, it is recommended that extra nodes should be placed at this location to help account for the rapidly changing conditions.

Another source of volume-balance error can be attributed to numerical instability. Approximations are made during the recession stages of the irrigation which can cause convergence problems leading to volume-balance errors. Effectively, system accuracy is often compromised to achieve numerical stability. Fortunately, these compromises usually result in only a very small error in terms of the overall volume-balance.

3.6.3 Sources of instability

Instability in the solution technique is most likely to occur during stages of rapidly changing conditions. This is primarily due to the difficultly in obtaining good parameter starting values during these periods. This is especially noticeable during phase transitions from advance to storage, from storage to depletion, and from depletion to recession, and also during later stages of the recession. Figure 3.16 show a progression of time-step iteration counts for a typical simulation, highlighting periods of instability.

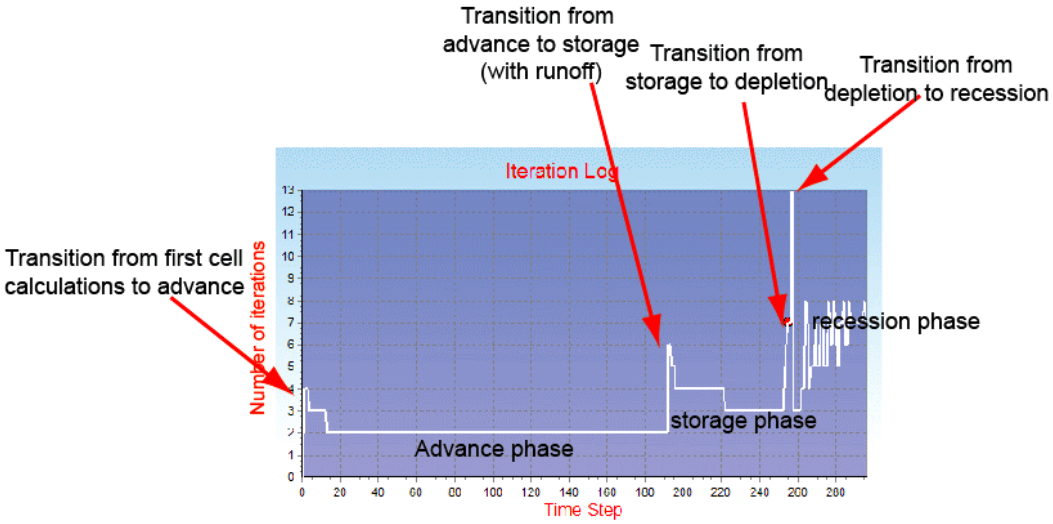


Figure 3.16: Typical iterations log for different irrigation phases

Some general observations from studying convergence during the course of this work include:

- Good initial starting estimates are crucial for avoiding stability problems.
- The advance phase is the most robust simulation phase, typically converging in two iterations. It has proven consistently reliable under all conditions.
- The storage phase is also generally robust, once the transition from advance to storage is completed. However, difficulty is often encountered

during this transition, although it rarely requires special treatment to achieve convergence.

- The transitions from storage to depletion and depletion to recession are key sources of instability, given the difficulties in obtaining good initial estimates of the surface and flow profiles.
- Most failures occur during the recession phases of the simulation, especially when the recession is very rapid, and flow rates and flow depths are very low. For example, saw-tooth fluctuations in the surface and flow profiles have been observed during the later stages of the recession.

Parameter convergence monitoring capabilities were embedded into the *FIDO* simulation engine to study the nature of instability patterns and convergence issues on the individual solution parameters Q, A, dx and dt. This was a key debugging tool that has contributed to four strategies for improving simulation robustness presented later in this chapter.

These capabilities provide insight into the sources of instability during periods of convergence difficulties. For example, Figure 3.17 shows an output of this extended analysis corresponding to a transition from the depletion phase to the recession phase in a typical irrigation (as shown by the peak in Figure 3.16). The two surface plots represent the absolute values of incremental parameter changes for A and Q for different node positions and iterations. In this example, the source of instability is located in the region of rapidly changing conditions at the upstream end of the furrow.

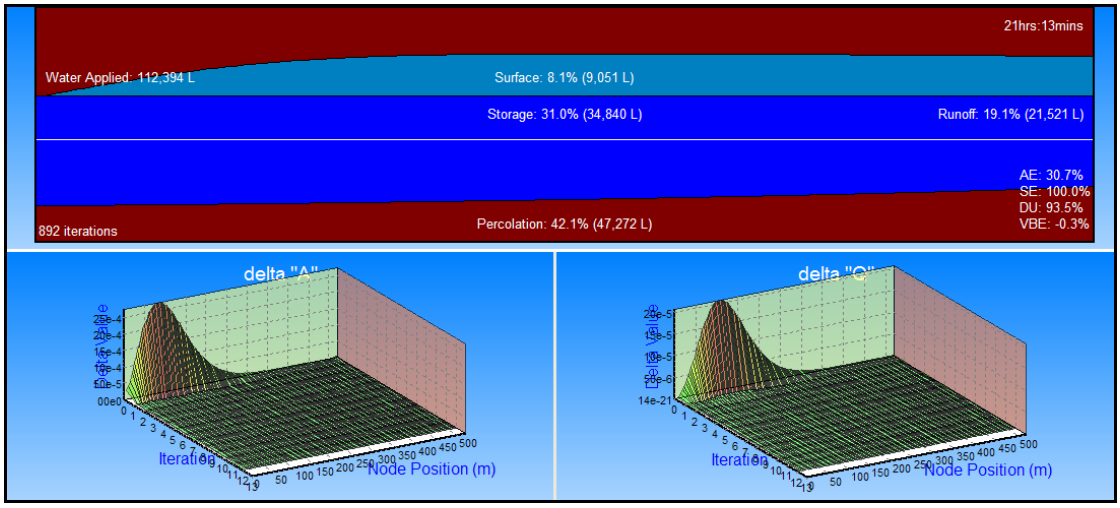


Figure 3.17: Convergence Log for A and Q parameters during transition from depletion phase to recession phase. The surface water and infiltration profiles are shown in the top chart. Convergence was achieved in 12 iterations.

Another example (Figure 3.18) highlights the difficulties in achieving convergence during the later stages of the recession. This example resulted in catastrophic failure of the simulation, which was terminated after 100 iterations of the time-step. A low-flow, low-volume surface profile was present during the previous time-step. Solutions to this problem are presented later in this chapter.

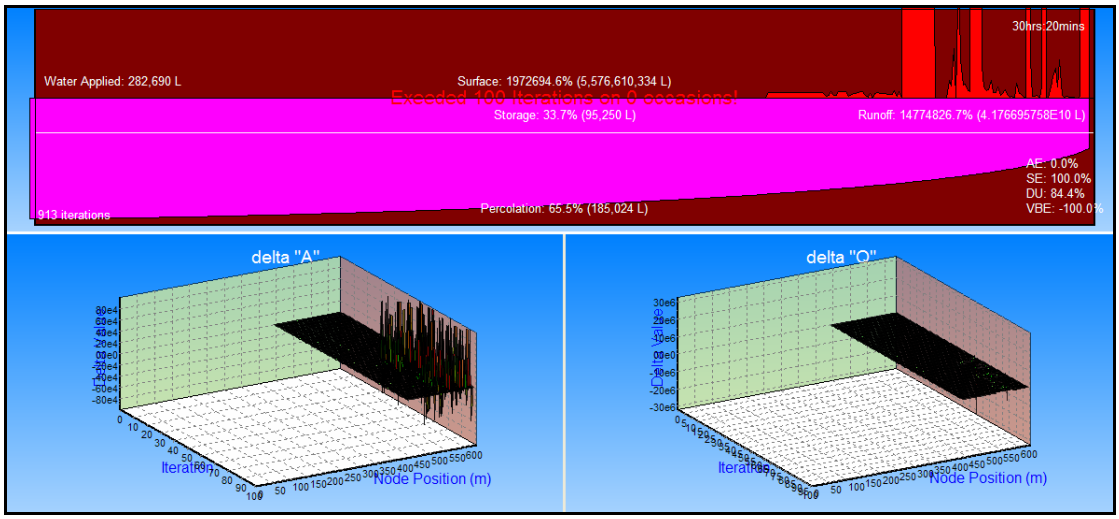


Figure 3.18: Example of convergence failure during the recession phase.

3.6.4 Effect of solution grid structure.

It was found that another source of error exists if the nodes on the solution grid are spaced erratically. During repeated trials with different configurations of the simulation engine, the most stable and reliable simulations occurred when the solution grid was regular. Note this does not imply that the solution grid was "uniform", but rather that changes in cell sizes occurred gradually from one cell to the next. For example, Figure 3.19 demonstrates the effect on the solution grid of increasing and decreasing the time-step midway through the simulation. Errors have been introduced in both cases due to the sudden change in time-step. It was noted that convergence was also slower in both these cases.

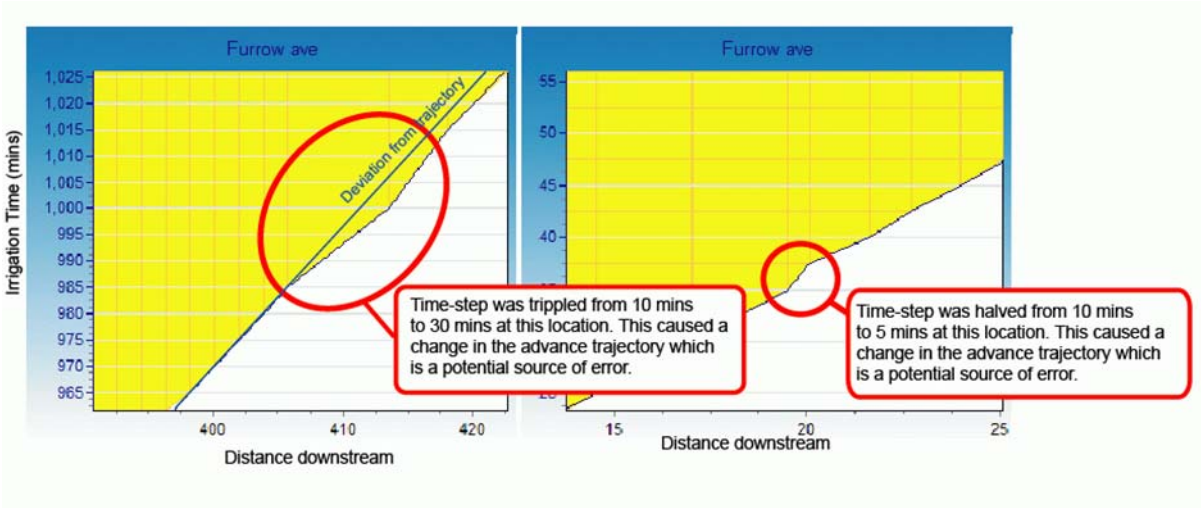


Figure 3.19: Effect of a sudden change in time-step on advance trajectory.

Consequently, this could imply problems when solving for the unknown time-step parameter (as in the fixed-distance-step method), especially when the advance is very slow and struggling to reach the next node on the solution grid. In this example, the time-step solution could be very large, with a subsequent penalty in stability and volume-balance error. For this reason, the fixed-time-step

methodology was preferred over this method when designing this simulation engine.

This also highlights that the implicit double-sweep solution technique is reliant upon stability restrictions on the solution grid structure. This is despite popular belief that only the explicit solution techniques are subject to this condition. Other authors have also verified this. For example, Strelkoff and Falvey, (1993) reported that *"... implicit schemes in which the simultaneous equations are solved with a double-sweep method- having one boundary condition picked up at the upstream end ... and another picked up at the downstream end - exhibit oscillations if the Froude number exceeds unity by more than just a little or for more than just a short time*".

Since the $FIDO$ simulation engine uses a fixed time-step to implicitly locate nodes in the x-direction, careful management of this time-step size is required to avoid convergence problems.

3.6.5 Recession approximations can cause instability

All existing surface irrigation models make approximations during the recession phases. That is, none of the existing models actually solve simultaneously for the time for the recession to pass each node[6]. Typically, the recession is defined when the depth of flow at a cell node is less than 5% of the normal depth. However, numerically speaking it is not uncommon for the depth of flow to be negative before the recession flag is activated (Figure 3.20). It was observed that this can cause irregularities in the recession curve due to the true recession point not coinciding with the nodes on the solution grid. Although it may have little effect on the accuracy of performance results, the effect is more pronounced on the reliability of the solution technique with convergence problems occurring in extreme cases. To deal with this, a monitoring system needs to be implemented to check for convergence problems at this location. This will be discussed later in this chapter.

[6] During this research, considerable time was spent trying to solve implicitly for both dx and dt during the simultaneous advance and recession phase of the irrigation. This involved deriving a new double-sweep algorithm whereby the direction of the two sweeps is reversed. Unfortunately, robustness could not be achieved, and the solution matrix was ill-conditioned. That it, there was an infinite number of solutions within the range of the remaining simulation times, as the cell flow-rate and flow-area could turn out to be 0 in the solution. This work was abandoned in favour of the more reliable approximation method. However, it may be possible to derive a single unique solution with different techniques and further research.

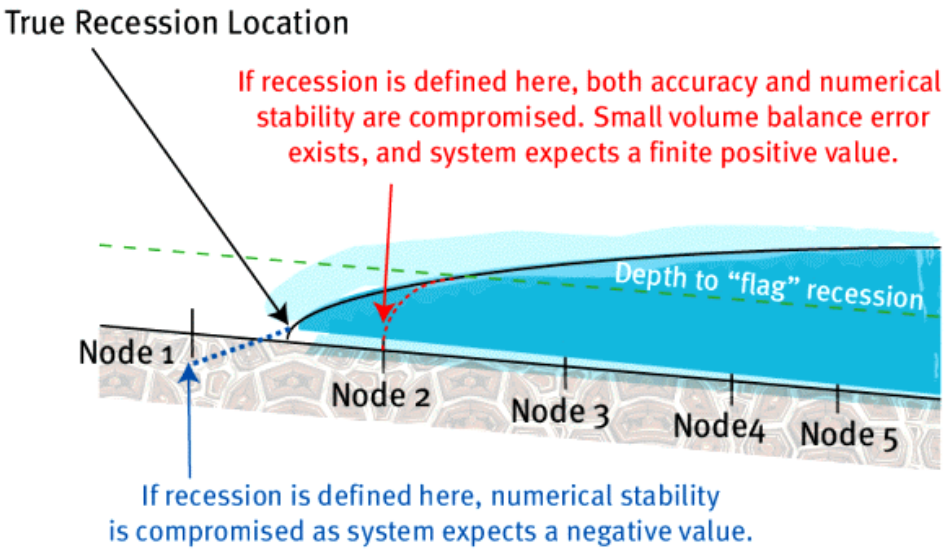
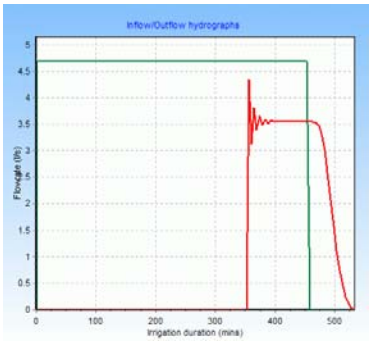


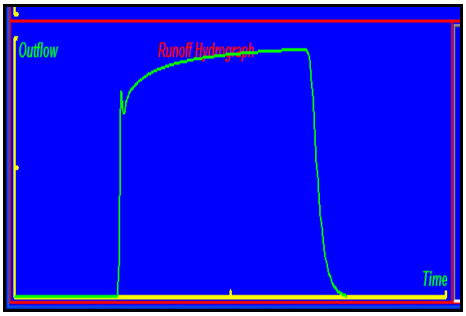
Figure 3.20: Problems with recession definition

3.6.6 Transition to runoff

It was often observed that the simulated transition from advance to runoff caused a fluctuation, or dampened oscillation, in the runoff hydrograph (Figure 3.21a). This characteristic is not unique to this simulation engine, but was also observed to occur in *SIRMOD* (Figure 3.21b).



(a)



(b)

Figure 3.21: Fluctuations in runoff hydrograph in (a) *FIDO* simulation engine and (b) *SIRMOD* output.

This phenomenon seems to occur more frequently and severely when the time-step size is suddenly reduced in order to position the last node with the end of the field (e.g. if the time-step was reduced from 5mins to 2mins for the last advance step). It was also observed to occur when initial starting estimates of the furrow end parameters are poor. In either case, convergence is usually achieved but the solution parameters are obviously inaccurate, although its effect on the overall accuracy of the simulation is thought to be minimal. These errors dampen quickly over successive time-steps as the furrow end conditions become more uniform. It appears that the system is weakly linked to the boundary conditions that are insensitive to the parameter variations, but this

requires further investigation. Fortunately, this situation can usually be avoided by ensuring that the time-step size for the last advance step is close to, or larger than, the default time-step. This may require "undoing" the "previous" time-step before adopting a larger time-step size to match endpoints.

3.7 Achieving simulation robustness.

It quickly became apparent when writing this computer program that deficiencies in the published methodologies are understated. Given the lack of information that exists on undertaking these transformations, it is important to document these idiosyncrasies and their solutions for future reference. The following techniques have been employed in the *FIDO* simulation engine to achieve the design goals of robustness and reliability:

- implement small time-steps for first few iterations;
- monitor parameter convergence during iterations;
- pre-test time-steps to remove collapsing cells; and
- automatic time-step refinement.

3.7.1 Early time-step calculations

It was previously mentioned that the upstream end of the furrow is potentially the main source of volume-balance error and numerical instability. This is because it is a location of rapidly changing conditions; that is, when the inflow is turned on and also turned off. It is also the location of the largest solution cell, which is generated during the first time-step. One way to minimise these problems is to introduce smaller solution cells in this region. This can be done by using small time increments for the first few time-steps:

$$\text{for } i <= 5; \quad dt_i = \frac{i}{5} dt_{default} \qquad (3.122)$$

where i is the current time-step index.

3.7.2 Parameter monitoring during iterations

Publications have typically neglected to mention critical solution problems such as parameter limits. Due to the iterative nature of the solution process, convergence on the final parameter values is obtained after oscillation around the true parameter values. If the true parameter values are very small, then the solution parameters may take on unrealistic or negative values during the convergence process. While this isn't really a problem if we consider that we are just solving a large set of equations in matrix form, it is a problem when these equations are representing physical processes and that all sorts of side-effects can spawn if our solution parameter values are unrealistic. Therefore we need to constrain our solution parameters to within a physically realistic range. These constraints must be applied the instant that a limit is exceeded and not just at the end of an iteration, otherwise errors will be introduced in adjacent cells during the marching technique.

Unfortunately, one of the side effects of constraining a parameter is that repeated mirrored-oscillations in the improved parameter values can sometimes occur over successive iterations. Usually, the mirroring of the oscillations occurs

when the constraint is first applied to a parameter, and continues until that same constraint is forced to be reapplied. Then the process restarts over again. Therefore the process becomes stuck in an infinite loop and convergence never occurs (Figure 3.22).

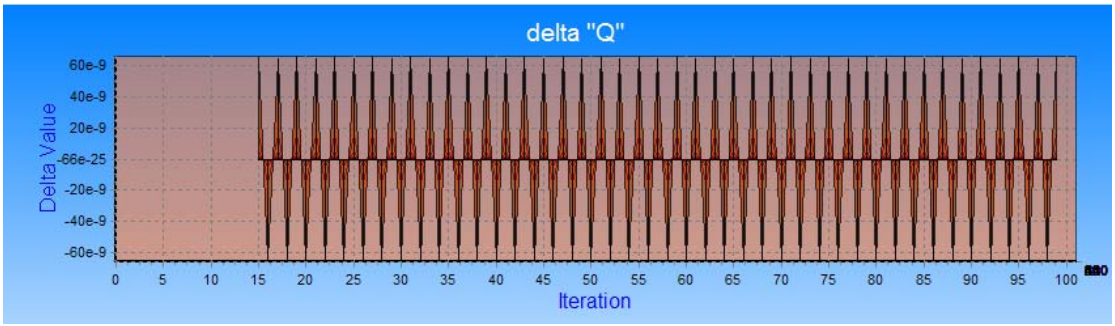


Figure 3.22: Repeating mirrored-oscillations

To overcome this, the solution parameter objects automatically monitor oscillations during each time-step, and report back to the simulation engine when any mirroring effect occurs. This typically happens during the recession phase and results when a finite (but small) depth of flow at the recession front is constrained to zero (as discussed in section 3.6.5); that is, when the true recession position does not coincide with the solution grid node locations.

Once oscillations have been reported, the cell in which the oscillations occur is removed from the current time-step. If the recession is sufficiently rapid (i.e. on steep slopes, or high infiltration soils), it may be possible that several cells are removed in one step during the process. As mentioned earlier, this may compromise solution accuracy (i.e. a very small error could be introduced) for a marked increase in simulation robustness.

3.7.3 Pre-testing time-step to remove collapsing cells

During the recession phase(s), a simple but effective technique is available to reduce instability in the numerical system, by removing upstream cells that are likely to collapse during the time-step, before they collapse. During the recession, there are no inflows into the upstream solution cell, while outflows can involve both infiltration and surface-flows into adjoining cells. While the surface flows cannot be determined until the end of the time-step, the potential infiltration volume is known (or can be estimated if wetted-perimeter effects are being accounted for) at the commencement of the time-step. Therefore, if it is obvious that the potential infiltrated volume for the time-step is greater than the surface water volume in the cell, then the upstream cell can be removed before the time-step is simulated (making sure to account for the infiltrated water). This avoids potentially having to constrain negative flow and storage values during the time step (as explained in Section 3.6.5), which is a major cause of instability. More than one upstream cell can be removed in a single pre-test. However, any subsequent cells must also test for possible inflows by considering the outflow of the adjoining upstream cell at the previous time-step.

3.7.4 Automatic time-step management

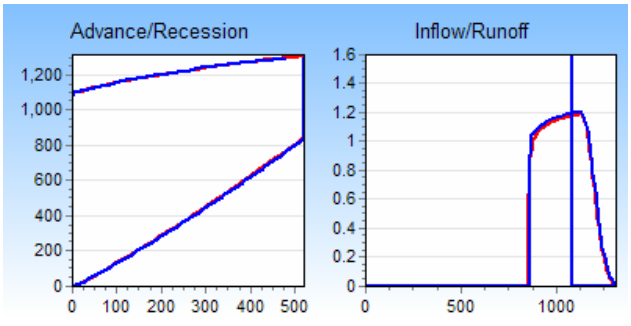
A simple "time-step management" technique has been developed and implemented into the simulation engine to ensure that simulations are completed without convergence errors. Given the simplicity and success of the technique, one could easily suspect that's its role in the simulation process is trivial. On the contrary, the robustness of the simulation is severely compromised without strict adherence of the technique's rules.

It was found during testing that many instability problems could be resolved if the time-step was changed mid-way through a time-step convergence loop. This process involves testing the convergence of the solution variables during the iterations, and if problems were acknowledged, then the time-step would be restarted with a different step-size. Usually, only a change of a few seconds was required to "kick" the simulation through its problem. This procedure was only implemented as a last resort having concluded that repeated oscillations (as outlined in the previous section) were not the source of the problem.

In the rare event of this failing to rectify the problem, the entire simulation would then be rerun using a different default time-step size.

3.8 Validation

The *FIDO* simulation engine has been validated by comparing its results with that from *SIRMOD* (e.g. see Figure 3.23). The literature review (Chapter 2) and case study (Appendix 2.2) confirmed that *SIRMOD* can accurately simulate a wide range of field conditions. Therefore, agreement between the results from *FIDO* and *SIRMOD* would confirm that the *FIDO* simulation engine is also accurate. This hypothesis is reinforced by the fact that the models share the same hydrodynamic model and that the solution techniques are similar albeit for the recession, runoff and stability management techniques.



Flowrate (m^3/sec)	0.0016
Time-To-Cutoff (mins)	1070
Field-Length (m)	520
Field Slope	0.0015
Manning n	0.1
Kostiakov A	0.074
Kostiakov K (m^3/min^a/m)	0.076
Kostiakov Fo (m^3/min/m)	0.00003

Figure 3.23: Sample output of validation of *FIDO* simulation engine against *SIRMOD* results. Blue lines are the *FIDO* output; Red lines are the *SIRMOD* output.

3.8.1 Accuracy of results

Eighteen data-sets representing a variety of free-draining irrigation conditions were used to validate *FIDO* against *SIRMOD*. Appendix 3.2 shows the outcome of these runs by overlaying advance and recession profiles of each model. An example of this output is presented in Figure 3.23. In all cases the advance curves generated by each model were identical, while recession curves were usually similar but not identical. Some differences were observed to occur between runoff hydrographs.

A scatter-plot analysis was also undertaken for each of the volume-balance components and efficiencies, showing good agreement between the two models' outputs (Figure 3.24). Results for inflow, stored and total-infiltration volumes were practically identical, while small discrepancies appeared with the application efficiency, uniformity, and loss components.

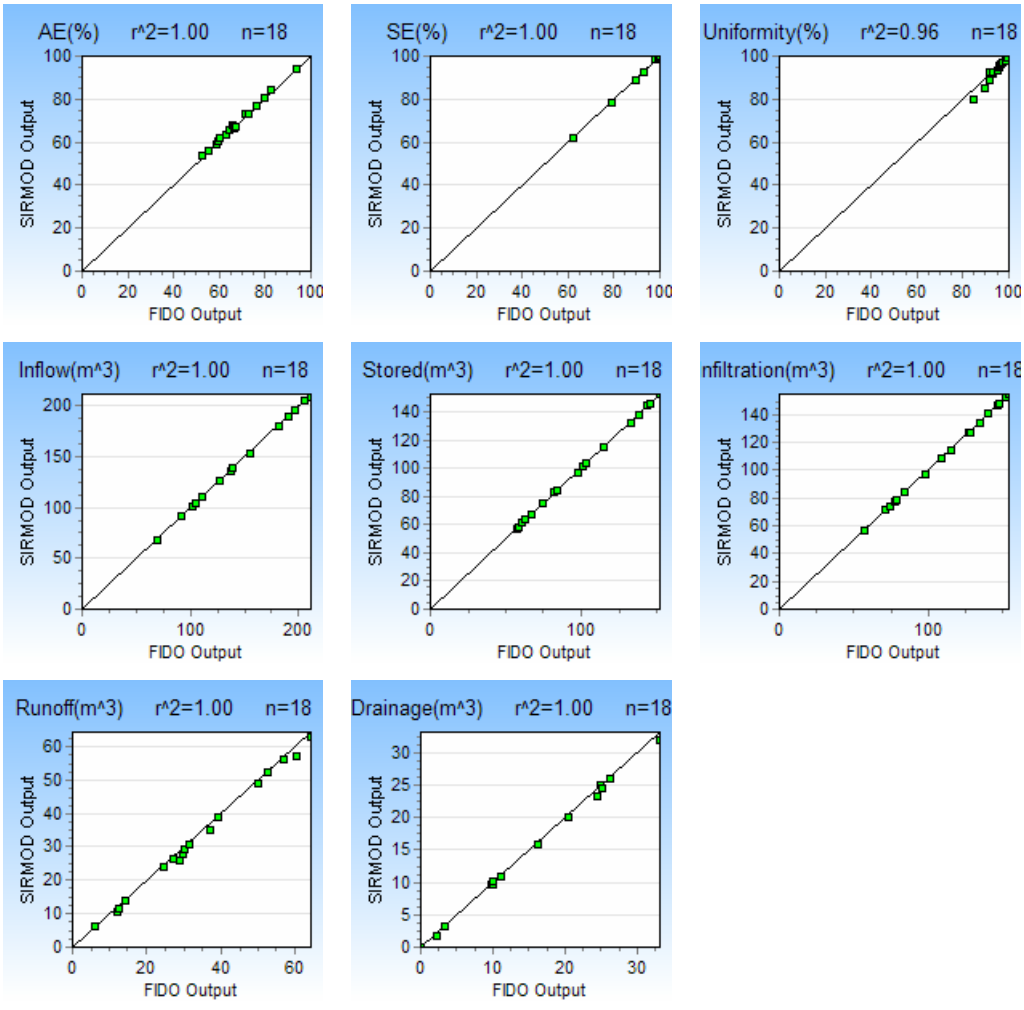


Figure 3.24: Scatter-plot analysis of *FIDO* vs *SIRMOD* outputs

Each model calculates uniformity differently, so this difference was not unexpected. The variations in the loss components (drainage and runoff) can be explained through slightly different treatments of simulated runoff between the models, and possibly by the stability measures in *FIDO* impacting on the volume-

balance results. For example, if instabilities are detected in the later stages of the irrigation, *FIDO* is able to switch to simulating only lateral flows, which is likely to have the greatest effect on the runoff and drainage results. As these are typically smaller components of the volume-balance, variations will appear more dramatic than for larger quantities such as inflow and stored volumes. It is likely that the stability measures will be downgraded in future versions of the software with further fine-tuning of the model.

3.8.2 Operation speed

The *FIDO* model has undergone several attempts over its development cycle at optimising the mathematical algorithms to minimise the overall simulation time. This is an important consideration given that the engine is to be used in calibration, optimisation and parameter analysis roles requiring potentially hundreds of repeated simulations. Running on the current generation Pentium 4 computers, simulation times typically range from a few hundredths of a second to up to three or four seconds using a ten-minute simulation time-step. The longer simulation times mostly occur when the simulation engine encounters difficulties and has to apply special treatment to individual time-steps to achieve convergence. Longer irrigation cutoff times also contribute to larger simulation times. The response-surface generation discussed in Chapters 5 and 6 provided total simulation times of approximately one hour to perform 10,000 simulations.

3.9 Conclusions

This chapter has described the development of a new object-oriented simulation engine capable of being implemented into a decision support system for furrow and border irrigation. This involved adaptation of the Walker and Strelkoff solution techniques, including the derivation of a simplified form of solution equations, and new algorithm for solving runoff. An object-oriented computer algorithm was developed for controlling the simulation, featuring intelligent input parameter objects. Robustness and reliability of the simulation engine was ensured by the development and implementation of four different treatments to avoid convergence problems. The simulation engine was validated against the *SIRMOD* model.

Chapter 4 Estimation of soil infiltration and hydraulic roughness parameters

4.1 Introduction

Accurately determining the spatial average value of the soil infiltration characteristic is a prerequisite for surface irrigation decision support operations. For example, evaluating irrigation performance is currently a two-stage process, whereby the infiltration parameters are typically estimated using a simple volume-balance model (such as the "two-point method") before being imported into a more complex hydrodynamic model for running the simulation. However, this is a potential source of error in the modelling process due to differences between the models' structures.

The goal of this chapter is to develop parameter estimation (calibration) capabilities for the *FIDO* decision support system that avoids this source of error by determining the soil infiltration and/or hydraulic roughness parameters using the same model that is used to perform the simulation. In practice, this will allow any of the three Kostiakov-Lewis infiltration parameters and/or the hydraulic roughness coefficient to be determined using a simple but powerful optimisation algorithm to minimise the error between the measured and predicted irrigation advance and/or runoff hydrograph.

The research in this chapter has five main objectives; *(1)* to present a background to developing calibration faculties for the decision support system; *(2)* to present as a preliminary study, the development a simple optimisation-based volume-balance inverse technique for determining the soil infiltration parameters; *(3)* using the optimisation-methodology developed in the previous task, to develop a more complex hydrodynamic inverse technique for the determination of any of the soil infiltration and/or the hydraulic roughness parameters; *(4)* to analyse the response-surfaces of the advance-based objective-functions; and *(5)* to validate the performance of the hydrodynamic inverse method using real field data.

This chapter is accompanied by a single appendix containing the output of the calibration validation showing simulated advance curves resulting from both the volume-balance and hydrodynamic inverse methods (Appendix 4.1).

4.2 Background to estimation of soil infiltration and roughness parameters

The inverse methods employ a form of simulation-model to determine infiltration parameters through matching field measurements with the simulated outputs. This alternative usage of the simulation model is essentially a calibration technique, yet no software packages are currently available which combine both simulation and calibration capabilities using the same model. Infiltration is

typically calculated using a simple volume-balance model with these results being imported later into the more complex hydrodynamic model. However, this may cause inaccurate results due to the differences in model structure and the empirical nature of the inverse technique.

Models used to solve the inverse problem consist of two parts. The first is an equation describing the process of infiltration, or entry of water into the soil. The modified Kostiakov-Lewis equation is the infiltration equation, which is most often used with furrow irrigation (Walker and Skogerboe 1987):

$$I = k\tau^a + f_o\tau \quad (4.1)$$

where I is the cumulative infiltration (m); τ is the time (min) that water is available for infiltration into the soil, otherwise known as the opportunity time; *a* (dimensionless) and k ($m^3/min^a/m$) are fitted parameters; and f_o ($m^3/min/m$) is the steady-state or basic infiltration rate for the soil. This equation is well suited to most soil types as it takes into consideration the basic infiltration rate. Failure to do this can lead to an underestimation of the cumulative intake (Hanson *et al*. 1993).

The second part of the inverse solution is a representation of the distribution of water temporarily stored on the surface of the furrow or bay. This links the infiltration equation to measurable parameters such as the inflow, surface water depth and the irrigation advance. This component of the solution usually takes the form of either a volume-balance model (consisting of the continuity equation only), or hydrodynamic advance model (consisting of a continuity equation and a momentum equation).

4.2.1 Objectives of calibration module development

The primary goal of the research in this chapter is to develop a calibration module for estimating soil infiltration (and roughness parameter) capable of being implemented into a decision support system for furrow and border irrigation. Specific objectives of the calibration module are:

- Unique infiltration and roughness parameters must be determined without user intervention;
- It must be capable of using any form objective-function (and hence measurement data) including those based upon advance and/or runoff data;
- It must be able to include any combination of soil and/or roughness parameters.

4.2.2 Elements of the calibration module

The calibration module required for solving the inverse problem to estimate soil infiltration and roughness parameters is composed of five principal components (Figure 4.1):

- calibration parameters;
- a simulation engine;
- field measurements;
- an objective-function; and
- an optimisation engine.

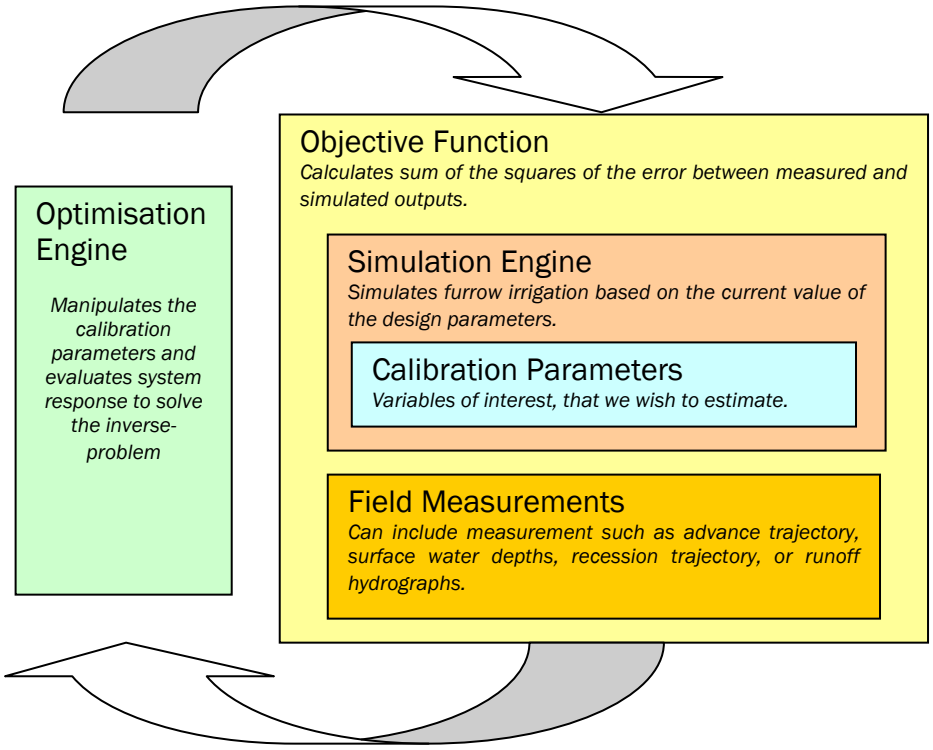


Figure 4.1: Fundamental components of the calibration module.

The *calibration parameters* represent the soil infiltration and/or hydraulic roughness parameters that we wish to estimate. The *simulation engine* is required to provide the simulated outputs to compare with the field measurements. The *field measurements* include data such as advance-trajectory, surface water depths, recession-trajectory and/or runoff hydrographs. The *objective-function* calculates the error between the field measurements and simulated outputs. The *optimisation engine* is the computer algorithm that manipulates the calibration parameters in order to minimise or maximise the objective-function response without user intervention.

The conceptual input/output functionality of the calibration module is displayed in Figure 4.2. This suggests that the calibration parameters such as a, k, f_o and/or *Manning* n are passed into the optimisation module. An objective-function is also required as input to the module. These input choices can be selected by the user through an appropriate graphical user interface. The calibration process is then performed by automatically curve-matching the measured and predicted advance and/or runoff data. The calibrated parameter values are then presented as output from the module, and returned to the decision support system.

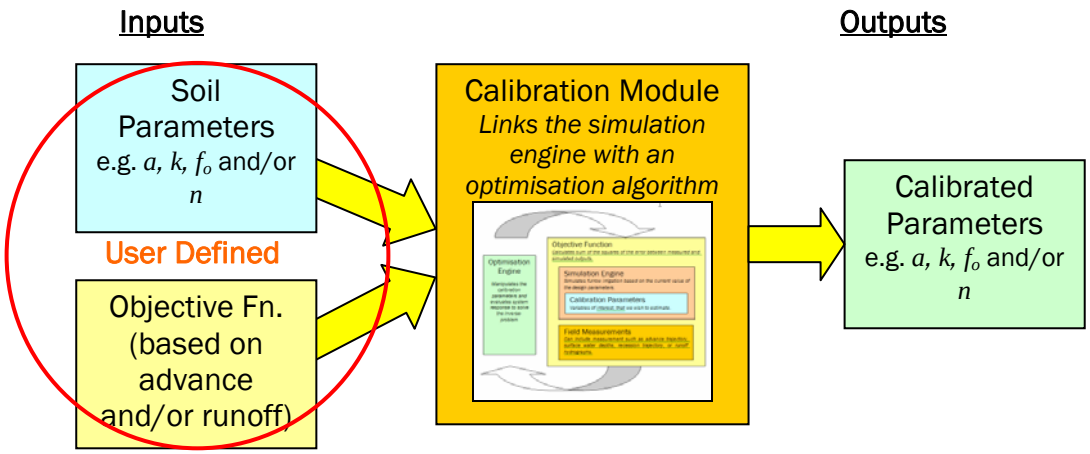


Figure 4.2: Conceptual input/output functionality of the Calibration module.

4.2.3 Limitations of existing techniques

Inverse techniques have proven the most popular and reliable methods of parameter estimation of soil infiltration and hydraulic roughness parameters. However, despite all of the methodologies available, there is currently no "best" method for determining these parameters. This is evidenced by recent work by the US Water Conservation Laboratory who is developing a new software tool for estimating infiltration and roughness parameters (Tamimi *et al*. 2003). The software has two goals: firstly, to provide a tool for evaluating different parameter-estimation methods in order to develop recommendations and guidelines; and *secondly*, to apply the most appropriate methodology for the data which is available.

The main problem with the existing methods is that they typically provide only a crude "calibration" of the target simulation model. Inaccuracies result from differences between the calibration and simulation model structures, ineffective solution techniques, and a deficit of reliable input data.

Despite recent attempts at combining optimisation algorithms with hydrodynamic and zero inertia models, the two-point method (Elliot and Walker 1982) has been the most commonly used technique to solve the inverse problem. This is partly due to the simplicity of the two-point method not requiring specialised software making it cheap and easily accessible. It is regularly taught in irrigation courses and is included in many irrigation texts. In comparison, the software-based optimisation techniques have not been made readily available to the public suggesting that problems outlined in the literature may be more serious than indicated. As well, the data requirements of the two-point method are minimal compared to many of the alternatives. Those techniques requiring the measurement of a surface depth profile are the least practical as this is a relatively complicated and expensive task.

Because only two points on the irrigation advance are used in the two-point method, a possibility for error exists if either of the points is not representative of the advance. Although only two points are required, the method still remains information expensive in that the basic infiltration rate and cross-sectional area

still need to be measured. Errors in the measurement of these quantities can lead to inaccuracies in the infiltration parameter values.

A limitation of all but the latest methods involving optimisation is that they require a lot of input data including advance measurement, surface depth profiles, inflow and outflow hydrographs. Temporal variations in the infiltration parameters are also important. The infiltration conditions of the soil tend to change between irrigation applications due to factors such as initial moisture content and degree of compaction. Therefore the infiltration parameters should be measured while the particular irrigation event is in progress. With existing optimisation methods there are typically three factors hindering this operation including the low speed of optimisation, the need for user intervention, and the need to measure f_o.

4.3 Preliminary study – *INFILT* volume-balance solution technique

As a preliminary study towards developing the calibration module for the decision support system in this dissertation, a simple volume-balance technique (*INFILT*) was developed to overcome many of the problems associated with the existing inverse techniques[7]. During the early stages of this research, it was not known whether structural differences between the simulation model (hydrodynamic model) and the inverse method (volume-balance model) would introduce significant errors in the simulation results. While the findings of this initial work were promising, it was subsequently found that these structural differences needed to be accounted for in the simulation model by adjusting the *Manning* n parameter (see Appendix 2.2). Nevertheless, this study was invaluable in developing the optimisation code used in the decision support system and for identifying the problems associated with the determination of the infiltration parameters.

The *INFILT* method couples a volume-balance model with an optimisation algorithm to minimise the error between the predicted and measured advance. The method differs from existing approaches in that only advance data and inflow rates are required. The average cross-sectional area of the furrow and the final infiltration rate are treated as fitted parameters and need not be measured.

The method improves on the two-point method using a model of similar form. It utilises the full irrigation advance (at least two advance points) while data requirements are reduced substantially by the omission of the need to measure the flow-area and final infiltration rate. This is possible through using an optimisation technique to find the values of the three infiltration parameters and the average cross-sectional area of flow.

By including the cross-sectional area of flow as one of the fitted parameters, it is treated empirically rather than as a physical parameter. Its resulting magnitude will then reflect the effect of spatial changes in the above-mentioned variables.

[7] This work was developed into a stand-alone software package called *INFILT* and published in the Journal of Irrigation Science (McClymont and Smith 1996).

The method may be suited to automation or real time control. A barrier to these processes is that the solution technique undertaken must provide the results quickly and without human interaction. Therefore a simple optimisation procedure is also required. However, the main purpose of the method is for application in conventional manually controlled surface irrigation, and the optimisation procedure described is well suited for either purpose.

4.3.1 Derivation of method

The proposed method follows from that of Smith (1993). It is based upon the volume-balance equation derived by Elliott and Walker (1982) for the two-point method, and is only applicable while there is no runoff from the end of the field. This volume-balance model is simply stated as:

$$Q_o t = V_I + V_S \quad (4.2)$$

where Q_o is the inflow (m³/min); t is the time (min) since commencement of the irrigation; V_I is the volume (m³) of water infiltrated; and V_S is the volume (m³) of water temporarily stored on the surface.

Eqn. 4.2 is modified by the substitution of $\tau = t - t_x$ to give Eqn. 4.3:

$$I_x = k(t - t_x)^a + f_o(t - t_x) \quad (4.3)$$

where I_x is the depth of infiltration (m) at a distance x (m) from the top of the field, and t_x is the time (min) for the advance to reach the distance x downstream. Eqn. 4.3 can be expressed in terms of x by assuming that the advance follows a power function relationship:

$$x = pt_x^r \quad (4.4)$$

where r and p are empirical parameters. Substituting this into Eqn. 4.3 and integrating over the wetted length of the field determines the total volume V_I of water infiltrated in time t:

$$V_I = \int_0^x I_x dx \quad (4.5)$$

The parameter p disappears in the integration. The volume of water V_S stored on the surface can be calculated from:

$$V_S = \sigma_y A_o x \quad (4.6)$$

where σ_y (dimensionless) is a surface storage shape parameter; and A_o is the average cross-sectional area of surface water at the upstream end of the furrow or bay.

Substitution of Eqn. 4.5 and Eqn. 4.6 into Eqn. 4.2 gives the volume-balance equation as used in the two-point method of Elliott and Walker (1982):

$$Q_o t = \sigma_y A_o x + \sigma_z k t^a x + \frac{f_o t x}{1 + r} \quad (4.7)$$

where σ_z (dimensionless) is subsurface storage shape parameter and is given by:

$$\sigma_z = \frac{a + r(1-a) + 1}{(1+a)(1+r)} \quad (4.8)$$

To solve the above equations for the infiltration parameters, an objective-function is formulated based upon minimising the sum of the squares of the error between the predicted and measured advance. Eqn. 4.7 is rewritten as;

$$x = \frac{Q_o t}{\sigma_y A_o + \sigma_z k t^a + \frac{f_o t}{1+r}} \quad (4.9)$$

The objective-function is then;

$$SSE = \sum_{i=1}^{N} \left[x_i - \frac{Q_o t_i}{\sigma_y A_o + \sigma_z k t_i^a + \frac{f_o t_i}{1+r}} \right]^2 = \text{a minimum} \quad (4.10)$$

where x_i the measured advance distance and t_i is the measured time at advance point i; and N is the total number of advance points.

To apply the method, a power curve regression first needs to be undertaken to calculate r from Eqn. 4.4. It is preferable that a non-linear regression technique be employed rather than the log-transformed linear regression as used in the two-point method. It was found that the simple log-transformation procedure placed too little emphasis on the later advance points. This could also be overcome using a weighted log-transformation.

A second non-linear optimisation (Eqn. 4.10) is then used to determine the four parameters a, k, f_o, and $\sigma_y A_o$.

4.3.2 Optimisation technique

In the early stages of this development, a computer program was written to test different optimisation algorithms. The Steepest Descent and Newton's methods of optimisation were initially employed to solve for the infiltration parameters. It was found that the Steepest Descent method provided reasonable results, although optimisation times were often long with thousands of iterations required. Newton's method failed to converge on the optimum solution when attempting to solve for four parameters.

A new procedure was therefore developed with the main goals of robustness and global convergence. This new method avoids the calculation of derivatives through a “common sense” approach of “forcing” the objective-function to decrease by changing the design parameters individually. This process undertakes several separate optimisations with only one design parameter changing in each. However, this process by itself is extremely slow. To increase the rate of convergence and to adopt the nature of a gradient search method, the routine follows up the individual changes with a “group” parameter change.

This is undertaken by changing all of the parameters together in the direction of the resultant vector obtained from the sum of the individual parameter vectors (Figure 4.3).

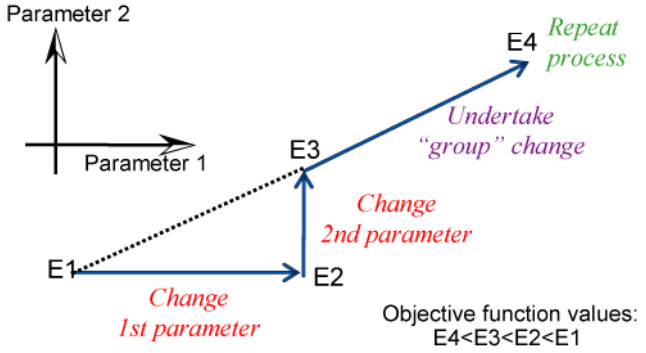


Figure 4.3: Step-cycle involved for two parameters

For example, for an objective-function with two design parameters, the steps in the minimisation process are;

(a) Select initial values for the two parameters.
(b) Change the first parameter until the objective-function can be lowered no further.
(c) Change the second parameter until the objective-function can be lowered no further.
(d) Change both parameters as a group in the direction of the resultant vector from Steps 2 and 3 until the objective-function can be lowered no further.
(e) Repeat (b) to (d) until the optimum design parameters have been found.

This method has the advantage over traditional methods in that it is mathematically simple and forces the objective-function to increase/decrease with the individual parameter changes.

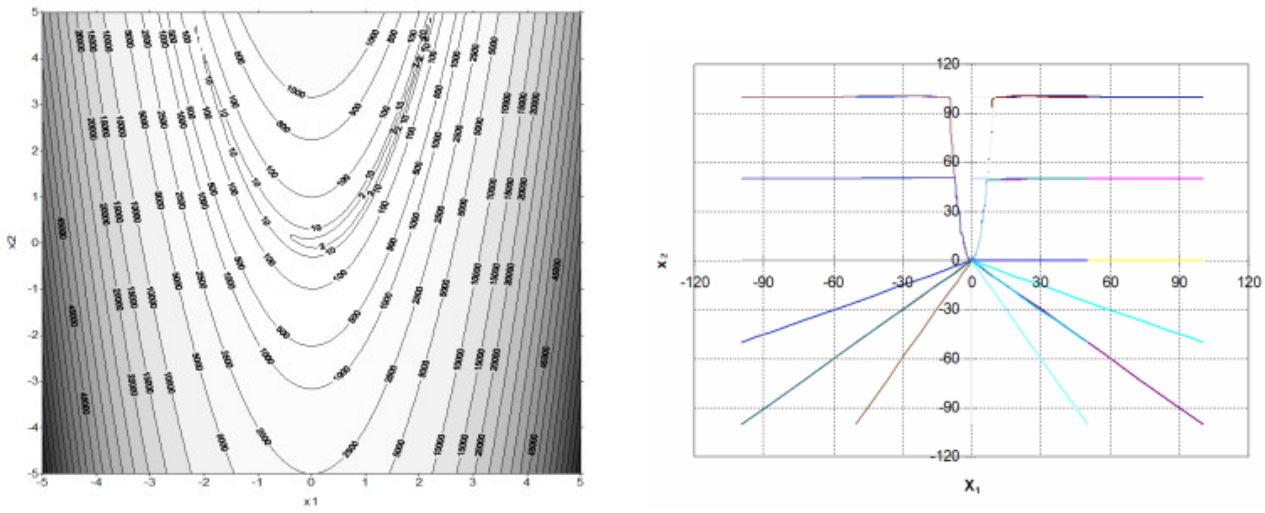
To test the optimisation method, it was first applied to a well-known test function for optimisation methods: Rosenbrock's function:

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \quad (4.11)$$

where x_1 and x_2 are the design parameters, and $f(x)$ is the function to be minimised. Figure 4.4(a) shows the response-surface for this function.

Twenty-five pairs of initial estimates were chosen to test the optimisation method with each test converging on the true optimum response value. Optimisation times were less than 4 seconds, running on an outdated Pentium 1 processor. In each case the final parameter estimates were the same at x_1=1.0 and x_2=1.0. Figure 4.4b demonstrates the paths taken by the algorithm for the different initial starting estimates. Straight lines on the graph indicate rapid convergence. The curved lines, corresponding to the positive initial estimates of x_2, appear to follow the response-surface contours (Figure 4.4a) and demonstrate the longest optimisation times.

Figure 4.4: (a) Response-surface of Rosenbrock's function; (b) Response trajectory of 25 sets of initial starting estimates.



(a) (b)

Having developed the optimisation algorithm, it was then applied to solve the inverse problem using the object-function presented in Eqn. 4.10. The steps followed in the process were:

Step 1: Initial values are selected for a, k, f_o, and $\sigma_y A_o$.

Step 2: Perturbation sizes are set for each parameter. Experience has shown the following to be suitable starting values:

$$\overline{P} = \begin{bmatrix} p_a \\ p_k \\ p_{f_o} \\ p_{\sigma_y A_o} \end{bmatrix} = \begin{bmatrix} 0.01 \\ 0.0001 \\ 0.00001 \\ 0.001 \end{bmatrix} \quad \text{.....(4.12)}$$

Step 3: Each parameter is then individually incremented or decremented by an integer number j of these perturbations, continuing while the objective-function (Eq.10) is decreasing, for example:

$$a_i = a_{i-1} \pm j_a \times p_a \quad \text{while} \quad SSE_i < SSE_{i-1} \quad \text{.....(4.13)}$$

Step 4: The parameters are then changed as a group by the same individual amounts as in Step 3, again until the objective-function can be lowered no further:

$$\overline{X}_i = \overline{X}_{i-1} + \overline{J}\,\overline{P}^T \quad \text{while} \quad SSE_i < SSE_{i-1} \quad \text{.....(4.14)}$$

where:

$$\overline{X}_i = \begin{bmatrix} a_i \\ k_i \\ f_{o\,i} \\ \sigma_y A_{o\ i} \end{bmatrix}, \quad \overline{J} = \begin{bmatrix} j_a \\ j_k \\ j_{f_o} \\ j_{\sigma_y A_o} \end{bmatrix} \qquad \text{.....(4.15)}$$

where the integers $\overline{J}$ are those as determined in Step 3. The effect of changing the parameters as a group is in effect a pseudo-gradient search method.

Step 5: Steps 2 to 4 are then repeated until the objective-function value can no longer be reduced by changing the parameters either individually or as a group. At this stage, the magnitude of the perturbation for each parameter is decreased and the process repeated. It is recommended that upper and lower limits (preferably 1 and 0 respectively) can be placed on each parameter during the optimisation so that unrealistic values are not obtained.

The optimisation process was found to have a wide range of convergence on the optimum solution. Tests undertaken, using both the upper and lower constraints as the initial estimates for the four fitted parameters, converged on the same parameter values as those employing more reasonable intermediate initial estimates. Optimisation times typically ranged from less than 30 seconds on a 486 computer to under 1 second on a new generation Pentium 4. All tests were undertaken without user input.

Another advantage of the method is that the user can easily set the values for any particular parameters. In this way, if the values of f_o and $\sigma_y A_o$ are known or can be assumed, the program can mimic the Two-Point method. Similarly parameters in the Phillip infiltration equation can be determined by setting *a* to 0.5.

4.3.3 Comparison with other methods.

Advance data and results (Walker and Busman 1990) for four irrigation events were used to evaluate the method. The infiltration parameter values obtained from the method are presented in Table 4.1 along with the measured values published by Walker and Busman and their results for the Simplex and two-point methods. To further facilitate a quantative comparison (in terms of *SSE*), values of $\sigma_y A_o$ for each of the published parameter sets were estimated through optimisation using Eq.4.10, since the authors did not publish the measured values. These “optimum” values and associated *SSE* are also presented in Table 4.1.

.

Table 4.1: Parameter and objective-function values for the proposed method compared to the results from Walker and Busman (1990) ($\sigma_y A_o$ and *SSE* values were calculated for each set).

	INFILT	Measured	Simplex	Two Point
Flowell Wheel				
a	0.444	.534	0.530	0.644
k ($m^3/min^a/m$)	0.00224	0.00280	0.00280	0.00150
f_o (m/min)	0.00033	0.00022	0.00022	0.00022
$\sigma_y A_o$ (m^2)	0.00667	0.00385	0.00385	0.00571
SSE	151.3	1089.78	1089.78	1543.5
Flowell Nonwheel				
a	0.788	0.673	0.681	0.698
k ($m^3/min^a/m$)	0.00190	0.00220	0.00260	0.00190
f_o (m/min)	0.00001	0.00022	0.00015	0.00022
$\sigma_y A_o$ (m^2)	0.00689	0.00698	0.00501	0.00796
SSE	138.4	412.4	287.7	433.3
Kimberly Wheel				
a	0.453	0.212	0.084	0.200
k ($m^3/min^a/m$)	0.00591	0.00880	0.0160	0.01030
f_o (m/min)	0	0.00017	0.00019	0.00017
$\sigma_y A_o$ (m^2)	0.00085	0.00554	0.00251	0.00318
SSE	71.0	1494.1	2866.4	1285.4
Kimberly Nonwheel				
a	0.625	0.533	0.514	0.504
k ($m^3/min^a/m$)	0.00578	0.00700	0.00850	0.00890
f_o (m/min)	0	0.00017	0.00017	0.00017
$\sigma_y A_o$ (m^2)	0.0035	0.002124	0	0
SSE	91.7	128.4	180.0	176.0

On first inspection, the results from the method seem to differ considerably from the others. However, evaluation of the method is best undertaken by substituting the derived parameter values into the target model, the modified Kostiakov-Lewis infiltration equation (Eqn. 4.1). Figure 4.5 shows the resulting cumulative infiltration curves

It can be seen from these curves that the results of the method compare favourably with those of Walker and Busman (1990). The only significant discrepancy occurs for the Kimberly wheel furrow (Figure 4.5c). Here the cumulative infiltration differs from the curves of Walker and Busman (1990) by about 10mm at the intermediate opportunity times. However, the *SSE* values for this trial as presented in Table 4.1 indicate that the published results fit the advance data relatively poorly.

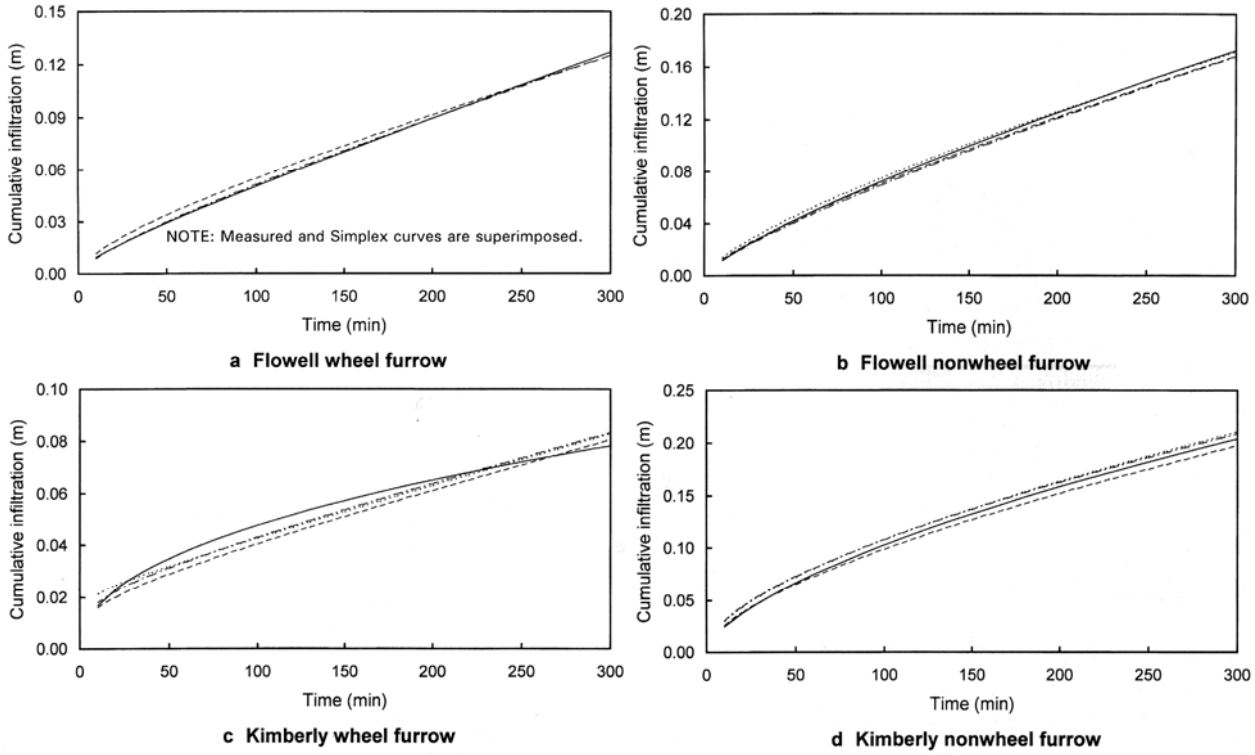


Figure 4.5: Cumulative infiltration curves for the Flowell wheel, Flowell nonwheel, Kimberly wheel, Kimberly nonwheel furrows, comparing the results of the method to that of Walker and Busman (1990) (_____ *INFILT*, _ _ _ _ measured, ... Simplex, _ . _ . _. _ Two Point).

4.3.4 Volume-balance errors

Volume-balance errors (VBE) were calculated (Eqn. 4.16) at different advance points for the Flowell nonwheel furrow (Figure 4.6).

$$\text{VBE} = \frac{\text{inflow vol.- (surface storage + infiltrated vol.)}}{\text{inflow vol.}} \times 100\% \quad \text{.....(4.16)}$$

A high initial volume-balance error of 20% is indicative of the relatively low volume of water applied at that point. The remaining errors are relatively low and support the model's validity.

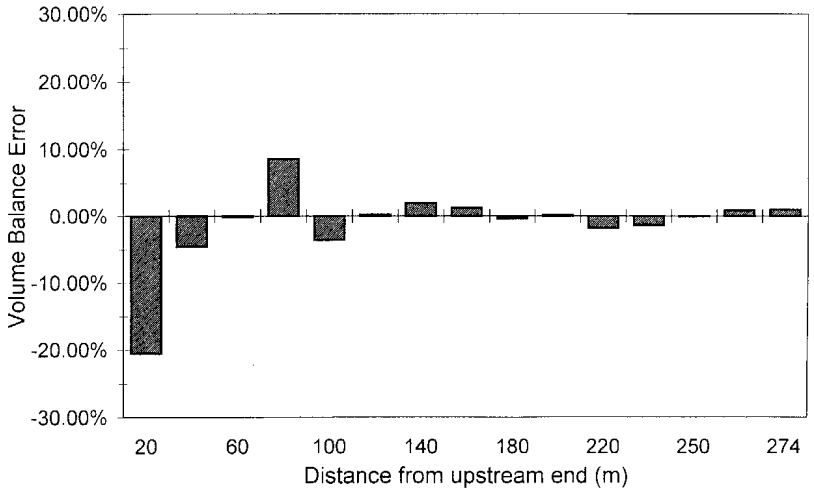


Figure 4.6: Volume-balance errors for the Flowell nonwheel furrow of Walker and Busman (1990)

4.3.5 Objective-function response-surfaces

The best way of evaluating the performance of an optimisation-based solution technique is to generate a response-surface for the design parameters, in order to visually validate the optimised results. This is simple when there are only two design-parameters, which can be evaluated in the form of a single three-dimensional surface, or as a contour-plot in two dimensions. However, the situation becomes complicated when there are more than two design-parameters, leading to a solution space greater than that which we can physically visualise. In this case-study example, we have four design-parameters (a, k, f_o and $\sigma_y A_o$) leading to a solution space in five dimensions.

To overcome this problem, a range of response-surfaces was plotted for combinations of the various parameters (Figure 4.7). The two parameters not shown on each surface were held constant at their optimum values. To display the full range of possible *SSE* values over the selected parameter range, the logarithm of the error was plotted on the vertical axis.

Figure 4.7a shows a strong parabolic relationship between the a and k parameters with a well-defined minimum. However, the a-f_o and k-f_o surfaces in Figure 4.7b and Figure 4.7f show long valleys corresponding to relatively fixed values of a and k respectively. This would suggest that the objective-function is less sensitive to changes in the f_o parameter.

An obvious potential limitation of the method is that the $\sigma_y A_o$ parameter, which represents the amount of water stored on the surface, is treated empirically. If the value is not correct in a physical sense, then the predicted infiltration may be affected. However, it can be seen from the surfaces showing the influence of the $\sigma_y A_o$ parameter (Figure 4.7c to e) that there is a true minimum on which the optimisation process can focus. This characteristic provides some reassurance that including the $\sigma_y A_o$ parameter in the optimisation is a viable option, and that the values obtained should be realistic.

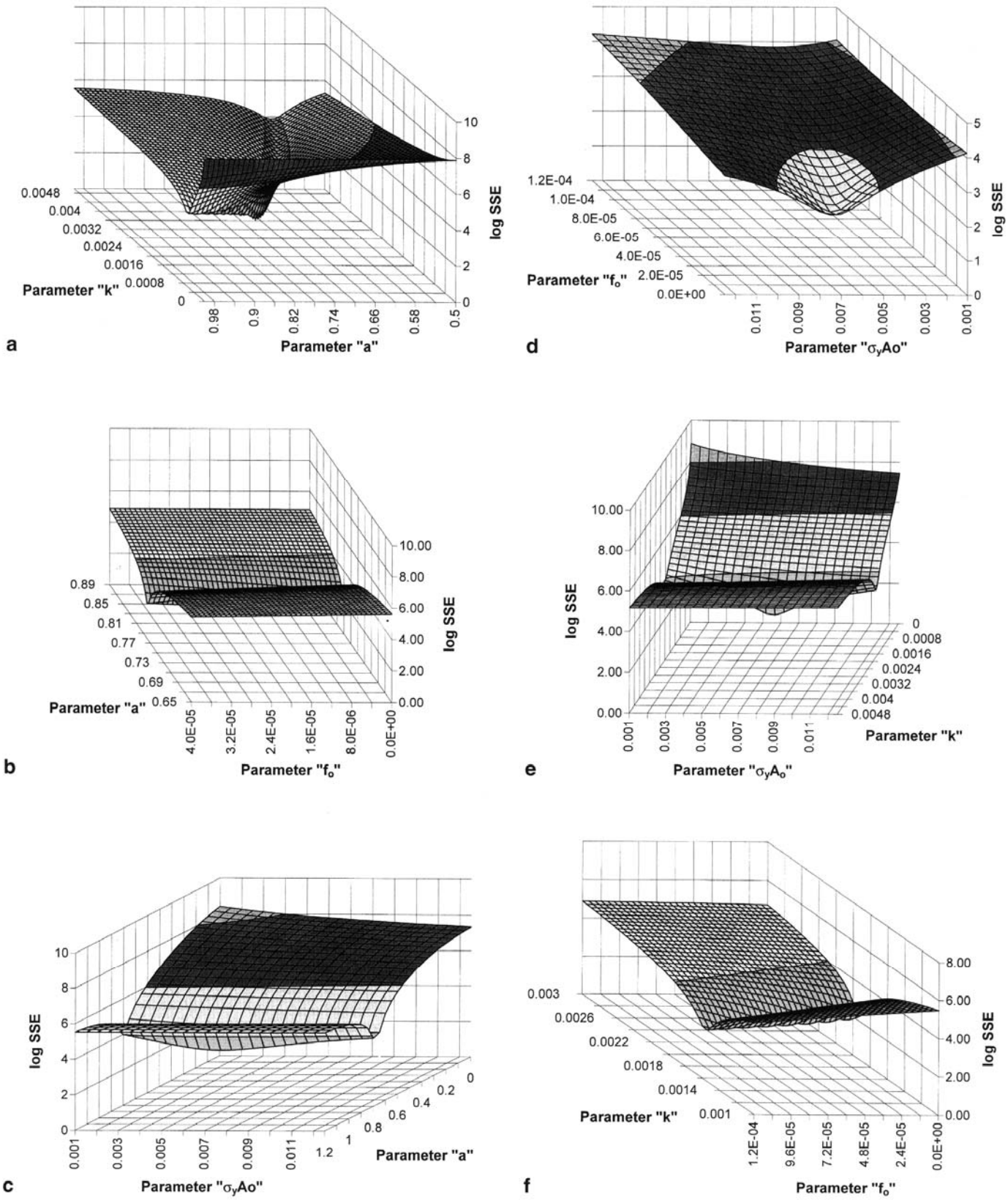


Figure 4.7: Objective-function response-surfaces for the Flowell nonwheel furrow of Walker and Busman (1990).

By treating $\sigma_y A_o$ as an empirical parameter, the method accounts for variations in geometry (slope, cross section, roughness) along the length of the furrow better than is possible by the use of an assumed σ_y with a measured Ao.

The treatment of f_o as an empirical parameter is only likely to become a problem with high infiltration soils involving short advance times. Under these circumstances, the fitted value may tend to zero and be much lower than the true physical value. This difficulty arises due to the inability of the model to differentiate between the transient and steady state components of infiltration at short advance times. This observation is in agreement with Bautista and Wallender (1993b) who found that the reliability of the parameter estimates increased for relatively long advance times.

4.3.6 Data handling

Two facets of the proposed technique that warrant consideration, and that are universal to all similar calculation methods, are the use of unconditioned data, and the sensitivity of the model to measurement errors.

The temptation to condition the data arises from a desire to make the optimisation process as efficient as possible; this efficiency being measured in terms of the speed and reliability of convergence, and in the accuracy of the results. Possible data conditioning strategies include;

- the use of an optimum number of advance points;
- the deletion of early advance data points; and
- the use of smoothed advance data.

It is reasonable to assume that the optimum number of advance measurements required for the solution technique will vary according to the quality of advance data. As a general rule, the use of a small number of advance points will lead to fast convergence on a solution. A greater number of advance points are needed to maintain accuracy in the presence of noisy or imperfect data. However, an excessive number of advance points may complicate the optimisation process so that the true optimum may not be found, hence reducing the accuracy of the solution.

Deletion of the early time advance data was investigated. It was found that the optimisation time was greatly reduced without significantly altering the results, indicating that less importance can be placed upon early advance data. This is in general agreement with the work of DeTar (1989) who suggested that data from the first quarter of the field could be neglected.

The *Flowell nonwheel* advance data from Walker and Busman (1990) were also used to determine the effect on the solution of smoothing the advance data. The power advance curve determined in the initial regression was used to generate pairs of advance data (x and t) for the subsequent optimisation process. Convergence on the solution was improved using the smoothed data. However, the cumulative infiltration curves (Figure 4.8) generated from the measured and smoothed advance data differ considerably over the full application duration.

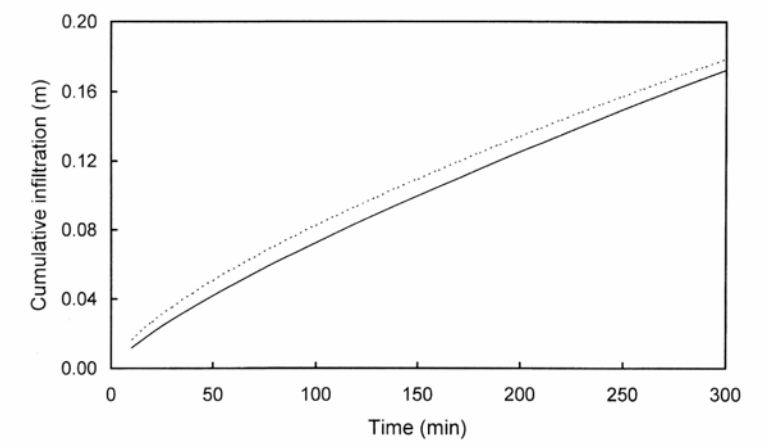


Figure 4.8: Cumulative infiltration curves for original and smoothed Flowell nonwheel advance data of Walker and Busman (1990) (_____measured; . . . . . .smoothed)

This difference is not an argument against smoothing but an indication of the limitations of the power curve to represent the advance and warrants further investigation. That the proposed method can handle a degree of noise without conditioning of the data is adequate demonstration of its utility. However, the reliability of the solution will still depend on the quality of the input data.

The only input data required by the method are the inflow rate and the advance. It is the measurement of the inflow rate that is most likely to be a source of error in determining the parameters. Errors in the measurement of the advance are not cumulative, and the model is designed to handle such noisy data.

To show the effects of inflow measurement errors, the inflow for the *Flowell nonwheel* furrow (Walker and Busman, 1990) was varied by $\pm 10\%$. The resulting effect on the parameters can be seen in Table 4.2. In all three cases, the proportion of water on the surface remains the same at a given point in time (t=432min). Therefore increasing the inflow rate effectively increases the volume of water predicted on the surface and vice versa.

Table 4.2: Effect of flowrate variations on the resulting parameter values for Flowell nonwheel furrow data from Walker and Busman (1990). The measured inflow is varied by ±10%.

% of measured inflow	90%	100%	110%
Inflow (m^3/min)	0.108	0.12	0.132
a	0.790	0.791	0.791
k ($m^3/min^a/m$)	0.00171	0.00190	0.00208
f_o (m/min)	0	0	0
$\sigma_y A_o$	0.00621	0.00692	0.00762
SSE (m^2)	137.47	137.49	137.43
surface water as % of inflow at t=432min	3.65%	3.66%	3.66%

Increasing the flowrate also increases the cumulative infiltration (Figure 4.9). The resulting infiltration curves differ by nearly 20 mm after 300 minutes; far greater than that seen in comparing the different calculation methods. We can also infer that the error induced in the parameters by incorrect inflow measurements is likely to be greater than that caused by any inability of the method to exactly model the volume-balance.

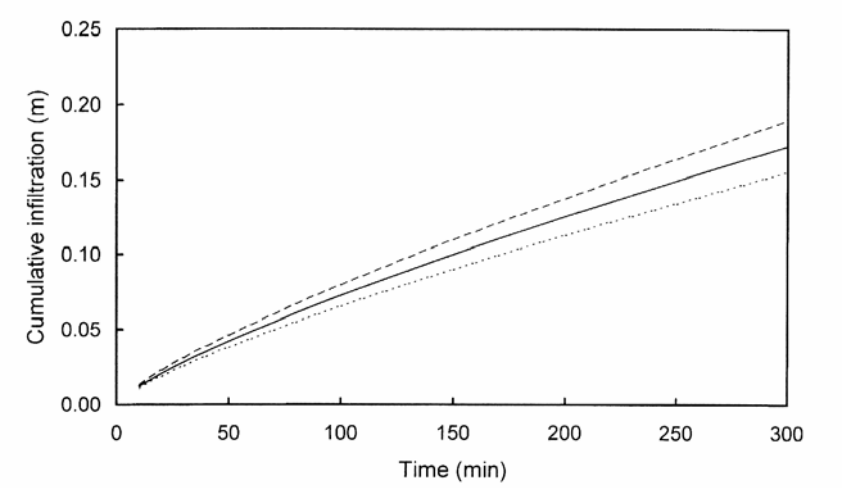


Figure 4.9: Effect of flow measurement errors on cumulative infiltration of Flowell nonwheel furrow of Walker and Busman (1990) at t=432 min (. . . . . -10%real flow; _____ real flow; _ . _ . _ +10% real flow)

4.3.7 Findings of the preliminary study

A volume-balance method was developed for the estimation of the infiltration characteristics of a soil from surface irrigation advance data. Important features of the method are that it:

- has an optimisation based on minimising the difference between predicted and measured advance curves which requires no manual intervention;
- includes the final infiltration rate f_o and the average cross-sectional area of flow $\sigma_y A_o$ as parameters evaluated in the optimisation;
- is able to handle noisy advance data effectively without the need to condition data before use; and
- requires a minimum of field data, but accurate inflow data.

Initial testing of the model indicates that it is a useful tool for determining the soil infiltration characteristic. Since it was developed, hundreds of copies of the *INFILT* software package have been downloaded over the Internet, and it has been used in university (http://www.usq.edu.au) and training courses (http://www.ncea.org.au), and also in practice for the last ten years. However, it must be recognised that since it is structurally different to the target hydrodynamic simulation model, the potential for error exists when using the results for any decision support operation. Nevertheless, the cumulative infiltration curves generated from the method compared well with the measured

values, and those from the Simplex and two-point methods. Spatial variations in the geometric and hydraulic properties of the furrow should be accounted for and reflected in the infiltration parameter values through the inclusion of $\sigma_y A_o$ as a derived parameter.

The simple optimisation algorithm developed for use in the method was found to be more reliable than the Newton and Steepest Descent methods. The speed, accuracy, and wide range of convergence of this algorithm may make the model suitable for use in real time control of furrow and border irrigation. It was decided that it is also suitable for inclusion in the hydrodynamic inverse solution developed in the next section.

4.4 *FIDO hydrodynamic inverse technique*

The optimisation methodology used in deriving the *INFILT* technique can be extended to determine the infiltration and hydraulic roughness coefficients using the full form of the hydrodynamic model. This overcomes the common problem of calibrating using one model (volume-balance model), and simulating with another model (hydrodynamic model). With this new methodology, the same model is used for both simulation and calibration.

In this instance, no time-of-advance equation is available to generate the advance profile. Complete simulations need to be performed, and the simulated advance (and/or runoff) trajectory extracted and compared with the measured data. Because all of the irrigation phases can be generated during the simulation, a range of objective-functions could be used in the calibration based upon the advance, recession, surface storage, and runoff, or combinations of each.

4.4.1 Algorithm design considerations

Using complete simulations during the calibration process introduces many new algorithm design considerations when compared to using a simple time-of-advance equation. While there are many benefits from using complete simulations during the optimisation, the solution process is generally much more complex, and many times slower than time-of-advance techniques.

One problem is that simulated outputs typically contain a degree of noise related to the discretised-solution process, which could impede the optimisation algorithm from finding the global minima. This manifests itself in the form of a rough uneven response-surface with bumps and pits which researchers commonly (and often wrongly) refer to as local minima.

Another problem can exist where the spatial or temporal locations of the measured data do not coincide with the simulation outputs. Interpolation algorithms such as cubic-splines must then be used to match pairs of data. This is more complicated when the simulated advance doesn't reach the measured node locations, as the response-surface must be generated using an equal number of measurements stations for each iteration of the calibration. A

negative advance (front end recession) can also occur during simulations that must be taken into consideration.

However, one of the main benefits of using the more complex approach is for the common case when the inflow is cutoff before the advance reaches the end of the field. Simultaneous advance and recession is likely to occur which cannot be adequately treated using the time-of-advance methods.

4.4.2 Derivation of *FIDO* hydrodynamic inverse method

The procedure for solving the inverse technique using the hydrodynamic-model is the same as that defined in Steps 1-5 in Section 4.3.2, except that the objective-function is now different and results from complete or near-complete simulations of the irrigation. Cubic-spline interpolation is used to match pairs of measured and predicted data, while penalty functions are introduced to ensure consistent data counts when comparing measured and predicted data.

The objective-function is determined by the availability of input data and is defined as the sum of the square of the errors between the measured and predicted data. The model can be calibrated using advance data, runoff hydrograph data, or a combination of both[8]. If runoff data are not available, then only the advance phase(s) of the simulation will be used in the calibration. If runoff data exists, then all phases of the simulation are included.

Any of the three parameters of the modified Kostiakov-Lewis infiltration equation can be included in the optimisation along with the *Manning n* roughness coefficient. The flexibility of the optimisation algorithm allows easy selection/deselection of calibration parameters.

4.4.3 Developing an object-oriented structure

The *FIDO* calibration component has been developed using an object-oriented structure in the C++ language. Some of the benefits of using an object-oriented design were discussed in Chapter 3. In this example, reusability of the objects is an important design consideration since the simulation engine, optimisation engine and objective-function objects will be reused by other components of the decision support system.

Figure 4.10 shows an overview of the object-oriented components of the calibration module including:

- a calibration manager, for overseeing the operation of the calibration process, and providing all input/output functionality of the system;
- an objective-function module, for storing different objective-function objects including those based upon the advance, runoff, and combined advance and runoff;

[8] The surface profile and/or complete recession trajectory could also be used but are less suitable to incorporate into objective functions since they are more difficult, and time-consuming to measure in the field. Implicitly, by including runoff as a component of the calibration, the final advance and recession times are automatically accounted for. Therefore, when runoff data is used, it could be argued that the calibration is also being performed on the recession.

- a simulation engine, for determining the simulated output;
- simulation parameter objects, which are the data storage containers for all of the simulation data including the measured data (advance, runoff etc) and calibration parameters; and
- an optimisation engine, which controls the optimisation process. This contains special parameter objects for linking to the calibration parameters and a pointer to the selected objective-function.

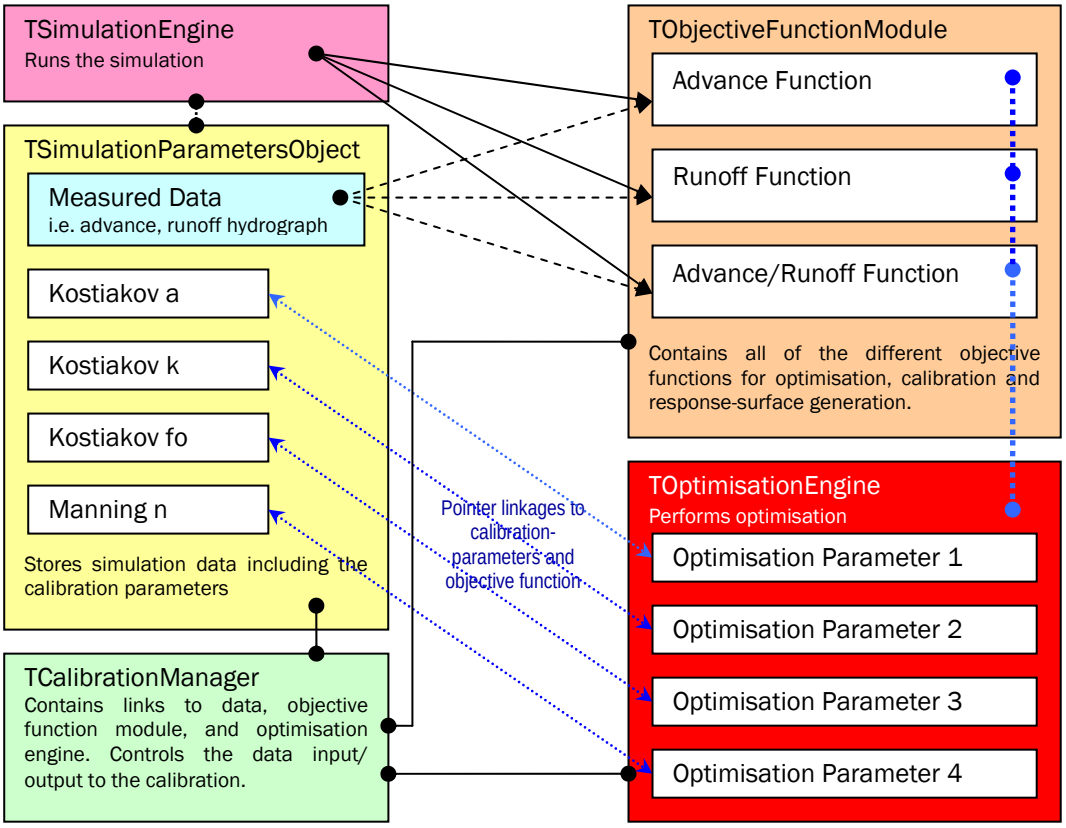


Figure 4.10: Object-oriented components for calibration module.

The optimisation parameter objects (called `TOptimisationParameter`) contained in the optimisation engine, are specially design "intelligent" objects that have multiple roles. They contain much of the functionality for linking the optimisation engine to the calibration parameters, as well as storage facilities for tracking parameter changes during the optimisation. They also contain many methods for interacting with the optimisation.

4.4.4 Calibration module algorithm

Figure 4.11 outlines the model algorithm used by the calibration module when estimating the soil infiltration and/or hydraulic roughness parameters. The algorithm revolves around one main loop for iterating through a range of data-files. This allows many calibrations to be performed in one go, simplifying the task for the user.

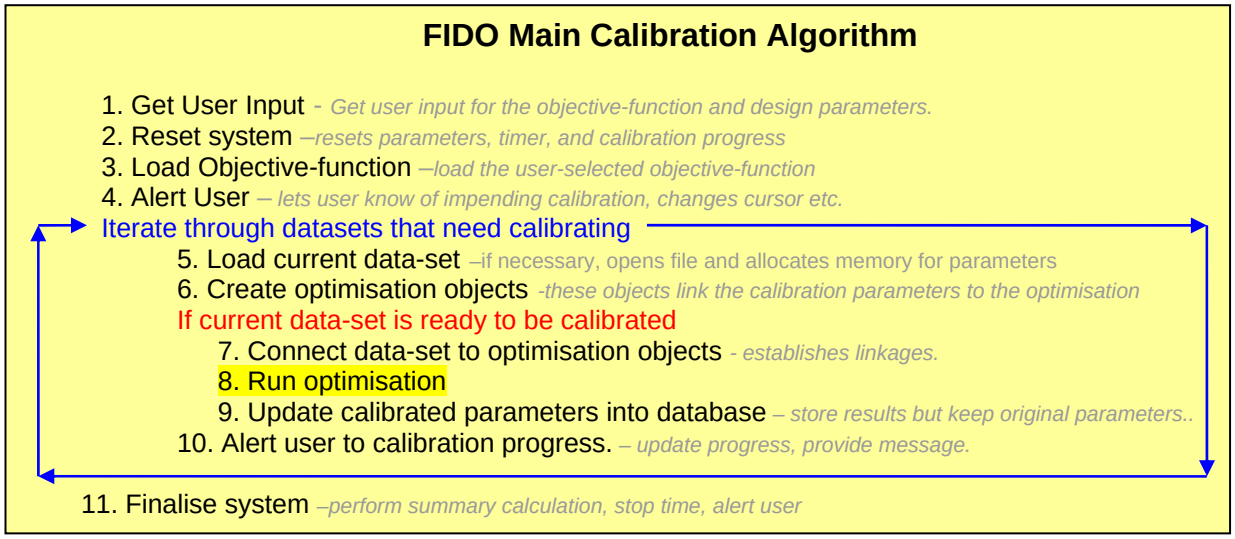


Figure 4.11: Calibration module algorithm

4.4.5 Objective-function algorithms

Advance-based objective-function

An objective-function based upon advance measurements aims to minimise the sum of the squares of the error between the measured and predicted advance times for different measurement stations. Figure 4.12 shows the algorithm for this function. When the function is called, the simulation engine and the objective-function error value are initially reset. The simulation is then performed and checked for successful completion. In the unlikely event that the simulation fails, a default penalty error is assigned to the function response.

The simulation solution nodes are not expected to coincide with the advance measurement stations. Therefore, the simulated advance trajectory is loaded into a special array for generating cubic-splines, in order to match up the spatial locations of the measured and predicted advance points for the objective-function calculations.

In the case where infiltration is very large, and the simulated advance does not reach some of the measured advance stations, the last predicted advance point can be repeatedly used as the reference point for calculating the error when iterating through the remaining advance stations. In effect, this introduces a significant time penalty to the function while the time-gap increases between the later advance measurements and the final simulated advance location.

Advance-based Objective Function Algorithm

```
Set SSEAdvance = 0.
Reset Simulation
Load New Parameters
Run Simulation
IF( Simulation was successful)
{
    Generate Advance Curve Fit (Cubic Spline)
    For( x=0 to number of measured data)
    {
        If (Front end recession has not occurred)
        {
            if(Last simulated Advance point<Last measured Advance Point)
            {
                SSEAdvance = SSEAdvance + ( MeasuredTime(x) – InterpolatedTime(x) )^2
            }
            else
            {
                SSEAdvance = SSEAdvance + ( MeasuredTime(x) – InterpolatedTime(last) )^2
                This penalises the response values since the change in time will become very large.
            }
        }
    }
}
Else
{
    SSEAdvance = NULL   This leaves a hole in the response-surface which the optimisation penalises.
}
Return the result to the optimisation algorithm.
```

Figure 4.12: Objective-function algorithm for advance data.

Runoff-based objective-function

Objective-functions based upon runoff hydrograph measurements can be derived in two different ways: *firstly*, by minimising the error between the measured and predicted runoff rates at each time-step; and *secondly* by minimising the error between the measured and predicted runoff volumes for each time-step. It was decided to use the second approach for the decision support system, since this is easier to calculate, and is less likely to be influenced by noise and discretisation errors.

Figure 4.13 shows the algorithm for the suggested runoff-based objective-function. As in the previous example when the function is called, the simulation engine and the objective-function error value are initially reset before running the simulation and checking that it performed correctly. If successful, the simulated runoff hydrograph is loaded into a special array for performing cubic spline interpolations before calculating the error value.

Runoff-based Objective Function Algorithm

```
Set SSERunoff = 0.
Reset Simulation
Load New Parameters
Run Simulation
IF( Simulation was successful)
{
    Generate Runoff Curve Fit (Cubic Spline)
    For( t=0 to number of measured data)
    {
        SSERunoff = SSERunoff + ( MeasuredVolume(t) – InterpolatedVolume(t) )^2
        Note: if the last simulated time is less than the last measured time, then the
        interpolated volume will be 0 for that time, and no penalty function needs
        to be introduced.
    }
}
Else
{
    SSERunoff = NULL   This leaves a hole in the response-surface which the optimisation penalises.
}
Return the result to the optimisation algorithm.
```

Figure 4.13: Objective-function algorithm for runoff data.

Combined "advance and runoff"-based objective-function

The objective-function based upon both advance and runoff data is more complex than in the previous examples because it is composed of both time-based and volume-based quantities. Therefore, user-defined weighting coefficients are introduced to equalise the relative magnitudes of the two error quantities.

Figure 4.14 presents the algorithm for this function. The procedure follows that of the previous objective-functions where advance and runoff errors are calculated as before. The resulting error portions are then multiplied by the weighting coefficients before being added together.

Advance and Runoff-based Objective Function Algorithm

```
Set SSERunoff = 0
Set SSEAdvance = 0
Set SSETotal = 0
Reset Simulation
Load New Parameters
Run Simulation
IF( Simulation was successful)
{
    Generate Advance Curve Fit (Cubic Spline)
    Generate Runoff Curve Fit (Cubic Spline)
    For( x=0 to number of measured data)
    {
        If (Front end recession has not occurred)
        {
            if(Last simulated Advance point<Last measured Advance Point)
            {
                SSEAdvance = SSEAdvance + ( MeasuredTime(x) – InterpolatedTime(x) )^2
            }
            else
            {
                SSEAdvance = SSEAdvance + ( MeasuredTime(x) – InterpolatedTime(last) )^2
                This penalises the response values since the change in time will become very large.
            }
        }
    }
    For( t=0 to number of measured data)
    {
        SSERunoff = SSERunoff + ( MeasuredVolume(t) – InterpolatedVolume(t) )^2
        Note: if the last simulated time is less than the last measured time, then the
        interpolated volume will be 0 for that time, and no penalty function needs
        to be introduced.
    }
    SSETotal= w1 x SSEAdvance + w2 x SSERunoff
    Where w1 and w2 are user-defined weighting coefficients.
}
Else
{
    SSETotal = NULL   This leaves a hole in the response-surface which the optimisation penalises.
}
Return the result to the optimisation algorithm.
```

Figure 4.14: Objective-function algorithm for combination advance and runoff data.

4.4.6 Achieving operational efficiency

One of the problems with using an objective-function based upon complete simulations of the hydrodynamic model is that convergence on the optimal solution is slowed down due to the time it takes to undertake individual optimisation iterations. Optimisations are typically an order of magnitude slower using the hydrodynamic method than compared to using objective-functions based upon time-of-advance equations. In early testing of this research, it was immediately apparent that this could limit the functionality of the decision support operations with some calibrations taking over ten minutes to complete (with an average time of approximate two minutes) using a modern Pentium 4 processor. Processing times were highly dependent on the quality of the initial parameter estimates for the optimisation.

Another problem is that despite all of the robustness measures encapsulated in the simulation engine, it could not always handle the extreme conditions that it was sometimes asked to simulate. For example, unrealistic parameter combinations are often encountered during the optimisation process

representing extremely high infiltration conditions. In this situation, the irrigation advance may only travel a few metres down the furrow, with several metres of infiltration occurring. While a simple time-of advance equation can adequately find a solution to this, it is outside of the operational range of the hydrodynamic model as configured for the decision support system[9].

Therefore, to improve the operational efficiency of the calibration module, it was decided to combine both of the methods developed in this chapter. That is, the *INFILT* method was incorporated into the calibration module to provide "good" starting parameter estimates for the optimisation involving the hydrodynamic-based objective-function. As well as speeding up the optimisation, this avoids the likelihood of the hydrodynamic-based objective function having to simulate any extreme infiltration conditions.

The two-stage methodology is characterised by a spike in the objective-function output during the optimisation when the objective functions are switched; that is, when the volume-balance calibrated parameters are inputted into the hydrodynamic-based objective function. This is an indication of the error associated with performing a calibration using a method with a different model structure to the target simulation model. Figure 4.15 demonstrates this effect by showing the output[10] of a typical advance-based calibration, with an annotation showing the transition (spike) between objective-functions.

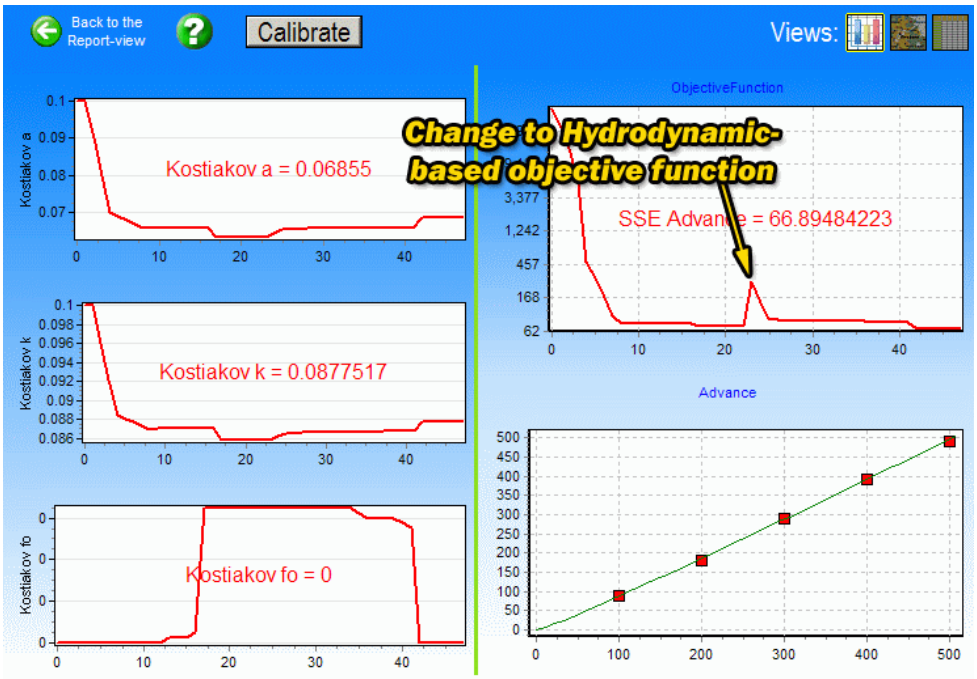


Figure 4.15: Advanced calibration output showing parameter and objective-function variations during optimisation. Arrow indicates the transition from the *INFILT* method to the hydrodynamic-based objective function.

[9] With extra refinement, it could be configured to handle this situation.

[10] This is an advanced-user analysis in *FIDO*, and is not normally displayed as part of the calibration process.

4.4.7 Response-surfaces

The surfaces generated during the initial case study in this chapter (Figure 4.7) presents one way of investigating the system response and optimisation accuracy by plotting a response-surface for each combination of the design-parameters. However, the true nature of the global minima remains unclear in these outputs, since many different reference coordinate systems are used when switching parameter-axis relationships. One way to overcome this is to generate a series of response-surfaces for the two most sensitive parameters (a and k), with separate surfaces presented for different combinations of the remaining design parameter (f_o).

Figure 4.16 presents an example of this analysis for the hydrodynamic form of the advance-based objective-function[11]. To improve visualisation of the outputs, the objective-function values are represented as the "log" of the sum of the squares of the error between the measured and predicted advance. Each surface represents a different value of Kostiakov f_o, and is generated by systematically varying the Kostiakov a and k parameter values.

These results prove that there is a true global minimum for the optimisation process to focus. It lies within a sharp deep parabolic-shaped value valley, which flattens and increases in magnitude for non-optimum values of f_o. The surfaces appear to be "smooth", but closer inspection reveals some surface roughness, which is not expected to impede the optimisation process.

4.5 Validation

Validation of the hydrodynamic inverse technique with the advance-based objective-function was performed within the $FIDO$ decision support system by running calibrations for real field data (thirteen irrigations), and investigating the simulated advance-curves based upon the calibrated modified-Kostiakov infiltration parameters[12]. The results of this validation are presented in Appendix 4.1 with a sample output presented in Figure 4.17. For comparison, and to show the error inherent with mixing model structures, the simulated output for the $INFILT$-calibrated infiltration parameters are shown in red on the charts.

These results show perfect agreement between the measured and simulated outputs based upon the hydrodynamic method, while the $INFILT$-based results often show significant deviation from the measured advance. In earlier research with the $INFILT$ method, this was accounted for by performing a second manual calibration by adjusting the $Manning\ n$ parameter when the calibrated infiltration parameters were entered into the simulation model (see Appendix 2.2). The new method alleviates the need for this, both simplifying the calibration process and providing a more accurate result though computer based-optimisation.

[11] Results for the other objective-functions have not been included in this dissertation, and will be followed up with future studies.

[12] For this study, the $Manning\ n$ parameter was not included in the calibrations. Also, the recession-based objective-functions were not evaluated and will be followed up with later studies.

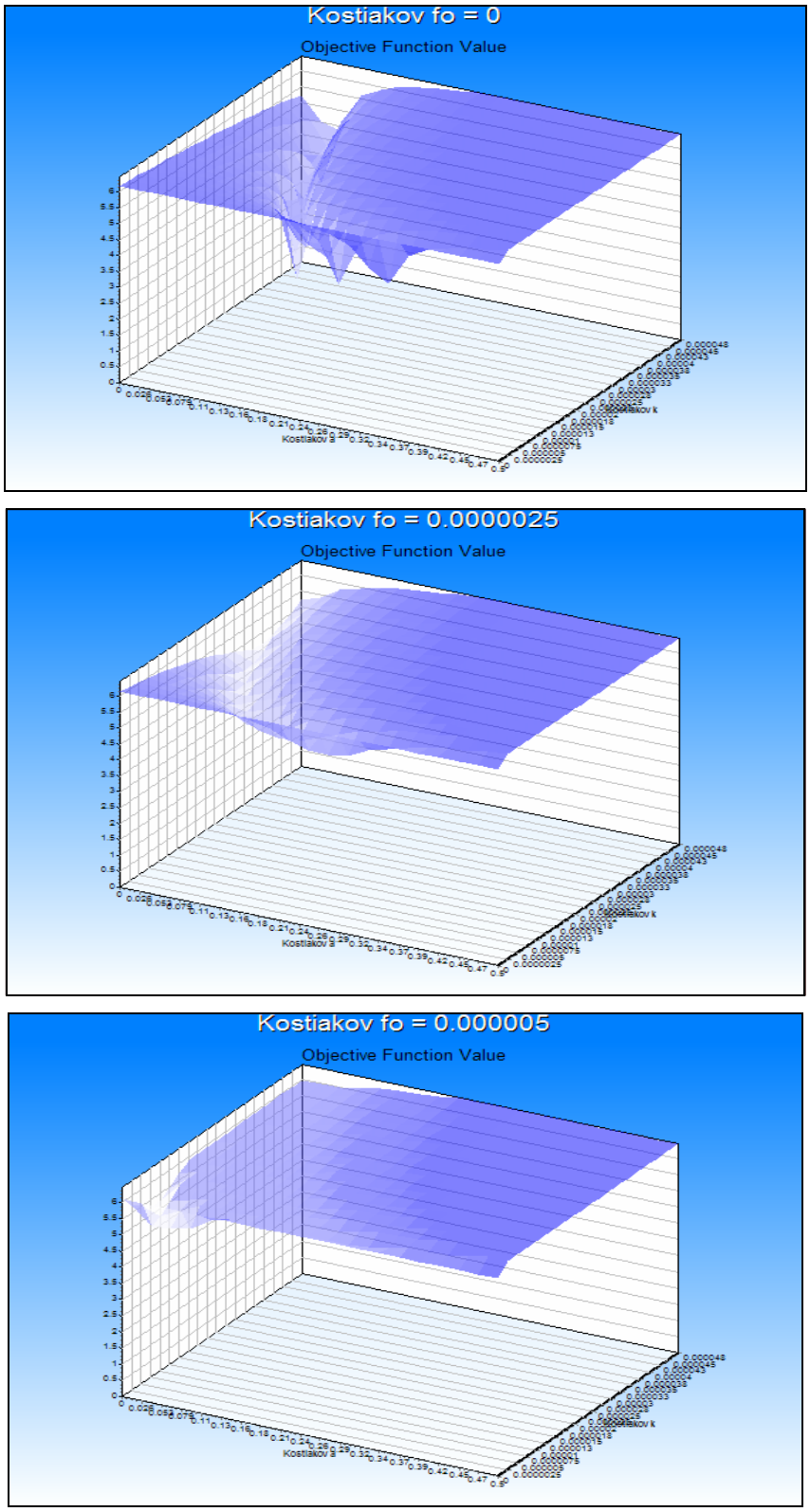
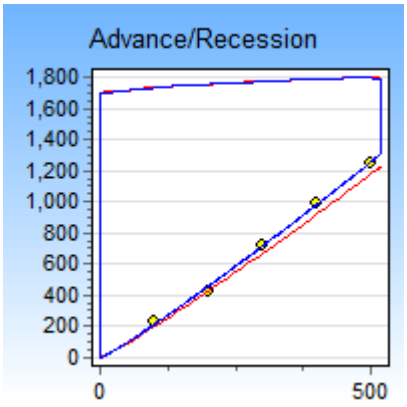


Figure 4.16: Hydrodynamic response-surface investigation for different values of ***Kostiakov fo***. Note that the spikes in the response-surface are artefacts of the grid size, and do not indicate local (or multiple) minima.



Measured Advance

x(m)	t(min)
100	234
200	426
300	724
400	988
500	1246

Flowrate (m^3/sec)	0.00194167 *NOTE: INFILT value =1.941667 l/sec*
Time-to-cutoff (mins)	1690 *NOTE: SIRMOD value =1690 mins Old Value 1689*
Field-length (m)	520
Field-slope	0.00151
Manning n	0.03
Kostiakov a	0.09162 **Previous: 0.10155**
Kostiakov k (m^3/min^a/m)	0.15781662 **Previous: 0.13916**
Kostiakov fo (m^3/min/m)	0 **Previous: 0**
Z-required (m)	280 *NOTE: SIRMOD value =0.28 m Old Value 0.111*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Figure 4.17: Sample calibration output. The red advance curves result from the *INFILT* calibration, while the blue curves result from the hydrodynamic calibration.

An important aspect of the validation that is not shown by these outputs is that the entire operation was performed using a single mouse-click without user intervention. This was performed by clicking on the "Calibrate" hyperlink for the property data record in the *FIDO* decision-support system (see Chapter 7 for more information). The total calibration time was under fifteen minutes for the thirteen sets of data (on a Pentium M 1.6GH processor).

4.6 Conclusions

Two new optimisation-based inverse methodologies were developed for determining the infiltration properties (and in the second method, hydraulic roughness) of the soil. The first method uses a volume-balance time-of-advance equation with a purpose built optimisation algorithm requiring a minimal number of field measurements. The method was found to be reasonably accurate, fast and reliable (the method has successfully been used in practice for the last ten years), although structural differences between it and the hydrodynamic simulation model were identified as a potential source of error.

The second method uses the same optimisation algorithm with the full hydrodynamic model in the objective-function calculations. This requires simulations to be run for each optimisation step, allowing a range of objective-function types to be used. This allows field measurements other than the advance to be used in the calibration whilst also accommodating situations where the inflow is cutoff before the advance reaches the end of the furrow. The hydrodynamic method was found to be accurate, but an order of magnitude slower than the volume-balance method. It was therefore decided to use the two methods in tandem during the solution process. Therefore, the faster volume-

balance method is first used to determine an approximation of the solution parameters, before using these results as input for the hydrodynamic method.

This technique proved reliable with a validation of the hydrodynamic inverse method (advance-based objective function) showing that the simulated outputs from the optimised infiltration parameters provided excellent agreement with the measured advance. Simulated outputs from the *INFILT*-calibrated infiltration parameters showed poorer agreement with the measured advance highlighting the error that exists when mixing simulation and calibration model structures. A response-surface investigation of both methodologies identified true minima for the optimisation to focus upon. Some "roughness" was apparent with the hydrodynamic method due to numerical discretisation errors; although it is not thought that this presents a serious problem to the solution process.

Chapter 5 Automatic optimisation of design and management parameters

5.1 Introduction

The automatic, optimal and reliable determination of design and management parameters to achieve maximum irrigation performance is a desirable goal in developing a surface irrigation decision support system. However, it remains a "stumbling block" for researchers whose attempts have been largely unreliable and inflexible. While a great deal of research has been undertaken in developing simulation models, very few have tried to couple these models to an optimisation algorithm, mainly due to limitations of the underlying model. Also, most of this research has been directed at maximising economic performance rather than the engineering performance of the irrigation.

Therefore, the goal of this research is to develop an optimisation-module for the *FIDO* decision support to enhance furrow and border irrigation design and management. In the process, a user-defined objective-function (based upon maximising the engineering performance of the irrigation system) has been proposed and tested demonstrating the potential of the methodology, and also its limitations. It was subsequently found that a range of optimal parameter configurations exists for this objective-function. Therefore it was recommended that only one design variable be included in the optimisation process. Response-surface "roughness" derived from numerical approximations in the simulation process was also found to impede the optimisation process. A key benefit of the tool, other than guiding design and management, is automatically benchmarking the potential performance of the irrigation system.

The research in this chapter has five main objectives: *(1)* to present a background of optimisation-module development outlining the design issues under consideration; *(2)* to develop a new user-defined objective-function for optimising furrow and border irrigation performance; *(3)* to develop an object-oriented algorithm for implementing automatic optimisation capabilities into the *FIDO* decision support system; *(4)* to evaluate response-surfaces for different configurations of the objective-function; and *(5)* to demonstrate the utility of the new method highlighting its strengths and weaknesses.

Two appendices accompany this chapter. The first presents the results of a response-surface analysis of different objective-function formulations (Appendix 5.1), and the second presents output from optimisations performed using the *FIDO* software (Appendix 5.2).

5.2 Background to optimising surface irrigation practices

Before developing the optimisation-module for the decision support system, the operational functionality of the module needs to be clearly defined, the

objectives of the development identified, the components of the module recognized, and methodology concerns considered. Each of these will now be discussed in turn.

5.2.1 What is the automatic optimisation of surface irrigation practices?

In the context of this dissertation, the automatic optimisation of surface irrigation practices is the process of using the decision support software to automatically calculate the optimum values of design and management variables to maximise the "engineering performance" of irrigation systems. This involves the coupling of an optimisation algorithm to the simulation engine to minimise a user-defined objective-function based upon four key hydraulic performance parameters relating to engineering performance; maximise storage efficiency, maximise application uniformity, minimise runoff, and minimise deep drainage.

The literature review (Chapter 2) showed that most of the irrigation optimisation procedures have focused upon optimising economic profit associated with performing the irrigations. In doing so, many authors have linked the hydraulic simulation model to external factors including crop return, irrigation scheduling, water quality, and labour and water costs. While this is potentially very powerful, it was assumed in the current research that maximum economic profit is inherently highly correlated to the minimisation of water losses while achieving the required depth of application in a uniform manner. Therefore, external economic and social factors were not considered as part of this research.

5.2.2 Objectives of optimisation-module development

The primary goal of the work reported in this chapter is to develop an optimisation-module capable of being implemented into a decision support system for furrow and border irrigation. This involves (a) coupling an optimisation algorithm with the simulation engine using an object-oriented programming structure, and (b) developing an objective-function for generating system response. The specific performance objectives of the optimisation-module are:

- It must be able to determine the optimum design and management parameter without user intervention;
- The answer provided must be independent of the parameter starting estimates input into the optimisation;
- The objective-function should be configurable for a range of design and management priorities, and also interchangeable with other objective-function types;
- There should be no restriction on the type of irrigation parameters included in the optimisation process. A limit on the maximum number of design parameters will be set to three; and
- The major components of the optimisation-module including the optimisation algorithm and simulation engine must be interchangeable with other alternatives for future development.

5.2.3 Elements of the optimisation-module

Conceptually, the optimisation-module required for optimising surface irrigation practices is composed of four principal elements (Figure 5.1):

- decision variables;
- a simulation engine;
- an objective-function; and
- an optimisation engine.

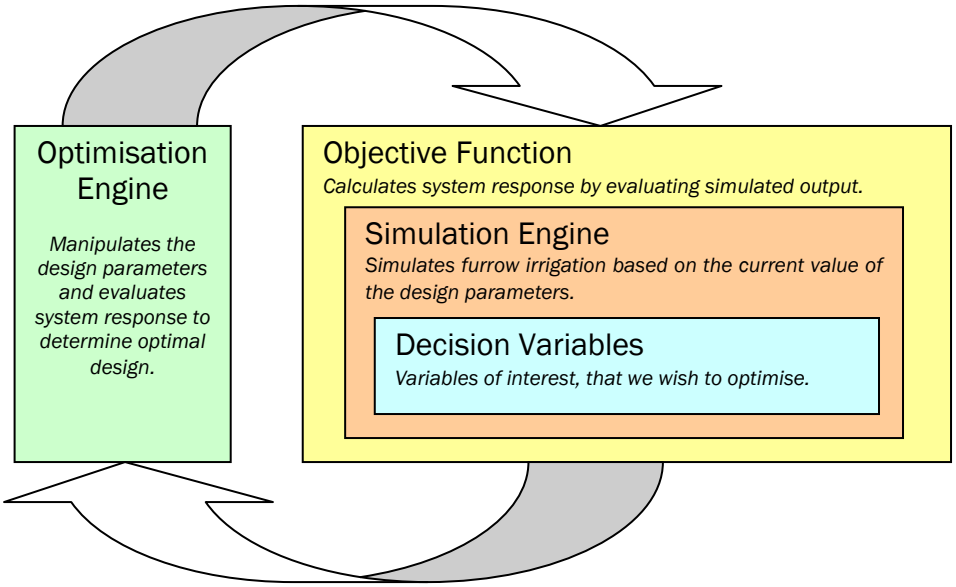


Figure 5.1: Fundamental elements of the optimisation component.

The *decision variables* represent the model variables of interest that we are able to change in the field to improve our irrigation performance. In furrow and border irrigation, this typically includes management variables such inflow-rate and time-to-cutoff, but could also include field design variables such as field-length and slope. The *simulation engine* is required to evaluate irrigation performance for the given set of decision variables. The *objective-function* represents a minimisation or maximisation function that is composed of irrigation performance values, or possibly external factors including costs associated with irrigating and growing the crop. The *optimisation engine* is the computer algorithm that manipulates the decision variables in order to minimise or maximise the objective-function, which it must achieve without user intervention.

The conceptual input/output functionality of the optimisation-module is displayed in Figure 5.2. This suggests that the design or management variables such as flowrate and/or time-to-cutoff are passed into the optimisation-module. An objective-function is also required as input to the module. These input choices can be selected by the user through an appropriate graphical user interface. The optimisation process is then performed before the optimised variables are presented as output from the module, and returned to the decision support system.

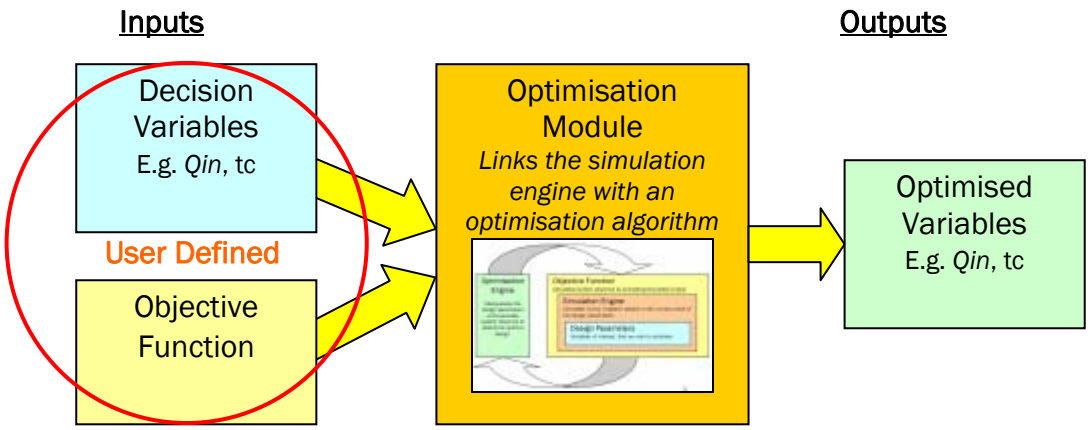


Figure 5.2: Conceptual input/output functionality of the optimisation-module.

5.2.4 Methodology considerations

From the review of the literature (Chapter 2), it doesn't appear that any of the research into automatic optimisation of furrow and border irrigation design and management variables has been developed into publicly available software for decision support. Software such as *BORDER* (Strelkoff *et al.* 1996) and *BICADM* (Maheshwari and McMahon 1991) do have a similar role but are based respectively on a stored database of runs, and a regression analysis of runs, for a fixed number of conditions and do not employ optimisation. No clear evidence could be found to provide reasons for the lack of progression of the technology, although it could be inferred that there are problems with the simulation engines, and/or problems with the optimisation algorithms in the techniques presented.

Conceptually, the automatic optimisation requirement is a simple mechanism. If the simulation engine is accurate and robust, the optimisation algorithm is powerful, and the objective-function is well defined, then theoretically, determining the optimum parameter values should be straightforward. However, in practice this turns out not to be the case.

Problems can occur at the simulation level. While a simulation engine may prove robust and reliable simulating normal field conditions, it is very likely that problems will arise when trying to model the range of parameter combinations that will be presented during the optimisation. It is an unfortunate reality that unusual conditions will be presented to the simulation engine at some time during the optimisation process. While parameters can be constrained to minimise this, the constraint domain across a multi-parameter spectrum is dynamic and not rectangular. Therefore, the simulation engine must be robust enough to handle impractical irrigation configurations.

Problems can occur at the optimisation level. Speed of optimisation and radius of convergence are issues to be considered. Many methodologies exist that could be used, with each having different strengths and weaknesses. Modern optimisation practices often recommend using a combination of methods in order to take advantage of each method's strengths.

Problems can occur at the objective-function level. There needs to be a unique minimum or maximum in the objective-function response-surface in order for the optimisation to determine the optimum parameter values. If a ridge or valley of peak objective-function values is present (as was found with this research) then multiple sets of optimum parameter values will exist. The surface also needs to be smooth and free of local minima. Any roughness in the response-surface that may occur due to numerical approximation in the simulation procedure could impede the optimisation process in its trajectory across the surface.

These were all issues faced during the development of the optimisation-module.

5.3 Objective-function formulation

Given that the simulation engine and optimisation algorithm have already been developed in other chapters of this dissertation, the only new piece of theory that needed to be developed for the optimisation-module (other than the computer algorithm) was the formulation of an objective-function for the optimisation.

It was apparent from very early on in this research that the objective-function should be customisable, due to the range of management conditions that could be considered. The traditional approach to design and management using hydraulic simulation models was typically based upon the objectives of maximising application efficiency, storage efficiency, and distribution uniformity. However, these objectives can change depending on site constraints (e.g. soil characteristics, water availability) and management variables (e.g. agronomic limitations, labour requirements).

For example, over-irrigation is the main contributor to irrigation inefficiencies. An irrigator typically over-irrigates by at least 30% to achieve the required depth of application over 80% of the field (Trout 1990). Furrow-to-furrow inflow and infiltration variations leave portions of the field under-irrigated. The natural response of the irrigator is to apply more water to maintain crop yields. In this situation, the management decision that the irrigator is faced with is not one of improving efficiencies, but one of balancing water and nutrient losses with that of crop yield.

Another example is where an irrigation system recycles the runoff-water. The optimisation objective may then require more emphasis on minimising deep percolation than runoff, whereas a system growing crops sensitive to water logging would more appropriately be optimised according to opportunity time and application efficiency. The objective-function chosen will also depend on whether the user is optimising field design or management practices.

For this research, an objective-function was developed using a weighting system based on the user-specified preference of minimising runoff, minimising deep percolation, maximising storage, and maximising application uniformity:

$$OFV^* = w_1\left(1.0 - \frac{RV}{TV}\right) + w_2\left(1 - \frac{DPV}{TV}\right) + w_3 SE + w_4 AU \quad \text{(5.1)}$$

where OFV^* is the temporary objective-function value that can vary in range from 0 to 1.0; w_1,w_2,w_3,w_4 are weighting coefficients that add up to 1.0; RV is the runoff volume; TV is the total applied volume; DPV is the deep percolation volume; SE is the storage efficiency; and AU is the application uniformity.

Complete simulations must be undertaken to calculate the objective-function. For a (theoretically) perfect irrigation with 100% storage efficiency, 100% uniformity and no losses, the value of OFV^* would be equal to 1.0.

Another key consideration in developing the objective-function is that the advance needs to reach the end of the furrow to avoid having parts of the field left under-watered. It could not be guaranteed that Eqn. 5.1 would avoid this occurrence. Therefore, an optional penalty-function can be applied to ensure that the entire furrow is watered. This is formulated based on two conditions. *Firstly,* if the advance has reached the end of the field (or if the penalty-function is ignored), the objective-function value is calculated as:

$$OFV = OFV^* \quad (5.2)$$

Otherwise, if the field is under-watered, the objective-function value becomes:

$$OFV = 0 \quad (5.3)$$

Therefore the objective of the optimisation is to minimise a reconfiguration of the value of OFV, which is represented by formulation:

$$ObjectiveFunction : Min\left[1.0 - OFV\right] \quad (5.4)$$

Alternatively, this could be formulated as a maximisation problem.

Another option that was considered during this research was to constrain the design based upon individual elements of the objective function. For example, using storage efficiency as a constraint could be used to insure the completion of the advance phase, rather than using the penalty function. Individual performance components could even be removed from the objective function if used as a constraint. However, the optimisation engine does not consider "external" constraints in its present form. Therefore it was decided to limit the study to use the objective function presented above, as it should still be able to implicitly handle any suggested design and management demands.

5.4 Computer algorithm development

Aspects of the computer algorithm that had to be developed for the optimisation-module include; an object-oriented structure for performing the optimisation; and an algorithm for implementing the objective-function.

5.4.1 Developing a structure

The *FIDO* optimisation-module has been developed using an object-oriented structure using the C++ language. The benefits of using object-oriented code have been highlighted in Chapters 3 and 4, and also hold true here.

The object-oriented structure required for the developing the *FIDO* optimisation component is similar to that developed in Chapter 4 for the calibration component. The main difference between these structures is that the optimisation component does not need to incorporate measured data when calculating the objective-function.

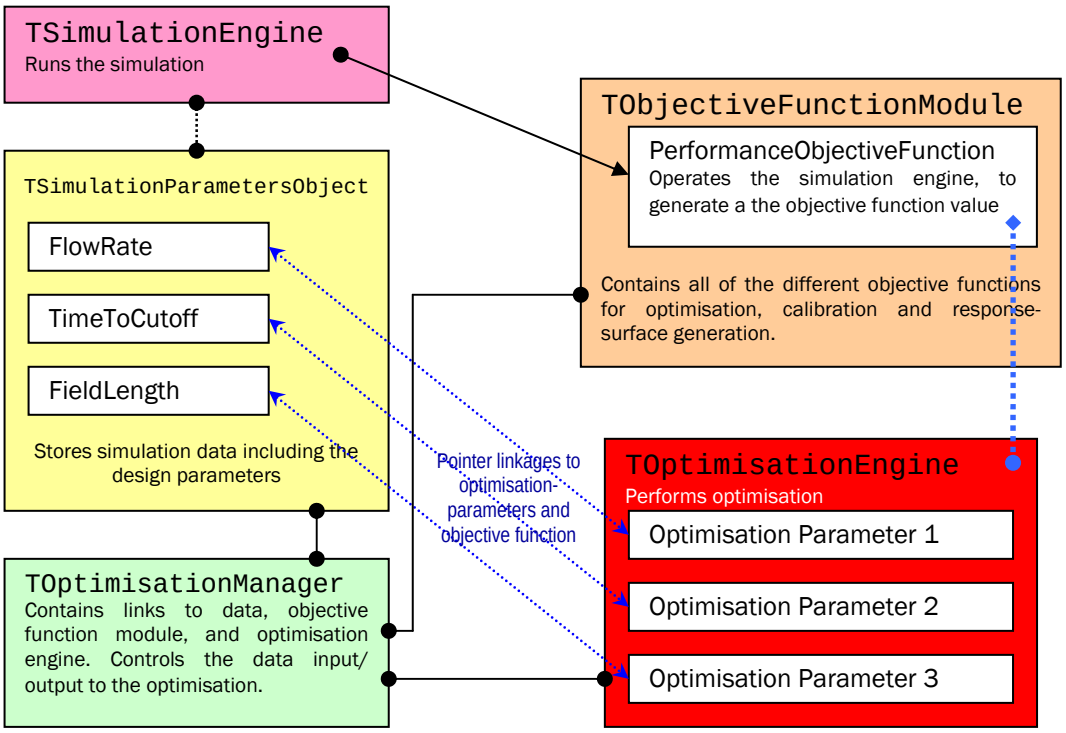


Figure 5.3: Object-oriented components for optimisation algorithm.

The central object in this structure is the `TOptimisationManager` class (henceforth known as the optimisation manager). This class contains all of the functionality to communicate with the decision support system and perform the task of optimising irrigation practices. It contains pointer links to the `TObjectiveFunctionModule`, which is a repository for all the available objective-function objects. The `TSimulationEngine` class is linked to each objective-function and is not required by the optimisation manager. However, the `TOptimisationEngine` component is linked to the optimisation manager, which controls to operation of the optimisation process.

Before optimisation commences, pointer links are used by the optimisation manager to connect the selected decision variables (from the `TSimulationParametersObject` object) and objective-function to the optimisation engine. The optimisation engine commences operation by changing the design values and updating the objective-function. Results are stored for each iteration and the progress is reported back to the optimisation manager, which passes this information back to the decision support system. When the optimisation is completed, the new values of the decision variables are stored alongside their original values (no replacement is undertaken) for further post-processing and reporting.

5.4.2 Objective-function algorithm

Figure 5.4 outlines the algorithm used to calculate the objective-function value for each step of the optimisation. Calculation involves running the simulation before checking that it was successful. The function will then calculate the individual performance parameters and temporary objective-function value. If the advance reaches the end of the furrow during a simulation, the function will return the temporary value, otherwise it penalises the result (if desired). Once the new objective-function value is returned to the optimisation algorithm, the process restarts.

```
Reset Simulation
Load New Parameters
Run Simulation
if (Simulation was successful)
{
    Calculate Performance Values
    Calculate temporary objective-function value (OFV*)
    Apply penalty function if required, to calculate objective-function value (OFV)

}
else
{
    Objective-function Value = NULL     This leaves a hole in the response-surface which the optimisation penalises
}
Return the result to the optimisation algorithm
```

Figure 5.4: Objective-function algorithm

5.4.3 Optimisation algorithm

The literature review (Chapter 2) has identified that a range of optimisation techniques have been used to optimise irrigation performance, with no particular methodology being seen as superior. However, from a programming perspective, the algorithm chosen must be numerically efficient, and be robust enough to handle a degree of noise in the results.

Chapter 4 outlined the development of a simple optimisation algorithm designed to satisfy these requirements through not requiring derivative function calculations and by forcing parameters to change in the presence of local minima. The same algorithm is reused here to determine the optimum design and management parameters for furrow and border irrigation. Other benefits of this algorithm are that it can easily incorporate any objective-function, and it can handle any number and type of design variables (without having to redevelop the objective-function each time).

5.4.4 Decision variable selection and constraints

Due to limitations that will be highlighted in the next section, the current version of the software has been temporarily “hobbled” so that time-to-cutoff is the only decision variable that can be included in the optimisation. Nevertheless, the optimisation-module has been designed so that the user can choose which design and management variables are included in the optimisation. This currently includes field-length, slope, discharge, time-to-cutoff, and the required

depth of infiltration, but can be extended to include functions representing variable inflow and variable field slope.

Upper and lower variable constraints have been linked to each decision variable and can be changed by the user. The range of these constraints must be practically feasible and implementable in the field. For example, flowrate must be greater than the steady state infiltration rate of the soil, and small enough not to cause soil erosion. This range can be further narrowed by considering the range of flows that can be delivered by the irrigator's infrastructure. Narrow parameter ranges can simplify the optimisation through avoiding having to evaluate impractical designs.

Under real field conditions, many of the decision variables including flowrate and field-length have a fixed integer set of acceptable values. Field-length is generally dictated by property and paddock boundaries, which can only be practically split into "fractions". Flowrate is often dictated by manufacturers' siphon or gate sizes. Therefore it would be beneficial for the parameters to take on predefined discrete integer values during the optimisation. However, this has not been considered for this initial version of the software. The nature of this research was more academic, with the primary objective of determining the "optimum" design. Therefore, further research is required to equate this to real field conditions, as the optimisation engine will need some modifications to cope with discrete parameter values.

5.5 Investigation of objective-function response.

Response-surfaces for different combinations of flowrate, time-to-cutoff and objective-function weighting coefficients have been generated to study the suitability of Eqns. 5.1 to 5.4 as an objective-function for furrow and border irrigation design and management. For this study, the penalty function (Eqn. 5.3) was omitted from the objective-function calculations as the purpose was to investigate the fundamental shape of the response-surface without any modifications.

This study was broken into four stages. *Firstly*, the response-surfaces for the different irrigation performance measures were generated to understand the nature of each component, their interrelationships and their contribution to the objective-function response. *Secondly*, the basic form of the objective-function response-surface was compared for different management strategies. *Thirdly*, the response-surface for the default-configuration objective-function was closely examined to identify key characteristics. *Finally*, this response-surface was regenerated for different combinations of a third decision variable (field-length) to further investigate the nature of the peak response.

5.5.1 Response of irrigation performance measures

The first objective of this study was to investigate the response-surfaces for different irrigation performance measures including objective-function response, irrigation efficiency measures, and volume-balance components. Figure 5.5

presents the response-surfaces from this analysis including that derived from the default configuration of the user-defined objective-function (w_1=w_2=w_3=w_4=0.25).

All figures are characterised by the absence of a unique global maximum. Peak regions are characterised by ridges, plateaus, and constraint-defined-highpoints (which are pseudo-maxima with no practical benefit). They are also free from local maxima, which should at least help the optimisation process detect the peak locations.

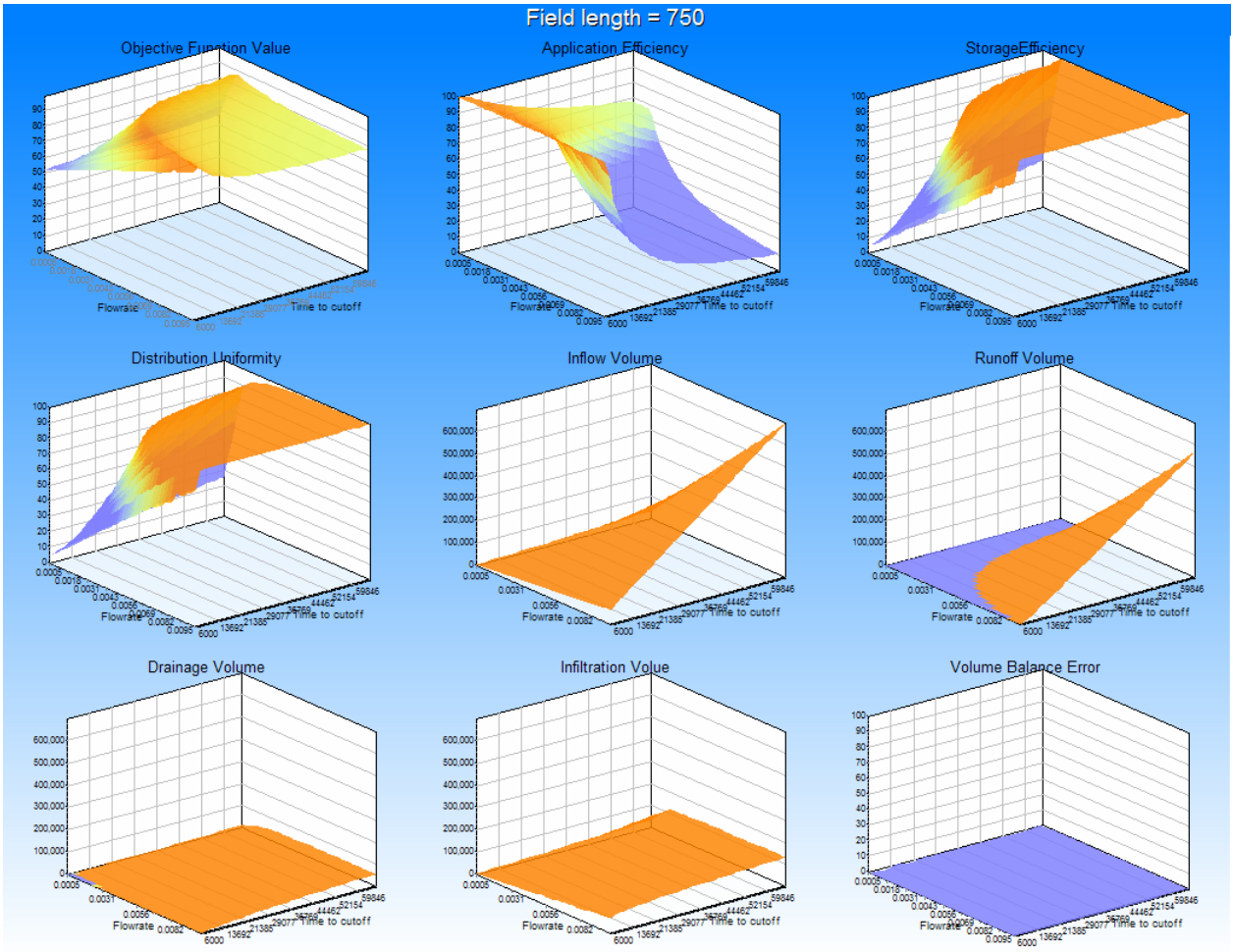


Figure 5.5: Response-surfaces for irrigation performance measures

The objective-function response-surface (shown in the top-left hand corner of the figure) is characterised by a ridge of near-constant performance values. This suggests that many different combinations of flowrate and time-to-cutoff can be used to achieve a similar level of performance. The ridge aligns very closely with constant levels of application efficiency, storage efficiency, and application uniformity. With the exception of runoff volume, the individual volume-balance components do not share this characteristic surface curvature.

Figure 5.5 shows the interrelationships between key performance outputs. For example, application efficiency is seen to decrease as storage efficiency and

application uniformity increase. In the region of the intersection of these surfaces lies the ridge of maximum performance as defined by the objective-function response. Other graphs in Figure 5.5 indicate that application efficiency decreases as inflow volume, runoff volume and drainage volume increase.

Figure 5.5 also demonstrates how objective-functions with surfaces exhibiting constraint-defined-highpoints are ill-suited for optimising performance. For example, the response-surface for application efficiency exhibits a pseudo maximum at the lower limit of flowrate and time-to-cutoff. In practice, choosing these design values would lead to a very poor irrigation associated with very low values of storage efficiency and application uniformity (as can be seen from the corresponding surfaces). Therefore, application efficiency cannot be used on its own as an objective-function for irrigation management (unless other performance indicators are used as external constraints).

5.5.2 System response for different management strategies

The second part of this study investigated different combinations of objective-function weighting factors to examine the different forms of the objective-function response. The weighting factors were chosen based upon practical management alternatives such as maximising performance efficiencies, minimising loss components, or emphasising particular performance components over others. These results are summarised in Table 5.1 while the graphical response-surface outputs are presented in Appendix 5.1. Both orthographic and perspective views of the objective-function response-surfaces have been included to help visualise the true nature of the surfaces.

For each combination of weighting factors summarised in Table 5.1, the objective-function response-surface failed to demonstrate a unique global maximum for the optimisation to focus. The results ranged from having a ridge of maximum performance (e.g. the equal weightings example), to a plateau of maximum performance (e.g. maximising storage efficiency), to an undefined asymptotic-like maximum (e.g. maximise application uniformity). This suggests that optimal management decisions based upon these strategies are going to require extra information so that practical constraints can be added to the system to better define the optimal solution. With these constraints in place, a unique “apparent” maximum may possibly be found. The alternative is to limit the number of decision variables included in the optimisation.

Table 5.1: Summary of response-surface results for different optimisation weightings.

Example name	w_1	w_2	w_3	w_4	Maxima Identified	Surface optimum type
Equal weightings	25%	25%	25%	25%	defined, multiple	"Level" Parabolic ridge
See Figure A5.1.1	*Notes: Presence of near level ridge.*					
Maximise storage efficiency	0%	0%	100%	0%	defined, Infinite	Plateau
See Figure A5.1.2	*Notes: Surface is the same as the response-surface for Storage Efficiency.*					
Maximise application uniformity	0%	0%	0%	100%	undefined	Asymptotic
See Figure A5.1.3	*Notes: Surface is the same as the response-surface for Application Uniformity*					
Minimise runoff	100%	0%	0%	0%	defined, infinite	Plateau
See Figure A5.1.4	*Notes:*					
Minimise drainage	0%	100%	0%	0%	Defined, infinite	undefined
See Figure A5.1.5	*Notes: dependent on time-to-cutoff*					
Maximise storage efficiency	50%	50%	0%	0%	undefined	Optima increasing
See Figure A5.1.6	Notes:					at parameter limits.
Neglect application uniformity	33%	33%	33%	0%	Defined, infinite	"Level" Parabolic ridge
See Figure A5.1.7	*Notes: Presence of near level ridge.*					
Emphasise maximising storage efficiency	16%	16%	**50%**	16%	Defined, infinite	"Level" Parabolic ridge
See Figure A5.1.8	*Notes: Presence of near level ridge.*					
Emphasise maximising application uniformity	16%	16%	16%	**50%**	Defined, infinite	"Level" Parabolic ridge
See Figure A5.1.9	*Notes: Presence of near level ridge.*					
Emphasise minimising runoff	**50%**	16%	16%	16%	Defined, infinite	"Level" Parabolic ridge
See Figure A5.1.10	*Notes: Presence of near level ridge.*					
Emphasise minimising drainage	16%	**50%**	16%	16%	Defined, infinite	"Sloping" Parabolic ridge
See Figure A5.1.11	*Notes: Presence of near level ridge.*					

5.5.3 Closer examination of response-surface characteristics

The third part of this study investigates prominent features of a typical objective-function response-surface. Figure 5.6 presents such a surface (which is a detailed view of that displayed in Figure 5.5) derived from the default objective-function configuration, where weighting coefficients are set equally to 25%. Note that "bumps" displayed in the surface are mainly a result of the grid spacing used in generating the surface, and are not indicative of any error or irregularity.

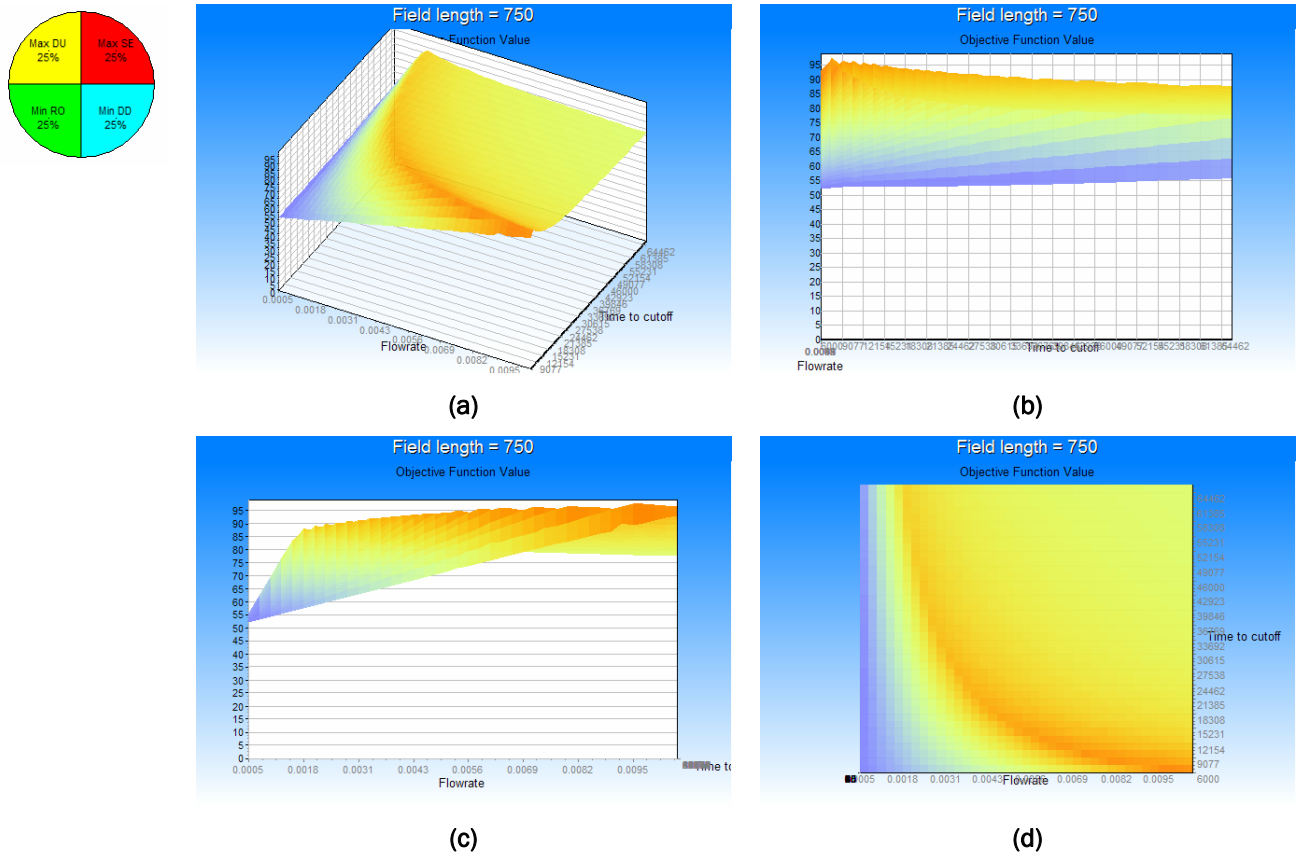


Figure 5.6: Response-surface for equal weightings of the objective-function components.

As indicated in the previous analysis, the resulting response-surface shows a "near-level" curved ridge of maximum objective-function value. However, closer examination of the ridge (Figure 5.6 c&d) reveals that it is not perfectly level, and that a very slight slope exists, showing performance decreasing with increased time-to-cutoff.

From a practical point of view, it can be argued that this effect is negligible and that different combinations of these design values occurring along the ridge will achieve a "similar" level of performance. This suggests that the optimisation process simplifies down to optimising on only one parameter; time-to-cutoff. Even if this assumption is incorrect, and that the slope does indicate that some "optimal" configurations would perform better than others, the nature of the slope implies that the better performing irrigations will occur at the higher inflow rates. In this case, the best management decision would be to choose the largest inflow rate possible for "safe" irrigation (avoiding erosion) and optimise on time-to-cutoff.

The effect of the sloping ridge may not be as significant as that presented in Figure 5.6. Given that the surfaces presented here are generated over a wide range of flowrates and cutoff times (0.5 l/s<Qin<10 l/s and 2 hrs<tc<18 hrs), the variation in peak performance values over the physically viable ranges would be considerably less than shown here. For example, an irrigator's inflow capabilities may range from 2 l/s to 5 l/s with irrigation times ranging from 5 hrs

to 10 hrs. This would restrict the optimisation to the central portion of the response-surface, with considerably less performance variation.

There is also a suspicion that significant volume-balance errors are influencing these results for lower values of time-to-cutoff. Figure 5.7 shows a magnified view of volume-balance errors which range from about -0.2% to -4%. This shows that the larger (more negative) volume-balance errors occur at the very low values of time-to-cutoff. One reason for this is that the fixed time-step of ten minutes that was used for all simulations during the response-surface generation is ill-suited for such low cutoff times, resulting in relatively few nodes in the simulation solution grid. Another reason is that the difficult-to-model simultaneous advance and recession phase predominates with these low cutoff times, which is prone to higher volume-balance errors. A third reason for this is that "percentage volume-balance errors" are magnified for small values off applied water, as is associated with the small cutoff-times.

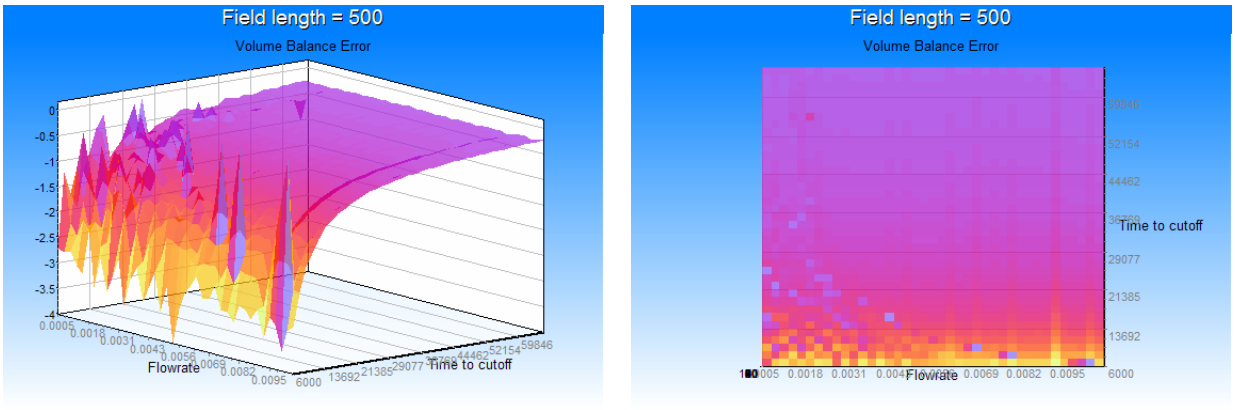
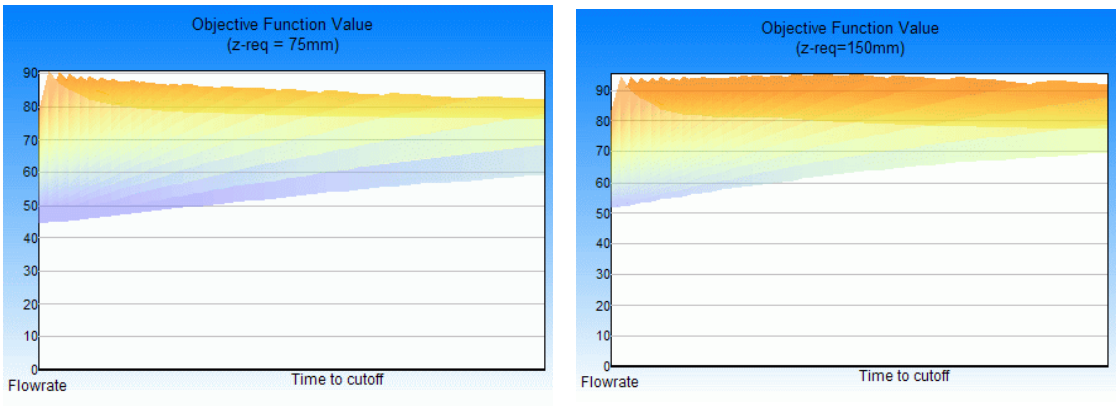


Figure 5.7: Dependence of volume-balance error on time-to-cutoff.

Unfortunately, these volume-balance variations could have a significant effect on the robustness of the optimisation process. This form of surface roughness can sometimes manifest itself as local minima, independent of the optimisation process used. This effect would be more pronounced with more than one solution variable included in the optimisation.

The sloping ridge effect was also found to be influenced by the value of the *z-required* variable, with larger values of *z-required* producing more level ridges (Figure 5.8). This appears to be a result of deep drainage having a larger impact (percentage wise) on the objective-function when using small values of *z-required* at lower cutoff times, while deep drainage has less of a contribution when larger values of *z-required* are modelled. This seems to introduce a slight shift in the surface trajectory that manifests itself as a change in slope of the peak response ridge.



(a) (b)

Figure 5.8: Influence of z-required on slope of maximum-ridge (a) z-req =0.075 m and (b) z-req=0.15 m

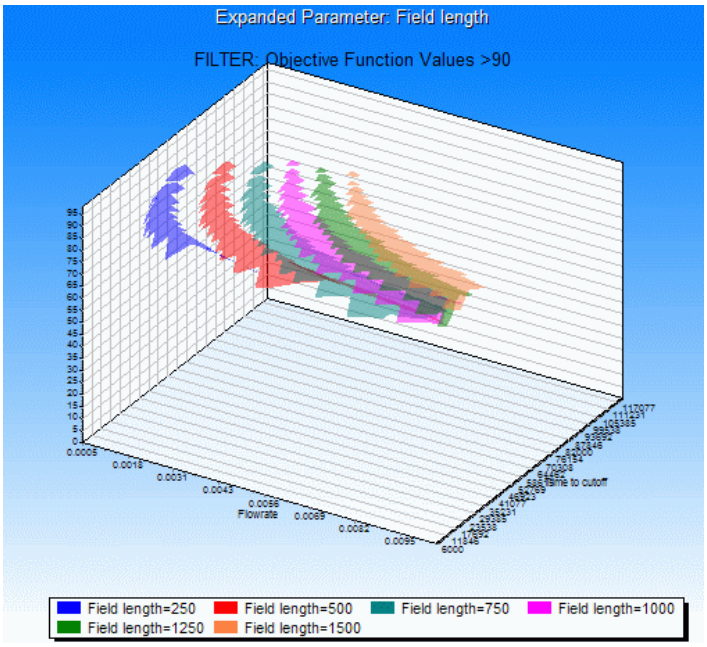
While a true sensitivity analysis was not carried out on the three decision variables, a similar analysis by Zerihun *et al.* (1996) showed that flowrate and time-to-cutoff have the highest impact on irrigation performance, with the system being less sensitive to field length. The authors also stressed that these results are subjective indicating the difficulty in undertaking the multivariate analysis.

5.5.4 Variations in system response for different field-lengths

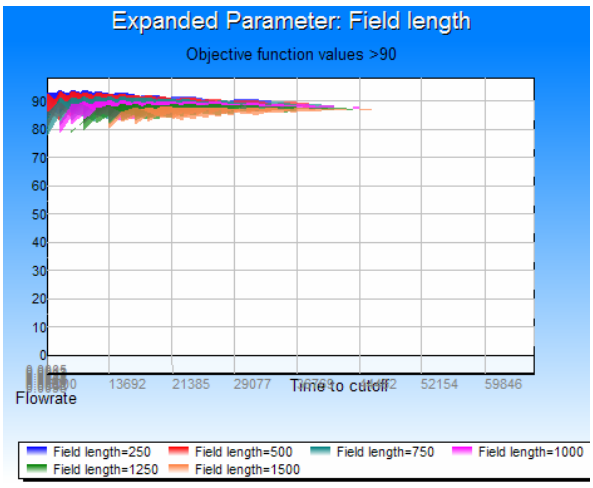
The last part of this study involved regenerating the default-configuration objective-function response-surfaces for different field-lengths. This enabled further investigation of the nature of the ridge of peak response. In particular, it was unknown whether the ridge would remain level for different field-lengths, or whether it would increase in slope or convert into a true global maximum. It was also unknown whether the maximum attainable performance level would increase or decrease with changes in field-length.

Advanced features of the *FIDO* parameter analysis component (see Chapters 6 and 7) were used to help visualise these results. This included overlaying all response-surfaces onto the same chart, and filtering the results so that only the ridge of peak response was visible (Figure 5.9). In this example, different coloured response-surfaces represent different field-lengths.

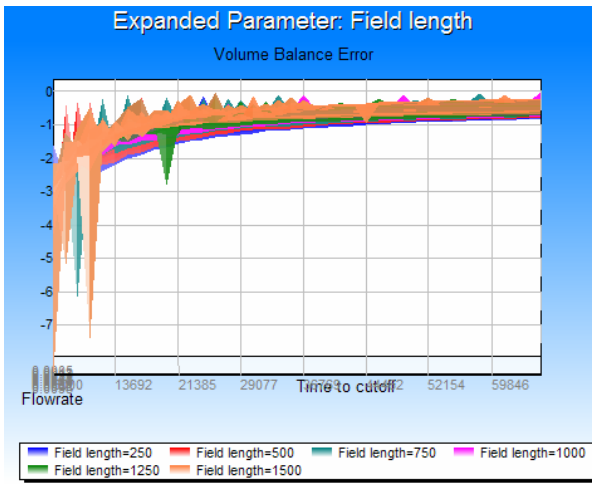
The ridge of peak response was observed to move around the parameter space, but maintained the same maximum attainable performance level for each field-length. This implies that the maximum irrigation performance can be attained for a large range of combinations of the three decision variables. Given that these are the key variables that can be changed by an irrigator, this greatly simplifies decision making by reducing the dimensionality of the problem. That is, in principle, the design and management of surface irrigation can be undertaken by fixing two of the decision variables and optimising on the remaining variable. Typically, field-length and inflow rate would be fixed, and time-to-cutoff would be determined through optimisation.



(a)



(b)



(c)

Figure 5.9: Relationship between (a) and (c) peak objective-function values (filtered) and (c) volume-balance errors.

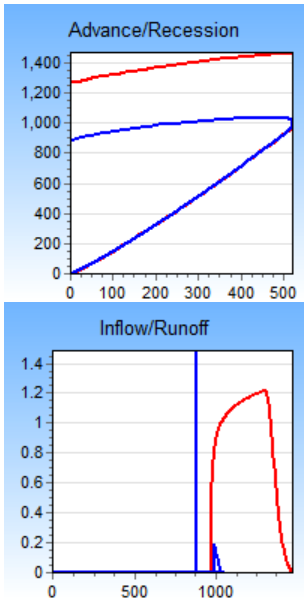
5.6 Optimisation validations.

To test the utility of the optimisation module, the irrigation data used to validate the simulation model (Chapter 3) was optimised for time-to-cutoff, using the default-configuration of the user-defined objective-function. The penalty function (Eqn.5.3) was not applied, and inflow rates were set at their measured values rather than at the maximum permissible rate (which was suggested in 5.5.3).

Solving for one parameter is a relatively straightforward and robust process using the optimisation algorithm developed in Chapter 4. However, the small variations in volume-balance errors exhibited across the surface (shown in Figure 5.7 and Figure 5.9c) would be expected to impede the optimisation process when optimising on more than one variable. In practice, global convergence was easily achieved irrespective of the initial parameter starting estimates.

Appendix 5.2 presents the output of the optimisations taken directly from the *FIDO* decision support software. A sample of this output is presented in Figure 5.10 showing the comparison of irrigation performance values for both the measured data and optimised results.

Optimisation 1. Run by at 09::26 17/11/2006



Optimisation Parameter	Measured	Optimised	Comments
Flowrate	0.001486	0.001486	
Time-to-cutoff	1248	870	
Performance Measure	**Measured**	**Optimised**	**Comments**
Application Efficiency	67.1	94.2	
Storage Efficiency	100	97.7	
Application Uniformity	95.6	91.8	
Applied Volume	111510.7	77569.2	
Runoff Volume	26965.2	262.7	
Stored Volume	74880	73189.4	
Drainage Volume	9715.8	4201.5	

Figure 5.10: Sample output from optimisation validation in Appendix 5.2. Blue lines denote optimised outputs, while red lines represent the measured condition.

The change in performance values is summarised at the property level in Figure 5.11. The optimised results demonstrate a marked improvement in application efficiency accompanied by a subtle reduction in storage efficiency and application uniformity. This suggests that the irrigator has over-watered their field in order achieve the required depth of application at the lower end of the furrows.

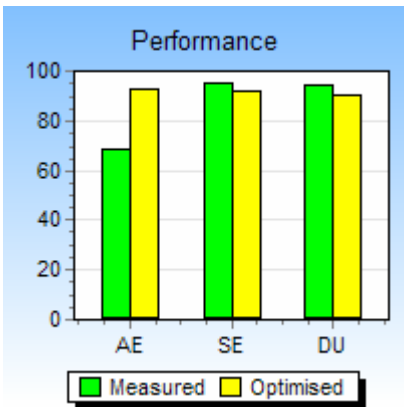


Figure 5.11: Comparison of performance values for optimised versus measured results. (where AE is application efficiency, SE is storage efficiency, and DU is application uniformity)

In this study, the optimised designs consistently watered the whole furrow, even in the absence of the penalty function. In most cases, a very small amount of

runoff did occur. However, in one instance, considerable runoff was seen to occur suggesting that the optimum management strategy will not always involve minimising runoff. That is, sometimes runoff is required in order to increase storage efficiency and application uniformity.

5.7 Discussion

This research is a first attempt at developing an optimisation-module for a decision support system for furrow and border irrigation. It provides a simple and efficient mechanism to demonstrate how to improve the performance of an individual irrigation event. Although problems with the methodology have been identified, this research provides a platform through which more powerful optimising capabilities can be developed.

Two main difficulties were encountered with the existing methodology. *Firstly*, the objective-function does not present a unique optimised solution, but a range of solutions providing similar levels of performance. *Secondly*, the response-surface is characterised by surface roughness that inhibits the optimisation process (note that this has very little impact when only optimising on one decision variable such as time-to-cutoff).

The first problem is not necessarily a drawback of the methodology. From a practical point of view, having a range of decision variables providing the same level of performance is probably an advantage, greatly simplifying the design problem. However, it is possible that this feature could be an artefact of a poorly defined objective-function. Given the nature of the different surfaces presented, it is likely that tighter constraints and limits will need to be applied to the decision variables and objective-functions before multi-variable optimisations are viable. This may include testing designs for satisfactory levels of performance. In effect, this is analogous to the linear programming example of applying constraints to better define the objective-function solution space.

The second problem is probably inevitable. So long as numerical approximations are used in the solution of the hydrodynamic equations, there will be some degree of noise in the objective-function response-surface. This didn't cause any significant problems during this analysis, but it is very likely to impede a multi-variable optimisation. The volume-balance components included in the objective-function seem to be especially sensitive to these variations. Three opportunities exist to minimise the effects of this problem. *Firstly*, the simulation model solution technique could be refined to improve numerical accuracy and stability. Options available for this could include; general optimising of code; decreasing the simulation time-step size (although this will cause optimisation times to increase); improving the stability measures introduced into the solution process; or even adopting a simpler model type. *Secondly*, the objective-function could be redefined to include elements that are less susceptible to numerical errors. For example, using the irrigation advance curve in the calibration objective-function was found to be relatively insensitive to these errors (although this would not be suitable for design and management). *Finally*, the optimisation algorithm could be updated to a more robust and globally convergent form.

Another problem for future optimisation research is contending with the spatial and temporal variability of infiltration. For an individual furrow, the optimal time-to-cutoff is usually that which results in the advance just reaching the end of the furrow causing minimal runoff (as was shown earlier, this is not always the case, especially with low infiltration soils). Given that infiltration varies from furrow to furrow, if a constant cutoff-time is used for the entire irrigation, some of the furrows are likely to be under-watered with the advance not reaching the furrow end. Therefore the objective-function must include a component representing infiltration variability to ensure maximum performance across the field.

Another limitation of this study is that it doesn't segregate the independent tasks of design and management. In this case, a single objective-function has been developed to represent both requirements. Although this function can be useful for field design purposes, a more suitable function would be one that aims to simplify irrigation management through minimising the effects of infiltration variability. This would involve incorporating seasonal irrigation summary information into the objective-function to optimise field-length and/or slope to minimise the range of management options over the season. The form of this function will require further research.

Despite these problems and limitations, the objectives of this study (Section 5.2.3) have been satisfied. As the optimisation included only one decision variable, global convergence was attained without user-intervention, with the results independent of the initial parameter estimates. While the software was limited to only optimising on time-to-cutoff, there is really no restriction on the type and number of decision variables that can be included in the optimisation. This is only limited by the type of objective-function that is linked into the module. Because of the object-oriented design of this module, different objective-function types, optimisation algorithms, and simulation components can easily be interchanged.

It is inevitable that this type of tool will not appeal to all practitioners. Many will prefer an interactive design capability, while others will require an engineer somewhere in the loop. Nevertheless, a key benefit of the tool that should appeal to all, is that it can be used to automatically benchmark the performance potential of an irrigation event. Subsequent design and management configurations can than be compared against this optimum value.

5.8 Conclusions

An optimisation-module for the automatic design and management of surface irrigation was developed for the decision support system. This involved combining the simulation engine with an optimisation algorithm and a user-configurable objective-function. The objective-function consists of components relating to the design and management priorities of maximising storage efficiency, maximising application uniformity, minimising runoff, and minimising deep drainage.

It was originally envisaged that this optimisation-module would undertake multi-variable optimisations, such as simultaneously optimising key decision variables (e.g. flowrate, time-to-cutoff and field-length). However, a study of the response-

surfaces of the objective-function formulations failed to identify a global maximum for the optimisation process to focus upon. Instead, the surface maximum was typically in the form of a “level” parabolic ridge. When combinations of the three key decision variables were analysed, this ridge was found to move around in parameter space at a constant level of attainable performance. This implies that maximum performance can be attained for many combinations of the three decision variables that greatly simplify decision making by reducing the dimensionality of the problem. That is, in principle, the design and management of surface irrigation can be undertaken through fixing two of the decision variables and optimising on the remaining one. And added benefit of this tool is that it can be used to automatically benchmark the performance potential of an irrigation event.

Chapter 6 Automated generation of field design and management guidelines

6.1 Introduction

Chapter 5 has reviewed and demonstrated the benefits and problems associated with the automated optimisation of design and management practices. While that methodology provides an optimum real-time management solution, it fails to present an overall picture of the design and management problem, including the effect of spatial and temporal variability of infiltration. Field design and management guidelines (also known as design charts) offer a medium to present this extra information, based upon the recorded irrigation history for a particular location.

Field design charts were one of the earliest forms of design and management aids dating back to the 1960s and were derived from analytical and empirical relationships based upon extensive field trials. Recent simulation models provide a simpler and more efficient way to develop these charts based upon repeated simulations of different irrigation configurations. However, this can be a time-consuming process. For example, design charts that were developed as a preliminary study during this research (Section 6.4) took approximately three standard working weeks to develop. Therefore, there is a need to develop an automated facility to generate these types of charts and to incorporate this facility into the *FIDO* decision support system. This facility is referred to as the "parameter-analysis module".

The research presented in this chapter has five main objectives: *(1)* to present a background for developing the parameter-analysis module outlining the design issues under consideration; *(2)* to develop a methodology for accounting for the spatial and temporal variability of infiltration for inclusion in guideline development; *(3)* to investigate design curve generation through a preliminary case study in order to define the required functionality of the parameter analysis tool; *(4)* to develop a suitable object-oriented algorithm to automate the process of generating design and management guidelines; and *(5)* to generate sample guidelines using the software.

6.2 Background to automating the development of field design and management guidelines

Before developing the parameter-analysis module for the decision support system, the module and its outputs need to be clearly defined, the components of the module recognised, and the objectives of the development identified. Each of these will now be discussed in turn.

6.2.1 What is automated generation of field design and management guidelines?

The literature review (Chapter 2) identified field design and management guidelines as a paper-based design tool generated from the systematic analysis of parameter responses (irrigation performance) using an irrigation model. Also known as parameter-analysis outputs, design curves, design charts, and response-surfaces, these guidelines were one of the first management and design tools available for surface irrigation (although it is doubtful whether they were ever effectively applied in practice). Initially, they were developed from field trial information, empirical relationships, and simple analytical functions. In recent times, they have been generated from the output of computer simulation models in the form of iso-curves and contours.

Many different configuration options exist when setting up design charts. Decisions must be made regarding how many charts to present, what to include in each chart, which elements will be represented by each chart axis, and which elements, or mixture of elements, will be plotted as iso-curves or contours. The last decision is probably the most important one, as it defines the segregation of the system outputs (irrigation performance values) from the system inputs (decision variables), and it ultimately affects the way in which the chart data will be generated.

For example, if the contours and iso-curves represent irrigation performance, then system inputs such as flow-rate, time-to-cutoff, and field-length will be represented by the chart axes. Potentially, this requires thousands of simulations to be run to account for all combinations of the decision variables. The only practical way of generating these outputs is to use an automated process of running the simulation and updating the results.

The alternative to this is to represent irrigation performance on one or more of the chart axes. Then iso-curves representing different values of the decision variables are plotted in the chart space. Typically, only four or five iso-curves will be generated for each decision variable. In this situation, only a relatively small number of simulations need to be run to account for a smaller number of design parameter combinations. This is suitable for manually run simulations.

6.2.2 Objectives for developing a system for automating field-guideline generation.

The primary goal of the research in this chapter is to develop an automated tool for generating field design and management guidelines (charts) capable of being implemented into a decision support system for furrow and border irrigation. This involves developing a new object-oriented computer algorithm for controlling the simulation and generating performance outputs based upon a user's specification. In particular, key objectives of the parameters-analysis module are:

- It must be able to automatically generate the system response given basic user direction;
- The results must be able to be configured to account for the multi-dimensional nature of parameter/objective-function response; and
- The module must be able to take into consideration the effects of spatial and temporal variation of the soil properties.

To maximise the effectiveness of the module, it should be configurable by the user in order to evaluate different forms of output. For example, the users should be able to develop design curves based upon different decision variables and objective-functions. The module should not be limited to analysing irrigation performance, but could be used to generate any kind of response-surface (as was demonstrated in Chapters 4 and 5). Successful development of these features will ensure that the module will be an effective research tool, as well as an operational tool for design and management.

6.2.3 Elements of an automated system to generate field design and management guidelines

The parameter analysis module developed for the *FIDO* decision support system is composed of seven principal components (Figure 6.1):

- design parameters;
- field measurements;
- a simulation engine;
- objective-functions;
- a parameter analysis manager; and
- graphical analyses.

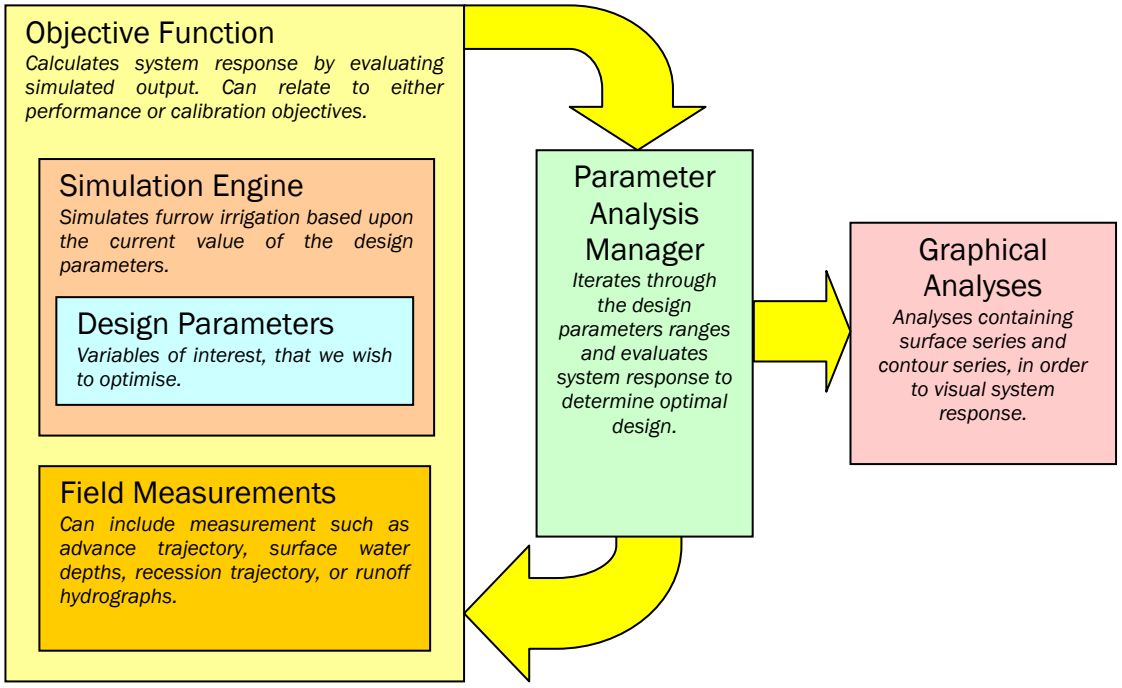


Figure 6.1: Fundamental components of the parameter-analysis module.

The *design parameters* represent the variables of interest that are manipulated in order to generate a response-surface. The *objective-functions* calculate the performance measures, and the error between the field measurements and simulated outputs. The *simulation engine* is required to provide the simulated outputs for calculating response-surface outputs from the objective-functions. The *field measurements* include data such as advance trajectory, surface water depths,

recession trajectory and/or runoff hydrographs and are used when the response-surface is based upon one of the calibration objective-functions. The *parameter analysis manager* provides the computer algorithm that controls the response-surface generation. The goal of the parameter analysis module is to allow the user to generate guidelines for management and design with a minimum of effort.

The conceptual input/output functionality of the parameter-analysis module is displayed in Figure 6.2. Design and/or management variables such as flowrate and/or time-to-cutoff are passed into the parameter-analysis module along with an objective-function and a measure of the infiltration characteristic for the analysis. These input choices can be selected by the user through an appropriate graphical user interface. System response information is then generated by the module before three-dimensional surfaces and contours (and the underlying data) are presented as output from the module, and returned to the decision support system for display and further analysis.

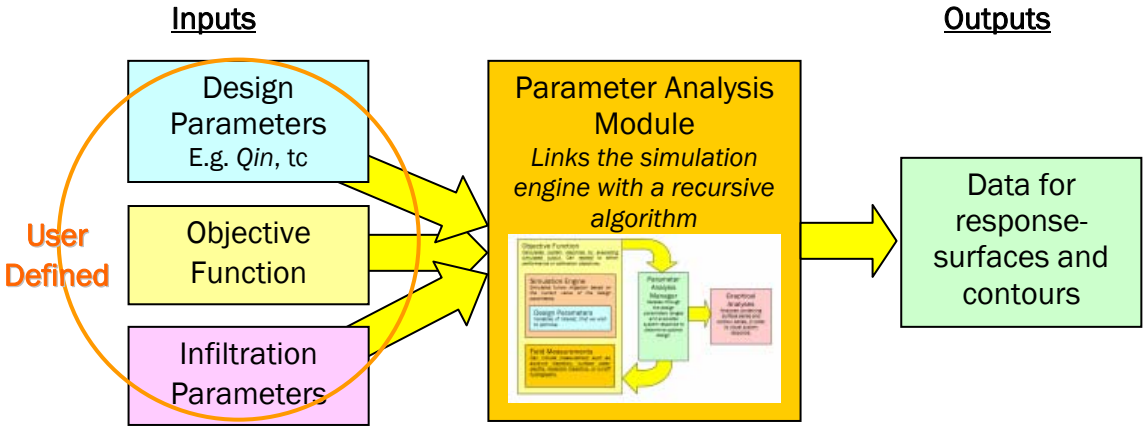


Figure 6.2: Conceptual input/output functionality of the parameter–analysis module

6.3 Accounting for infiltration variation

A key consideration in developing design charts for irrigation design and management is that they must take into account the spatial and temporal variability of the soil. This could be achieved through lumping measured irrigation characteristics together to produce an "averaged" set of design curves. Unfortunately, this doesn't provide any information on the range of performance results that could be expected for different infiltration conditions. Therefore, the upper and lower ranges of infiltration variability could also be included in this analysis to provide confidence limits on design and management decisions.

A simple method for implementing this is to generate three sets of design charts for high, low and average infiltration properties of the soil. This involves a pre-processing of paddock-specific infiltration data to determine three sets of the modified Kostiakov-Lewis infiltration parameters. This requires a range of cumulative infiltration curves for a nominal opportunity-time from a paddock's recorded history (Figure 6.3).

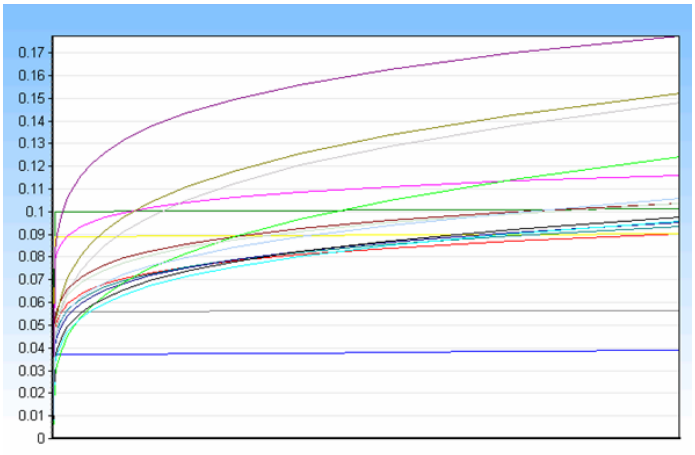


Figure 6.3: Infiltration range is calculated from high/low and average of paddock- specific infiltration curves.

The infiltration parameter sets representing high and low infiltration can be selected directly from the historical set of cumulative infiltration curves. These are chosen from curves with the highest and lowest cumulative infiltration values at the nominated opportunity time.

The average infiltration characteristic is more difficult to determine and requires an optimisation algorithm to fit the infiltration equation parameters to match the averaged result of all infiltration curves. The optimisation algorithm developed in Chapters 4 and 5 was used for this purpose. However, a new curve-matching objective-function was developed to plug into the optimisation, based upon minimising the error been the average infiltration curve, and the fitted equation curve (Figure 6.4).

Average Infiltration Parameters Algorithm

Set SSE = 0.
For(t=0 to number of desired time-increments)
{
    SSE = SSE + (AverageCumulativeInfiltration(t) – PredictedCumulativeInfiltration(t))^2
}
Return the result to the optimisation algorithm.

Figure 6.4: Objective-function algorithm for calculating average infiltration curve

In comparison to the curve-matching objective-functions presented in Chapter 4, this procedure is simple and efficient as no simulation-runs or cubic spline interpolations are required in the calculations. Both the averaged and fitted equation infiltration curves are generated using the same constant time-interval.

In practice, optimising on the three infiltration parameters performed quickly (in only a fraction of a second). However, problems were encountered during development that hindered the optimisation from converging on the true global minimum. In particular, the optimisation was observed to repeatedly default to fitting a straight line through the averaged infiltration curve (Figure 6.a). It was assumed that this was because of a flattening on the response-surface for the

objective-function, causing a sticking point in the optimisation. This was overcome (Figure 6.b) by increasing the sensitivity of the optimisation algorithm.

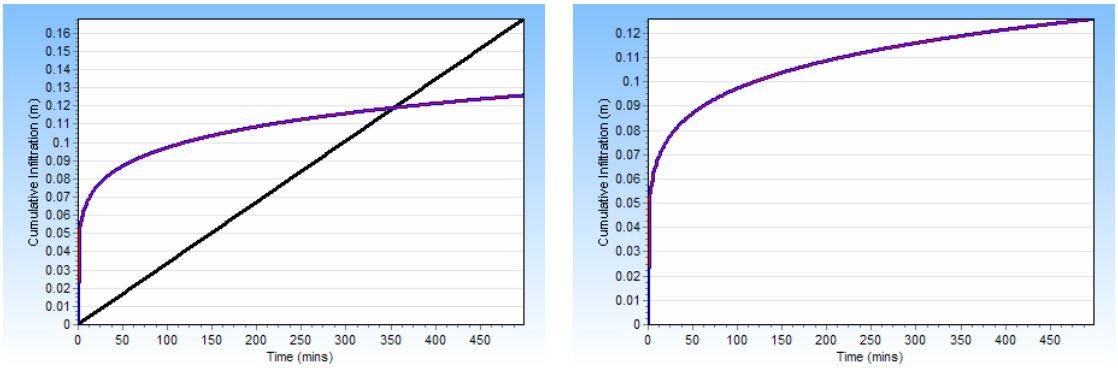


Figure 6.5: Cumulative infiltration curve fitting results showing (a) "sticking point" encountered in early research, and (b) correct curve fitting results.

6.4 Preliminary Study: Development of guidelines for surface irrigation

As a preliminary study in developing the parameter analysis module, site-specific design and management guidelines were prepared by repeated manual simulations using the *SIRMOD* simulation model. The analysis was undertaken to show the effect of infiltration variation on irrigation decision-making, and assist in developing the functionality of the parameter analysis tool.

The guidelines were developed based upon including the three main decision variables (flowrate, time-to-cutoff, and field-length), along with infiltration variability into a single set of design charts. Data was provided from seventeen surface irrigations monitored in the Burdekin Delta region in Queensland Australia.

6.4.1 Field data

The data used in study were collected from irrigations on sugar cane at the Jarvisfield site (Raine and Bakker 1996) during 1994-95 season. Irrigation advance and volume-balance parameters were usually measured on two furrows for each irrigation, and those with more than four advance measurements and complete volume-balance measurements were used in the analysis. The average root zone soil deficit was 0.6 ML/ha and the average volume applied was 1.4 ML/ha. Application rates varied from 2.0-3.4 L/s with an average of 2.6 L/s. The average irrigation time-to-cutoff was 644 mins with a range of 453 to 913 mins.

6.4.2 Pre-analysis of infiltration data

From the seventeen measured irrigations, twenty-two infiltration functions were evaluated for use in the simulation model and for assessing infiltration variability (Figure 6.6a). The Kostiakov-Lewis infiltration parameters were calculated for each set of data using the modified two-point method (Elliot and Walker 1982), while the manning n parameter in the *SIRMOD* model was also adjusted using the measured advance to finalise the calibration (which usually only required a small adjustment).

Then high, low and average infiltration functions were calculated using the methods proposed in Section 6.3 for an 800 min opportunity time.

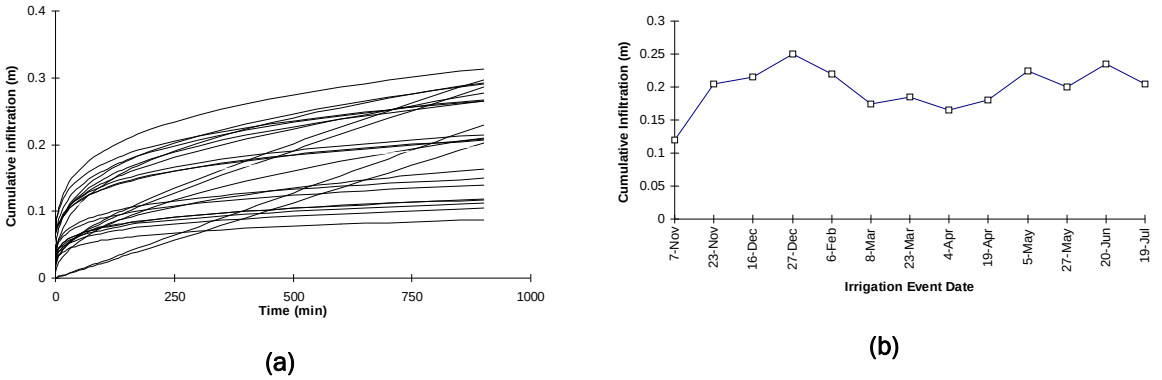


Figure 6.6: Summary of infiltration information Jarvisfield site throughout the 1994/95 irrigation season. (a) Cumulative infilgration curves (b) cumulative infiltration opportunity time of 500mins

Infiltration was found to vary considerably between furrows during the same irrigation, and with time over the season. Total infiltrated volumes for individual irrigations varied from 1.2 to 2.4 ML/ha over the season for an arbitrary time-to-cutoff of 500 mins (Figure 6.6b). The substantial spatial and temporal variability observed during these irrigations reinforces the need to consider the complete range of infiltration characteristics during the generation of design charts.

6.4.3 Evaluation of management strategies

Before developing new design charts, two different management strategies were evaluated using the simulation model and compared against the measured results. The first strategy involved applying the optimum flowrate and time-to-cutoff calculated for the seasonal average infiltration function to all of the irrigations throughout the season. This was done to evaluate the effectiveness of using an average infiltration function in the decision making process, as would be done when using generalised decision curves.

The second management strategy involved manually optimising flowrate and time-to-cutoff for each irrigation during the season. The objective of this optimisation was primarily concerned with maximising application efficiency whereby the irrigation was designed so that the advance would just reach the end of the field with zero runoff occurring. It was intended that this real-time control strategy could be later be used to generate design charts of “optimised performance”.

The results of this analysis are presented in Table 6.1. Manual optimisation of the management practices using the seasonal average infiltration function suggests that an application rate of 3.7 L/s with a cutoff time of 190 mins should be applied throughout the season. Results from using this strategy indicated that the average application efficiency would increase from the measured value of 41% to 71%. However, this corresponded with a decrease in storage efficiency from the measured value of 98% to 83%. The high measured value of storage efficiency is indicative of the commercial practice of completely refilling the root zone. Where

management parameters were optimised for each irrigation during the season, the average application efficiency was seen to increase to 93% with a storage efficiency of 90%. Distribution uniformities remained similar for each management option.

Table 6.1: Performance results for different management practices in the Burdekin Delta Region.

Management practice	Application Efficiency (%)	Storage Efficiency (%)	Distribution Uniformity (%)	Seasonal water application (ML/ha)
Measured	41 (±2)	98 (±2)	92 (±2)	26.4
Optimised from average infiltration function	71 (±2)	83 (±2)	93 (±1)	17.9
Real time control	93 (±2)	90 (±2)	88 (±3)	12.2
Results are presented as the mean (± standard error)				

These results indicate the danger of using the average infiltration function in isolation for design and management, because of the potential for a significant reduction in storage efficiency. However, results based upon this function do provide very useful information and can be used as a "starting point" in the design process. That is, the optimum design can be derived for this situation, before considering alternative infiltration conditions that are likely to occur, to provide a range of possible outcomes. Design values can then be defined based on upon a margin of safety.

On the other hand, the real-time control situation provides an efficient design, but can only be used while the irrigation is occurring, or in "post-analysis" mode to determine the performance potential of an irrigation (as was done in this study). It is this second methodology that can be used to develop design charts, by reporting optimised solutions for a range of decision variables.

Therefore, both of these strategies can contribute to the development of design charts through (a) including the generalised infiltration information as a starting point in the design process, and (b) composing the charts from the output of optimised simulations.

6.4.4 Investigation of design curves.

At the time of this preliminary case study, neither *SIRMOD* nor the *FIDO* simulation engine was capable of being incorporated into any automated batch-processing procedure. That is, *SIRMOD* had no provision for running in a batch mode (and still hasn't), while *FIDO* wasn't robust enough for the purpose. Therefore, the design curves developed in this study were generated from manually run simulations, necessitating that irrigation performance be represented by the vertical chart axes. Even though very few variations of the decision variables were analysed, this still turned out to be a very time-consuming process with the entire study taking about three weeks to complete.

At the start of the study, some experimentation with different design chart configurations was undertaken to investigate the nature of the response, and to evaluate the effectiveness of each chart-configuration. The first attempt (Figure 6.7) was designed to investigate the effect of field-length on irrigation performance using the seasonal average infiltration function for a fixed inflow rate (2.6l/s) and a range of cutoff-times (190min, 360mins, and 720mins). In each case, application efficiency was observed to increase with increasing field-length due to a reduction in tailwater losses. This coincided with a minor reduction in storage efficiency as it became increasing difficult to completely fill the root-zone at the furrow outlet for long field-lengths.

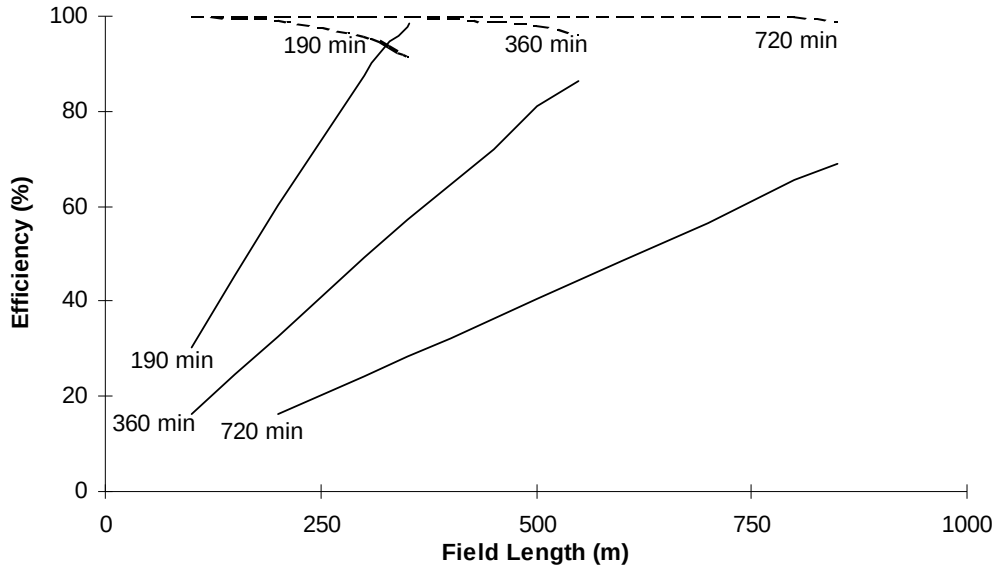


Figure 6.7: Application efficiency (-) and storage efficiency (- - -) for a simulated irrigation performance using the seasonal average infiltration function, a fixed water application rate of 2.6l/s and a range of irrigation periods from 190-270 mins.

In a second example (Figure 6.8), the maximum application efficiency and optimum field-length were presented for the high, low and average infiltration functions, and a range of cutoff times. This highlights the effect of infiltration variability on performance and optimum field design, demonstrating that the optimal field-length for high infiltration conditions is less that half of that for low infiltration conditions. However, for this site, application efficiencies were never higher than 70% for the high infiltration condition, irrespective of the field-length and cutoff time. In was found that deep drainage losses were accounting for the low efficiencies.

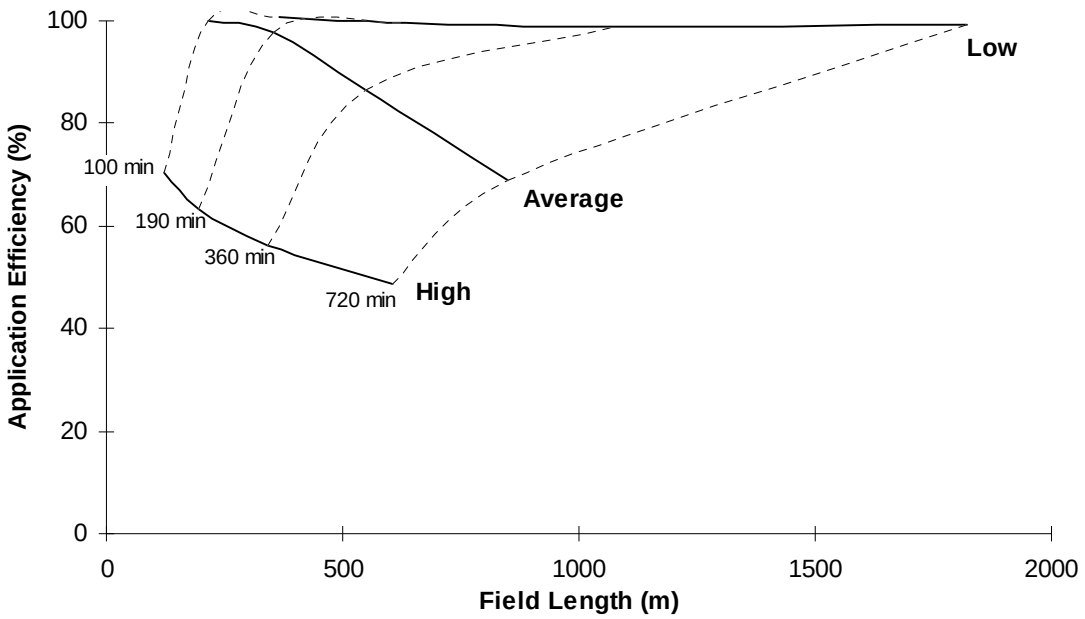


Figure 6.8: The effect of field-length of the maximum application efficiency of the soil with low, average and high infiltration characteristics when water is applied at 2.6l/s for a range of irrigation periods.

6.4.5 Finalisation of guidelines

While both of the previous examples (Section 6.4.4) provide useful information, they are limited by the use of a fixed inflow rate. Therefore, to better represent the multi-dimensional nature of the design problem, the three key decision variables (flowrate, time-to-cutoff, and field-length) were combined into the same set of design charts (Figure 6.9). The only way to achieve this was to apply a fixed management strategy, whereby zero-runoff occurred during the irrigations. This is equivalent to the real-time irrigation situation presented in the Section 6.4.3. To achieve this, field-length was included as a "quasi" system output. That is, each irrigation was simulated using an initial infinite field-length and a preset flowrate and time-to-cutoff. The final irrigation advance location was then designated as the effective field-length for the zero-runoff strategy. By assigning effective field-length to the horizontal axis, and irrigation performance to the vertical axis, iso-curves for different flowrates and time-to-cuff can be plotted on the charts. A matrix of six charts was produced with the columns representing different infiltration characteristics and the rows representing different irrigation performance measures.

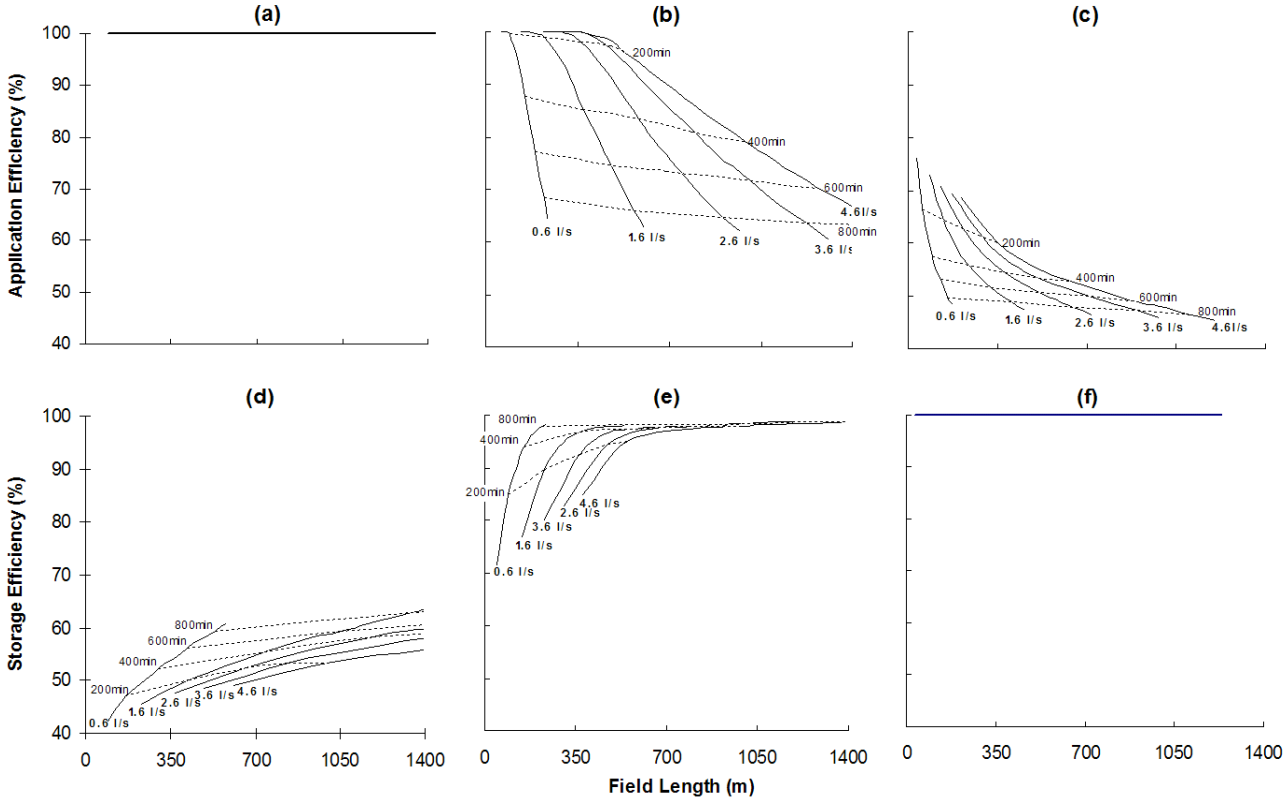


Figure 6.9: Design charts based on (a & d) high, (b & e) average and (c & f) low infiltration characteristics.

From these charts, it is possible to choose an acceptable level of irrigation performance for the average infiltration characteristic, and read off the range of design values for flowrate, time-to-cutoff and field-length to achieve this performance. By repeating this for each infiltration characteristic, a range of design values can be selected to ensure an acceptable level of performance for the different conditions. It is then up to the user to select a suitable design with an appropriate safety margin.

6.4.6 Discussion of case study

The finalised form of the design charts is dominated by three main factors. *Firstly*, the multidimensional nature of the analysis, which includes describing infiltration variation, necessitated that several charts be developed (instead of a single chart) to try and simplify explaining the system response. *Secondly*, the technology available at the time wasn't suitable for automating the process to generate complete response-surfaces. Therefore the only option available was to represent the system outputs by the chart axes and plot a small range of decision variable iso-curves. *Thirdly*, optimised runs with a zero-runoff objective were needed to develop the charts in order to accommodate three decision variables.

At the time, the prospect of producing design charts describing so much information and being based upon optimised simulation runs was seen as an exciting and potentially revolutionary tool for design and management. However, several deficiencies are apparent which limit their practical application.

Possibly the biggest limitation is that only a small portion of the system response is presented in the charts. Specifically, the curves are generated for the zero-runoff situation, which was assumed to be the optimum management strategy. Therefore, they are limited to evaluating and designing irrigations for this specific situation, and are invalid when runoff occurs or is desired. That is, they fail to describe the performance response for non-optimal and alternative priority design and management practices. Because of this they can be misleading.

Also, the charts are not intuitive and can be difficult to interpret, especially for an untrained user. Readability is impaired as the iso-curves tend to be bunched together and interpolation is difficult because of a non-linear interpolation space. There are also many blank regions on the charts including whole charts being empty.

The initial two design charts (Figure 6.7 and 6.9) that describe only two decision variables are generally simpler to understand. However, they are still (arguably) difficult to use, possibly because the chart axes represent both system inputs and outputs. A third decision variable can be represented by producing multiple copies of the charts and substituting decision variables. For example, time-to-cutoff or flowrate could replace field-length on the horizontal axis, which changes the charts from a design tool to a management tool. This was the approach of Hornbuckle *et al.* (2003) who developed design curves with time-to-cutoff on the horizontal axes (instead of field-length) and iso-curves representing different flowrates (Figure 6.10). The relative simplicity of these charts reinforces the benefits of representing only two decision variables.

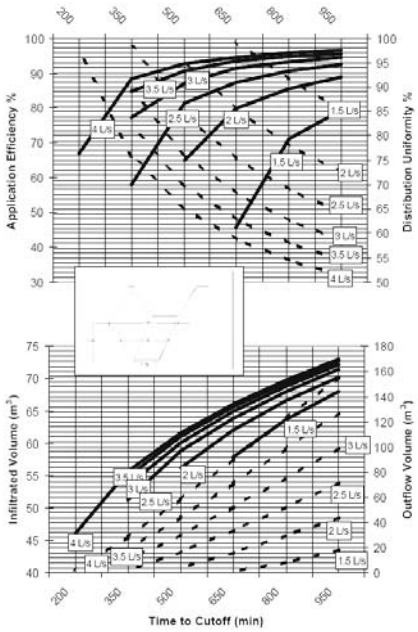


Figure 6.10: Example of a design chart by Hornbuckle *et al.* (2003) for furrow irrigated field on a self-mulching clay soil with furrow length 200m. The solid line in upper chart corresponds to distribution uniformity, and in the lower chart corresponds to the infiltrated volume.

6.4.7 Recommendation from case study

In retrospect, trying to include variable infiltration effects and three decision variables was probably too ambitious with the potential benefits of visualising a large number of dimensions being negated by a limited range of response outputs for a fixed management objective. Nevertheless this provided a basis for developing the guideline-generating capabilities for the decision support software.

The key findings of this study which were used to develop the parameter-analysis module include:

- It is preferable to represent system outputs as contours and iso-curves, rather than by the chart axes. This will maximise the visualisation of the system response and not limit it to showing the results for a particular management strategy.
- It was better to represent different infiltration conditions in separate design charts, rather than trying to incorporate all this information into one chart.
- The choice of which variables can be assigned to each chart axis should be user defined, as different configurations can provide different explanations of the response, and different operational objectives (e.g. design versus management).
- Preferably only two decision variables should be represented in each chart, although multiple system outputs could be represented.

6.5 Computer algorithm development

Aspects of the computer algorithm that were developed for the parameter-analysis module include; an object-oriented structure for generating and displaying the system response; objects for defining the analysis and storing and manipulating the response; and analysis objects for configuring and displaying the outputs

6.5.1 Developing a structure

Parameter-analysis facilities for the decision support system were developed using an object-oriented structure using the C++ language (Figure 6.11). This structure is more complex than that developed in the previous chapters, and warrants a more concise explanation.

The central object in this structure is the `TParameterAnalysisManager` class (henceforth known as the parameter analysis manager). This class contains all of the functionality to communicate with the decision support system and perform the task of generating response-surfaces and design charts. It contains pointer links to the `TObjectiveFunctionModule`, which is a repository for all of the available objective-function objects. Two graphical analyses have been developed for displaying and arranging contours and response-surfaces (`TSurfaceParameterAnalysis` and `TUserDefinedParameterAnalysis`). The `TSimulationEngine` class is linked to each objective-function and is not required by the parameter analysis manager.

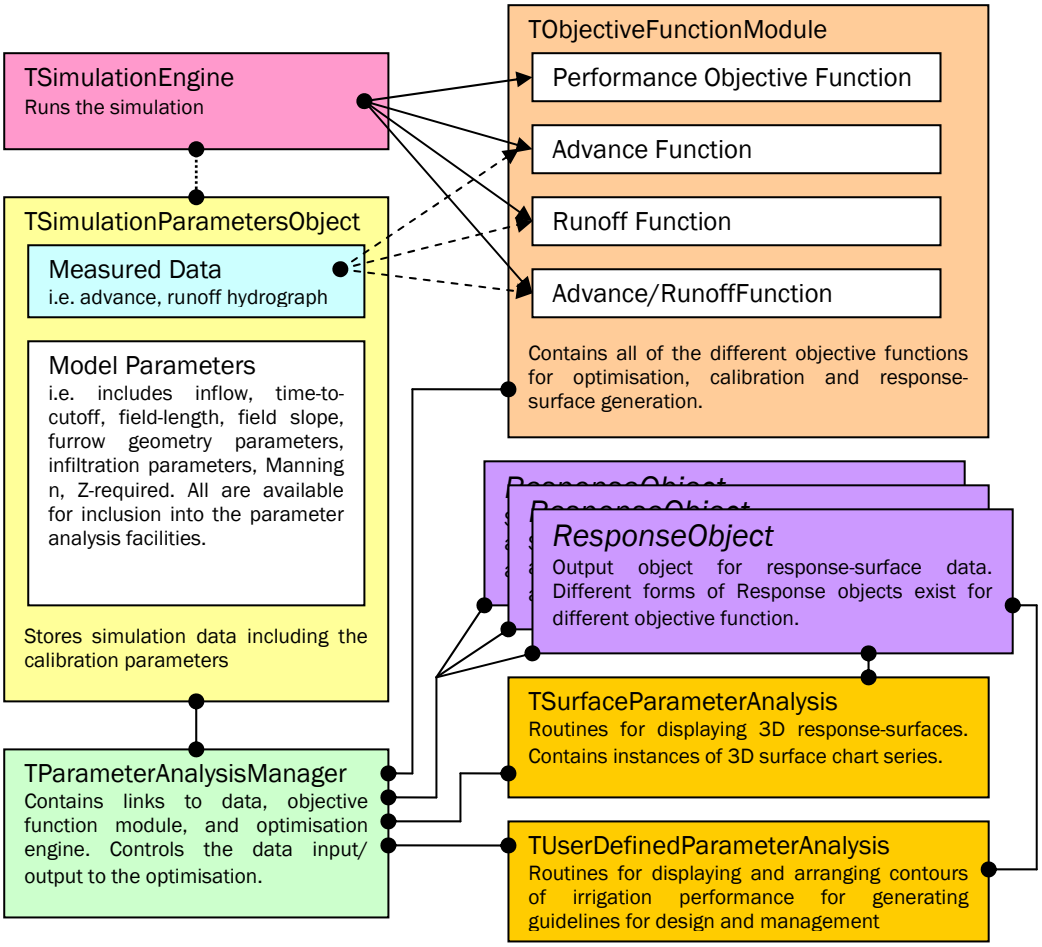


Figure 6.11: Object-oriented components for design and management guideline generation.

Integral parts of this system are the response objects (derived from `TCustomResponseObject`) that manage and store the system response data in n-dimensional arrays (`TPADataArrayObject`) corresponding to n decision variables (n<=3). Each response object can contain a range of these arrays to hold all of the system response information for the selected objective-function. There are two forms of these objects: those that store performance response information (`TPerformaceResponseObject`); and those that store calibration response information (derived from `TCustomCalibrationResponseObject`) of which there are several types for different calibration objective-functions. The parameter analysis manager is designed so that many of these response objects can be stored in memory simultaneously in order to compare outputs and accumulate information from different scenarios into a single set of design charts.

Figure 6.12 shows the class hierarchy of the response objects. The virtual base class `TCustomResponseObject` contains the functionality to add new response data, load and save the response data to and from a file, rearrange the order of the dimensions of the response data, as well as pointer linkages to the parameter analysis manager, objective-function module, selected objective-functions, and individual parameter components (to remap parameter configurations). It contains a special definition object called `T_PADefinition` which contains enumerated

types of all the elements of a parameter-analysis operation including objective-function type, infiltration type (high, low, average, measured), grid sizes, and selected parameter types.

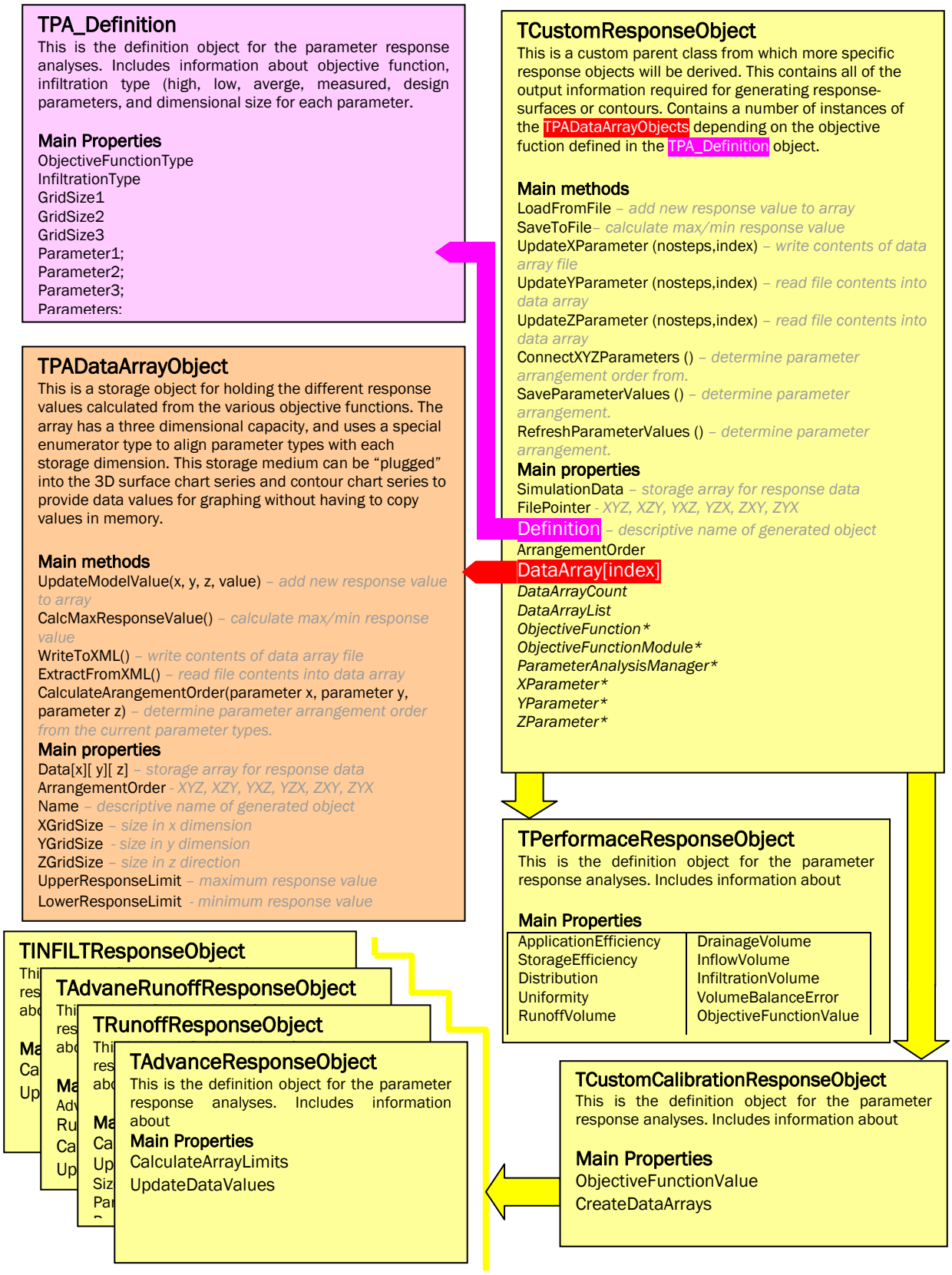


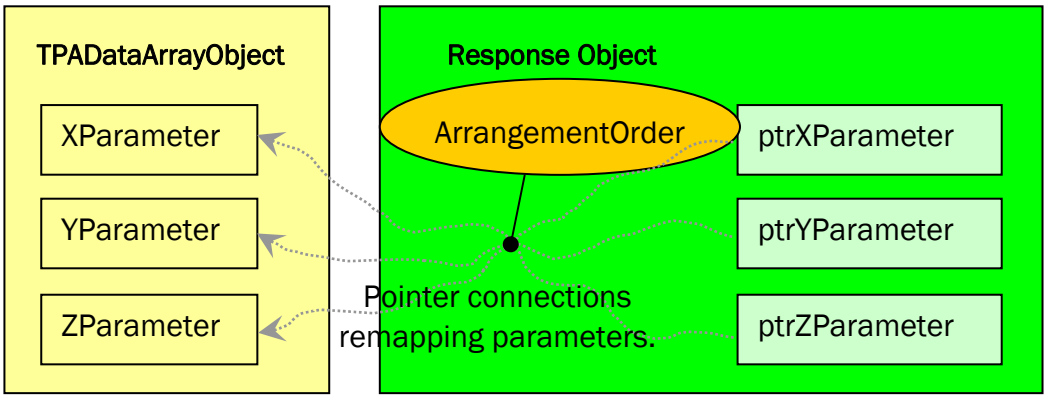
Figure 6.12: Storage and definition objects of Parameter Analysis Manager

Child classes derived from `TCustomResponseObject` contain the individual storage functionality associated with the linked objective-functions, as well as specialised functions for updating the storage arrays and calculating maximum and minimum response values.

The response object has been designed to work efficiently with other elements of the decision support system. The *n*-dimensional arrays in which the response information is stored are part of an especially designed class called `TPADataArrayObject`. The array has a three dimensional capacity, and uses a special enumerator types to align decision variable types with each storage dimension. This storage medium can be "plugged" into the 3D surface chart series and contour chart series to provide data values for graphing without having to copy values in memory.

The `TPADataArrayObject` class facilitates remapping of the parameter dimensions between the storage data array and response object through parameter-pointers located in the response object base class (Figure 6.13). This allows a switching of variables when instances of `TPADataArrayObject` class are "plugged" in the graphical outputs. For example, flowrate and time-to-cutoff could be interchanged providing a mirroring effect of the graphical response-surface. Alternatively, field-length could be interchanged with flowrate to switch from a management scenario to a design scenario.

Before any parameter analysis operation commences, a dialog is presented to the user to select the key elements of the study for the current selected record. This includes the decision variables (from the `TSimulationParametersObject` object), objective-function, and grid sizes for each decision variable. Pointer links are then established between the parameter analysis manager, selected decision variables and objective-function. Response-surface data is then generated by the manager and stored in the response object, through changing the design values and updating the objective-function. Progress is reported back to the parameter analysis manager, which passes this information back to the decision support system. When the response-surface data has been generated, the data is saved to disk and linked to the selected record.



Parameters dimensions are "virtually" rearranged (remapped) according to the ArrangementOrder variable. This is done using pointers to access different memory segments of the stored data matrix. This allows direct memory access to different configurations of the response data during surface and contour generation; e.g. Flowrate vs TimeToCuttof, Flowrate vs Fieldlength, Fieldlength vs TimeToCutoff.

Figure 6.13: Mapping of dimensions from the Response object to the data array object.

6.6 Analyses for displaying output

Two analyses have been developed for displaying the outputs from the parameter analysis manager. This includes a three-dimensional response-surface analysis (`TSurfaceParameterAnalysis`), and a two-dimensional design chart analysis (`TUserDefinedParameterAnalysis`).

6.6.1 Response-surfaces

The response-surface analysis (`TSurfaceParameterAnalysis`) is designed to maximise the visualisation of the system outputs in two, three or four dimensions. Visualising the fourth dimension is achieved through "chart-splitting" or through a slider-bar control of the third decision variable. The analysis is highly interactive with many interface components for changing surface features, coordinate systems, and parameter interactions. Key features of this analysis include:

- Interaction with a third decision variable using a slider-bar control;
- System response for different values of the third decision variable can be expanded into separate charts ("chart-splitting"), removing the interaction with the slider-bar control;
- Any of the decision variables can be interchanged with one another on the chart axes and slider-bar control.
- Synchronized zooming, panning and rotation capabilities for all charts. If the view of one chart is changed, then the other charts are also updated accordingly;
- Merging of response-surfaces into a single chart to visualise surface interactions;
- User-selectable outputs including the ability to incorporate results from more than one analysis;
- Parameter response filtering to hide/show different parts of the response-surface (for example, see Figure 5.9);
- Different colouring options including fixed colours, colour gradient, and colour palette, and also user-defined transparencies; and
- Different surface outputs including solid surface, contours, wire-frame and dot-points.

Sample outputs from the response-surface analysis have already been presented the Chapters 4 and 5 when validating the calibration and optimisation modules.

6.6.2 Guidelines for design and management

The user-configurable two-dimensional (contour) design-chart analysis (`TUserDefinedParameterAnalysis`) was developed to allow custom generation of design and management guidelines. Many of the features that were developed for the response-surface analysis have also been incorporated in this analysis including: the slider-bar control for the third parameter; parameter response filtering to hide/show different parts of the response-surface; and different colouring options for the outputs. Key features of this analysis, that differ from the response-surface analysis include:

- A user-definable chart array, whereby the user can configure the number of chart-rows and chart-columns for setting up the design analysis;
- A custom editor for assigning decision variables to individual chart axes;

- Drag and drop set-up of system-outputs onto the design charts;
- Multiple system outputs allowable for each chart, with filtering options available to improve readability; and
- Custom captioning of each chart.

Figure 6.14 shows a sample output from this analysis for three different infiltration conditions. The charts were developed in under five minutes through running three separate parameter analyses for each infiltration condition; setting up a 3x2 array of charts; assigning flowrate and time-to-cutoff variables to the individual chart axes; and dragging and dropping the "application efficiency" and "storage efficiency" outputs onto the charts.

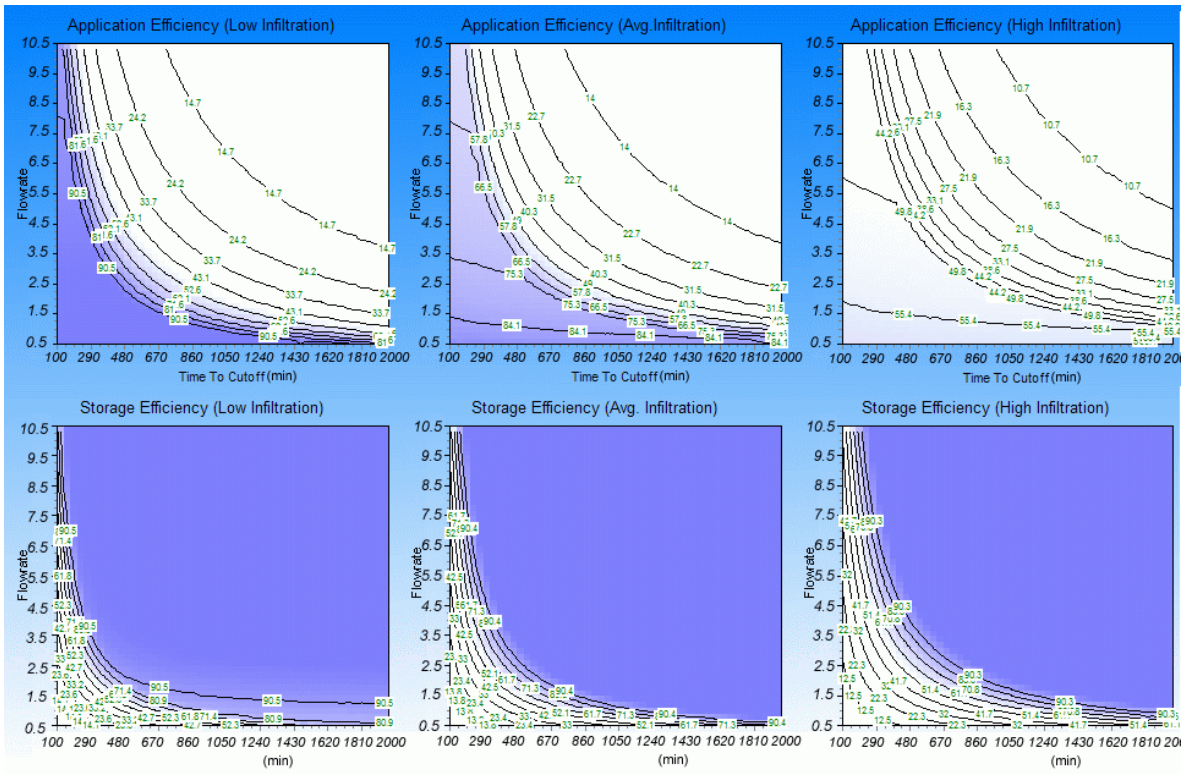


Figure 6.14: Sample design chart output from the parameter-analysis module.

In using these charts, the user would be advised to choose values of the decision variables that would correlate with the blue regions of the charts, which represent regions of high efficiency. Typical design selections would lie in the region of the parabolic intersection of the two outputs for the individual infiltration conditions. From these charts, the user can deduce that application efficiency decreases with increasing infiltration conditions, and that a maximum application efficiency of around only 50% can be achieved under the high infiltration state. These charts also show how easy it is to overwater the field, with large areas of peak storage efficiency evident.

6.7 Discussion of parameter-analysis facility

The usefulness of the parameter-analysis module has already been demonstrated through the generation of the response-surfaces presented in Chapters 4 and 5. This was a purely academic application of the tool, yet it only demonstrated a small

proportion of its capabilities. From a practical point of view, the tool offers the potential to quickly and easily generate a set of design and management guidelines given historical paddock infiltration data.

A potential limitation of the module is that it is not capable of generating curves in the form presented in the preliminary case study. That is, the tool cannot represent system outputs by the chart axes, but instead must display them as contours with the decision variables represented by the axes. It is possible to include this capability in a later version of the software, and even utilise the optimisation algorithm to generate system outputs as was done manually in the case study. However, having studied these output formats, it was decided that the most useful method is that which has been developed here, with the influencing factor being the need to visualise as much as the system response as possible.

6.8 Conclusions

A preliminary study was carried out to show the importance of infiltration variation on irrigation decision-making, and to provide a first attempt at generating design charts. The resulting charts combined the effects of variable infiltration and three decision variables using a fixed management strategy of minimising runoff. A limited range of response outputs for a fixed management objective negated the potential benefit of visualising a large number of dimensions. Nevertheless, this study provided the basis for the subsequent development of the guideline-generating capabilities for the *FIDO* decision support system.

Recommendations from the study included representing system outputs as contours and iso-curves, rather than by the chart axes; representing different infiltration conditions in separate design charts; allowing the user to assign variables to each chart axis; and representing only two decision variables in each chart. These features were then incorporated into the parameter-analysis module using a detailed object-oriented structure to combine the key elements of a manager-controller class, design parameters, field measurements, simulation engine, objective-functions, and two highly interactive graphical analyses.

Chapter 7 Software engineering a decision support system for furrow and border irrigation

7.1 Introduction

The literature review (Chapter 2) has highlighted the usability and reliability problems with existing surface irrigation design and management software. The research presented in the subsequent chapters has attempted to overcome these problems by developing new object-oriented tools for furrow and border irrigation design and management tasks. The goal of this chapter is to combine these tools with a database and a simple user interface to develop a new decision support system for furrow and border irrigation called *FIDO* (Furrow Irrigation Decision Optimiser).

The research in this chapter has four main objectives: *(1)* it will discuss the software engineering design options available while developing the application; *(2)* a suitable object-oriented program structure is developed to accommodate program elements; *(3)* it will develop a new *XML*-based surface irrigation database; and *(4)* a simple graphical user interface is created using advanced third-party libraries.

This chapter is accompanied by seven appendices providing further information on the software engineering tools used to develop the decision support system (Appendix 7.1), the *XML*-based data structures (Appendix 7.2), and the evolution of the *FIDO* simulation (Appendix 7.3), calibration (Appendix 7.4), optimisation (Appendix 7.5), parameter analysis (Appendix 7.6), and database (Appendix 7.7) graphical user interfaces.

7.2 Decision support system design criteria

Before developing the decision support system, the operational functionality of the system needs to be clearly defined, the objectives of the development identified, and the software engineering tools presented. Each of these will now be discussed in turn.

7.2.1 What is the *FIDO* decision support system

The *FIDO* decision support system is a computer software tool combining a database, graphical user interface and surface irrigation design and management tools into a single program. The required functionality of the software is to:

- store property, paddock, event and model information;
- evaluate the performance of existing irrigations;
- estimate infiltration parameters from irrigation advance and/or runoff data;
- assess the optimum performance potential of existing irrigations;
- review and compare performance, infiltration and optimum potential over time;

- evaluate performance sensitivity to changes in design and management parameters and options; and
- create site-specific design charts for irrigation design and management.

7.2.2 Objectives of decision support system development

The primary goal of this chapter is to develop a decision support system for furrow and border irrigation by combining the tools developed in Chapters 3 to 6 into a single program containing a database and graphical user interface. This involves:

1. **developing an object-oriented structure to accommodate program elements**: This structure must separate the graphical user interface components from other task related objects while sharing resources and communicating effectively between elements. It must remain flexible for future development.
2. **developing a database to store property, paddock, event and model information**: The database is required to store measurements, modelled results, and other irrigation information. Average irrigation and infiltration performance at the property, paddock and event levels needs to be monitored and automatically recalculated each time model results are updated. The database should use a format accessible to other applications while existing irrigation data formats (such as *SIRMOD* and *INFILT* data files) should be easily imported.
3. **creating a user-friendly graphical user interface**: The interface must be simple and intuitive to use and not distract the user from the tasks of irrigation evaluation, design and management. Therefore, recognized principles of graphical user interface design need to be adhered to. It should utilise proven third-party components and libraries to maximise programming efficiency and power.

The target audience of this tool is primarily researchers, and irrigation consultants. However, this is not trying to limit this software away from users at the farm level.

The scope of this chapter is limited to presenting a decision support system for furrow and border irrigation capable of demonstrating the components developed in Chapters 3 to 6. The goal is not to develop a commercial quality software product ready for distribution within its target market. This would require a further number of months of development, testing and bug fixing. However, the structure, database and interface as presented in this chapter are valid contributions to this eventual ambition.

7.2.3 Software engineering tools

FIDO has been developed in C++ as a *Win32* application under *Microsoft Windows* using *Borland C++ Builder* and the *Visual Class Library* (www.borland.com). *XML* (eXtensible Mark-up Language), *XSD Schema*, and *XSLT* technologies (www.w3.org/XML) were chosen to develop the database components of the software while prototyping was undertaken using *XML Spy Suite* software (www.altova.com). Several third party libraries including *TeeChart* (www.teemach.com) and *VirtualTreeView* (www.delphi-gems/VirtualTreeview)

feature prominently throughout the *FIDO* decision support system. A description and discussion of these software engineering tools is presented in Appendix 7.1.

7.3 Program framework

FIDO has been designed using a multi-threaded object-oriented design structure that separates the graphical user interface code from the separate but interrelated tasks of data storage, simulation, calibration, optimisation, and parameter analysis. Task and storage orientated objects called "managers" have been developed for each of these elements encapsulating their features within. Special analyses have been developed to post-process and display outputs from the managers. Tools such as the simulation and optimisation engines remain external to these managers and analyses, but are easily accessible to each.

7.3.1 Design methodology

Developing a framework for the decision support system has centred on the primary goal that the graphical user interface remains external to the mathematical libraries and databases. In this manner, the decision support system can be compiled as a library *sans* user interface and distributed to other researchers and developers for inclusion in their projects. This has already been applied in undergraduate and postgraduate projects (Ma 1994; Gillies, M. *pers.comm.* 2005-2006).

Separating the user interface from the rest of the program is contrary to the natural programming doctrine of using a "Rapid Application Development (RAD)" tool such as *Borland C++ Builder*. These RAD environments encourage "form-based" programming where the simplest path for developers is to encapsulate non-interface code inside the automatically created form-classes. The developer creates user-interface components and controls graphically at design time that are then instantly accessible to the non-interface code. While this is powerful when creating smaller applications, large decision support systems such as *FIDO* require careful code and object management to simplify program structure and reduce memory loadings. Externalising the interface plays a large role in this simplification.

Because of the size of the *FIDO* project (consisting of over two hundred individual object types contained in over one hundred and fifty ".cpp" files), early development consisted of creating five separate programs to focus upon the separate tasks of data-management, simulation, calibration, optimisation and parameter analysis. The main purpose of this was to increase development efficiency through faster compile and linking times. However, it also had the added advantage of improving the overall program structure through better object design and management. That is, the program objects have been designed better as a result of having to work under a range of operating environments. Having the system work in the absence of a user interface has further enhanced this.

7.3.2 Structural components

The structure of *FIDO* can be broken down into five conceptual element groups of "user interface units", "managers", "tools", "analyses" and the "project". These can be further classified into their graphical user interface and program code components (Figure 7.1).

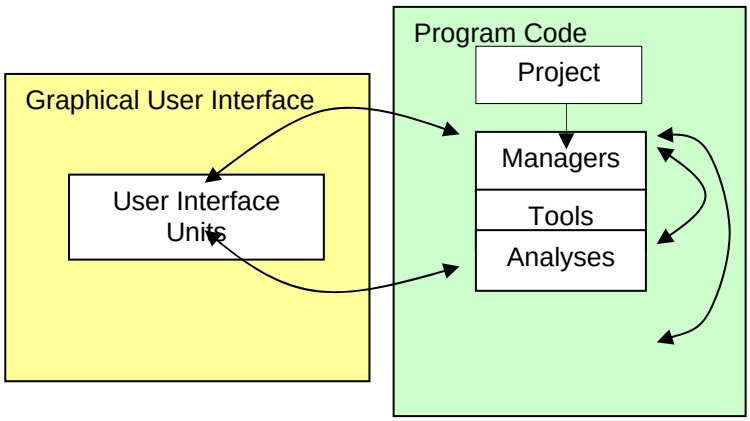


Figure 7.1: Conceptual view of *FIDO* program elements showing separation between graphical user interface and other program code. Arrows represent communication lines between objects.

User interface units

The user interface units consist of forms and dialogs containing a range of graphical controls. While the appearance and content of these elements will be discussed in detail later in the Chapter, it's the structure and order of the units that is examined here.

All of the main interface units are derived from the `TForm` class (the "form" component in *Borland's Visual Class Library*) and represent a visual "window" or form inside of *FIDO*. The key form is `T_MainForm` that is the first unit created when *FIDO* is started, and the last to be deleted upon closing down. `T_MainForm` ultimately contains and controls instances of all other interface elements with the responsibility of displaying, resizing and positioning them. Other form elements embed seamlessly into `T_MainForm` so that it appears that only one window exists.

Figure 7.2 highlights the relationships between `T_MainForm` and the other main interface units, forming three embedded window layers. `T_MainForm` constitutes the first window layer. The second layer consists of two possible input-related forms (`T_RecordSelectorForm` and `T_ParameterInputForm`) and two possible output-related forms (`T_DatabaseMainForm` and `T_AnalysesForm`). A third window layer exists for database manipulation where a further three possible output-related forms are available (`T_DatabaseRecordsForm`, `T_ReportForm`, and `T_BrowserOutputsForm`). Two of these forms, `T_BrowserOutputsForm` and `T_AnalysesForm`, are similar in structure in that they both contain specialised controls to provide the graphical interface for the "analysis objects".

Finally, there are also three dialogs available (`T_RuntimeDialog`, `T_ParameterAnalysisDialog`, and `T_ChartEditorDialog`) which appear to float above the main window upon activation.

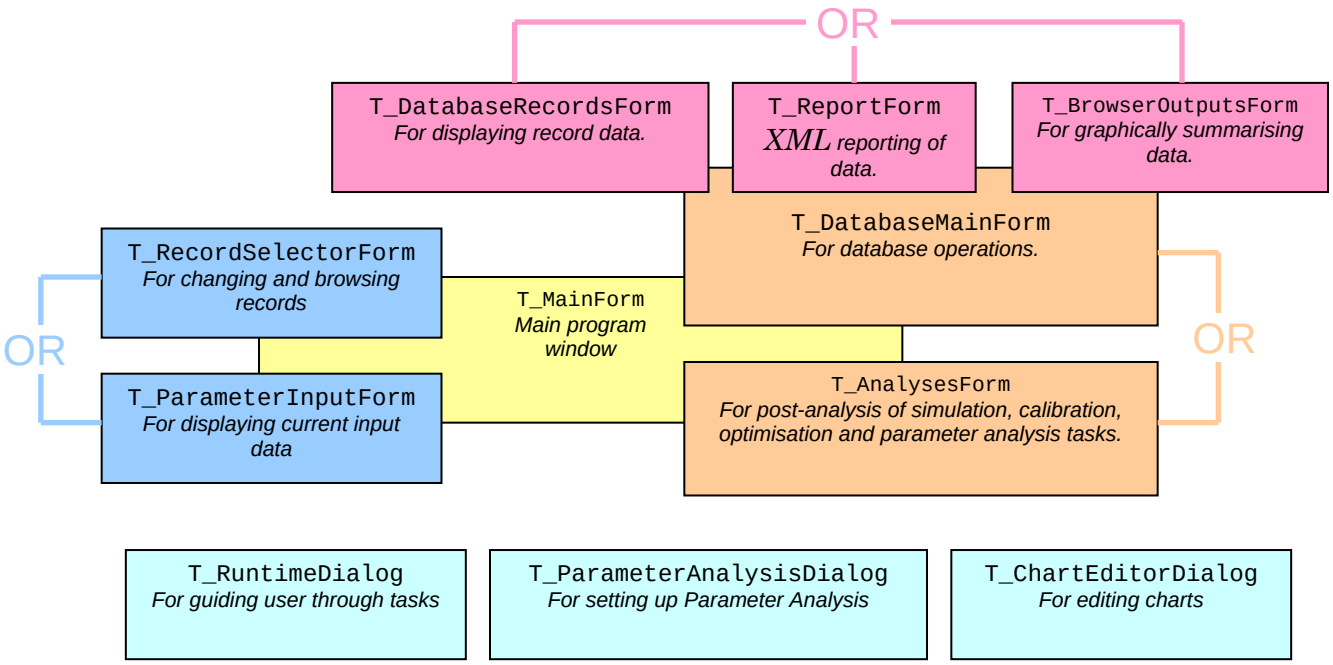


Figure 7.2: Main user-interface units in *FIDO* showing relative positions and parent objects. Layers designate parent/child relationships and "OR" symbol suggests that either one element or another will be shown depending on current program conditions.

`T_MainForm` also contains non-interface elements such as the "Program Manager" (see next section). While this may appear to contravene the original design goal of separation of interface and non-interface code, in this instance, it is only an object which is contained and not fragments of code. Any developer who uses *FIDO* as an external library can easily relocate this object.

Managers

"Managers" are the workhorses in the *FIDO* program containing the execution loops for generating output and performing the decision support tasks. Three different types of managers are used in *FIDO* (Figure 7.3) and include the program manager, storage managers, and task managers:

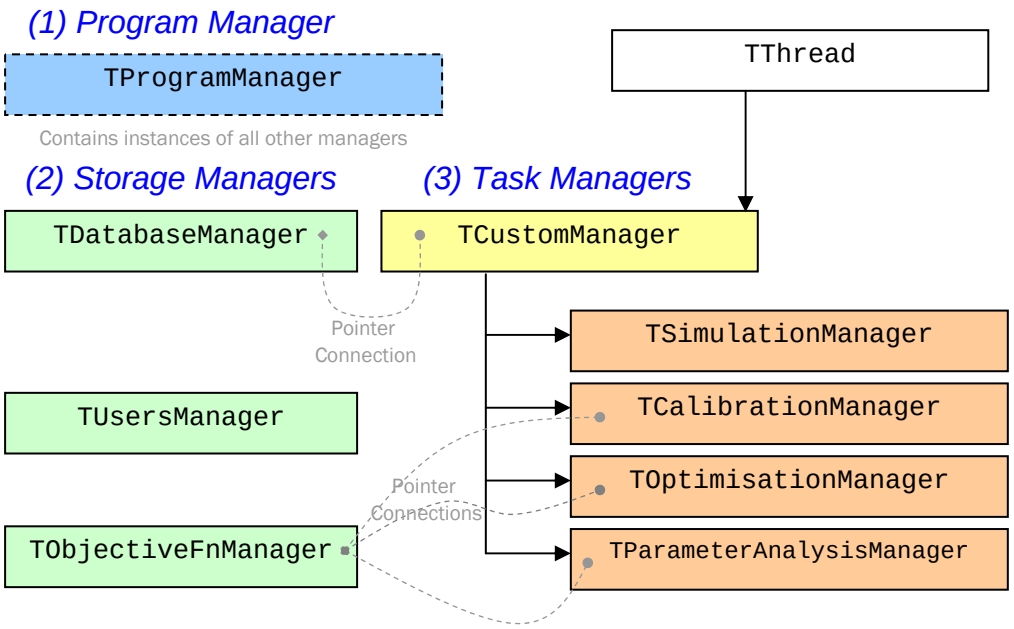


Figure 7.3: Derivation of manager objects used in *FIDO*.

1) **The program manager (`TProgramManager`) is designed primarily as the execution point and communication centre for all decision support operations.** All major non-interface components are created and contained inside of this manager including the other managers and tools. This effectively unifies the program code keeping it separate from the user interface. It is through this manager that the user interface units communicate with the task managers and analyses responsible for decision support operations.
2) **Storage-managers are responsible for grouping related data or methods.** This manager-type is conceptual since existing examples are not derived from a common ancestor, although they still have a similar purpose. Three independent examples exist in *FIDO*:
   - `TDatabaseManager` contains several graphical analyses for summarising the stored data. This includes an *XML*-based report generation feature to display a user-customisable review of the selected records. Historical performance and infiltration analyses are also available along with the ability to explore measured input data.
   - `TUsersManager` contains a database of user information. This information is used to track changes to data-variables, and in report generation.
   - `TObjectiveFnManager` is used to centrally store the different objective-function objects used in the calibration, optimisation and parameter analysis.
3) **Task-managers are designed to perform particular actions or tasks such as running a simulation or calibration.** *FIDO* contains several of these managers and these are derived from a common parent called `TCustomManager`. This base class is itself derived from an independent operating thread class (`TThread`) so that mathematical processes do not monopolise processor time. This multi-threading capability allows the user to interact with the graphical user interface while simulations and optimisations are being performed. Task Managers include:

- `TSimulationManager` for irrigation evaluation tasks;
- `TCalibrationManager` for calibration and parameter estimations;
- `TOptimisationManager` for optimisation of irrigation performance; and
- `TParameterAnalysisManager` for in-depth parameter evaluation.

Analyses

Output from the different managers is sent to the "analysis" objects for post-processing and display. It is through these analyses that detailed graphical and textural outputs are presented to the user. An example of this is the animation of water flowing down a furrow developed in `TAnalysisSIM_FlowAnimation` (see Figure 7.18 in Section 7.5.6)

All analysis objects are ultimately derived from a specially created class called `TCustomAnalysis` (Figure 7.4), which has been designed to "plug-in" to the user interface units `T_AnalysesForm` and `T_BrowserOutputsForm` containing the graphing and spreadsheeting functionality. The base class `TCustomAnalysis` is itself derived from an independent operating thread class (`TThread`) to extend the program's multithreading capabilities.

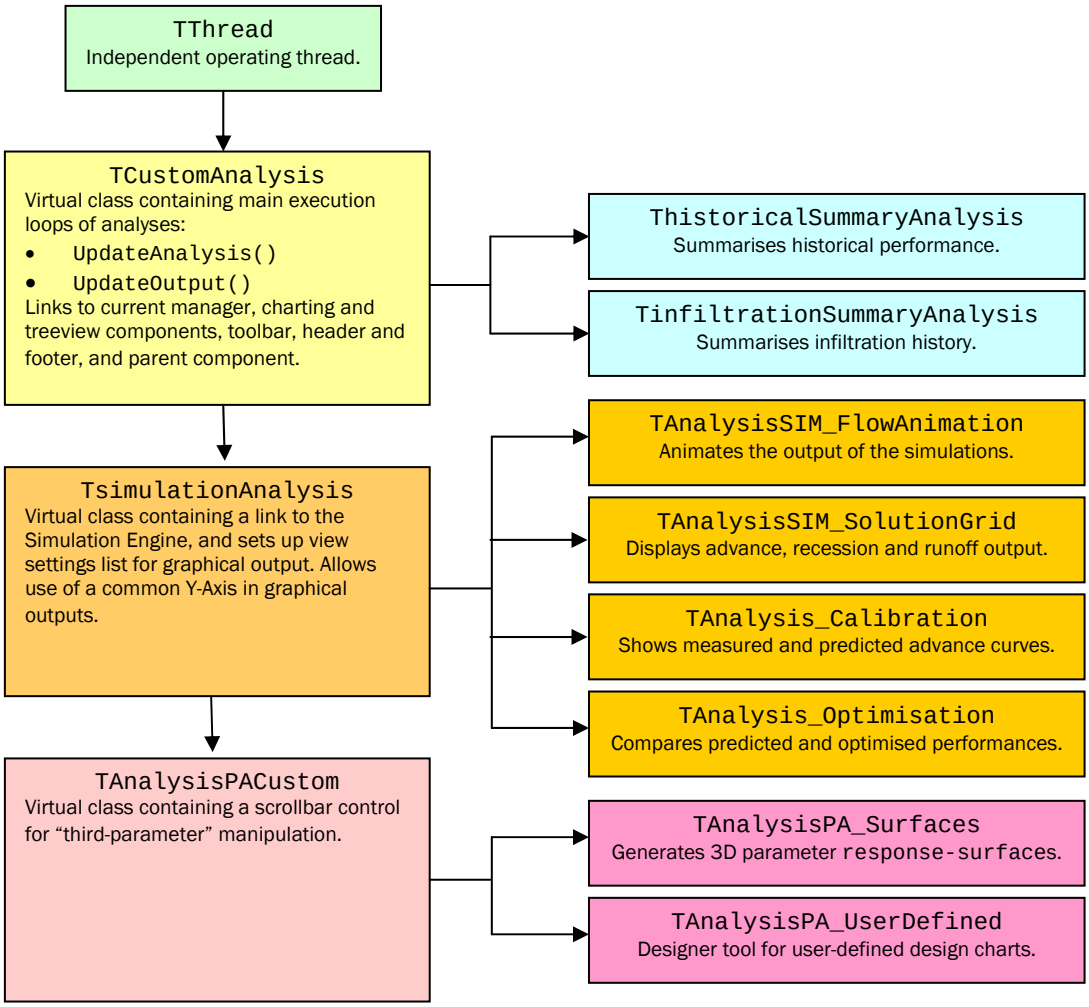


Figure 7.4: Derivation of analysis classes in *FIDO*.

Further abstraction of the base class continues through the development of the specialised virtual classes; `TSimulationAnalysis` which adds extra common functionality for simulation, calibration and optimisation operations; and `TAnalysisPACustom` which contains information common to the "parameter analysis" tasks.

In all, *FIDO* currently (at the time of writing) contains eight analyses, although more can easily be developed and added at a later time. These current analyses are created and stored in their corresponding task-managers and include:

- two database analyses including `THistoricalSummaryAnalysis` and `TInfiltrationSummaryAnalysis` for summarising historical performance and infiltration;
- two simulation analyses including `TAnalysisSIM_FlowAnimation` and `TAnalysisSIM_SolutionGrid` for graphically animating the simulation outputs and presenting advance, recession and runoff information;
- one analysis called `TAnalysis_Calibration` for displaying calibration "fits" of measured and predicted advance;
- one analysis called `TAnalysis_Optimisation` for viewing optimisation progress; and
- two analyses for detailed parameter analysis of response-surfaces and curves: `TAnalysisPA_Surfaces` and `TAnalysisPA_UserDefined`.

Tools

FIDO contains two main tools which are available to the task managers: the simulation engine, (`TSimulationEngine` as developed in Chapter 3); and the optimisation engine (`TOptimisationEngine` as developed in Chapter 4). These tools are created inside the program manager (`TProgramManager`) so that only one instance of each is required, so as to be accessible to all managers.

Project

The project object (`TFIDOProjectTreeObject`) is the main database control in *FIDO*. It contains routines to load and save data, and is responsible for communicating with the program manager (`TProgramManager`) to pass information in and out of the task managers and analyses.

7.3.3 Structural connections

Although the "user interface units", "managers", "tools" and "analyses" are separate objects, they are not designed to function independently. Instead they coexist with each other through a network of external links and parent-child relationships.

Figure 7.5 demonstrates these relationships for the main program elements. This shows how the program manager (`TProgramManager`) is located inside of `T_MainForm`, and that all other managers and tools are located inside of `TProgramManager`. While the database manager (`TDatabaseManager`) has no parent/child relationship with any of the task managers, connections still exist

between them through external links. Likewise, the objective-function manager (`TObjectiveFnManager`) is linked to three of the task managers using the same mechanism.

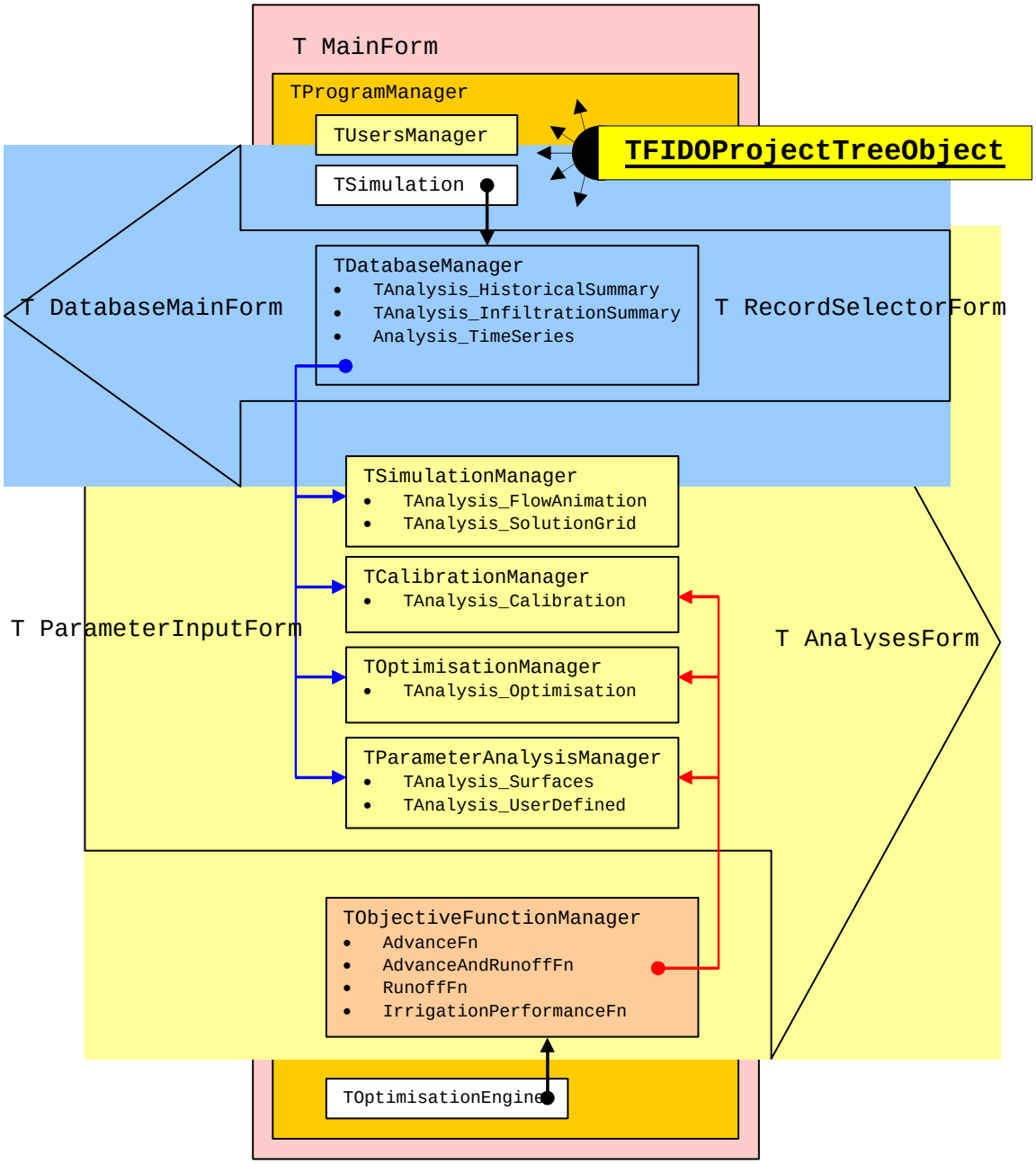


Figure 7.5: *FIDO* structure demonstrating interactions and connections between the central "user interface units", "managers", "tools" and "analysis" components. The project object is visible to all components.

"Pointers" (programming term for memory referencing) are used extensively throughout *FIDO* as the mechanism for these links. Using pointers establishes visibility between the non-related objects that extend in one direction through related links. For example, through pointer connectivity, the simulation engine (`TSimulationEngine`) is accessible to the database manager (`TDatabaseManager`), which in turn is linked to the other task managers. The

task managers then automatically have access to the `TSimulationEngine` as they are sharing the link with `TDatabaseManager`. However, the reverse is not true since `TSimulationEngine` does not know anything about the database manager or the task managers. This is because the reverse link has not been defined, and in practice it is not required.

Figure 7.5 also shows how the analysis objects are stored in their respective storage and task managers. The large transparent arrows that underlay these objects indicate how the managers and analyses interact with the different input and output user interface units. Note that different interfaces are used for database and non-database related tasks.

7.4 Developing a surface irrigation database

The *FIDO* database component (project object) is a key component of the decision support system from which all tools receive and send information. It has been developed using *XML* and *XML-Schema* technologies, and is based upon a four-tier hierarchical structure of property, paddock, event and model information. Data is not stored in flat file or relational tables as in traditional database design (although this approach was used in early development), but through parent-child relationships between the four structural elements. The primary element is the property data, which is stored as independent *XML* data files containing all information relating to the property, including the paddock, event and model data. The project object (`TFIDOProjectTreeObject`) serves to link all of the property data and facilitate searching and retrieving requirements.

Developing the database was a complex process. The design methodology was evolutionary with many prototypes being developed using different structures and technologies. Nevertheless, having learnt from earlier "failures", the current *XML* database has been designed and developed in an organised manner. This included the steps of identifying design issues, designing schema representations of the data, establishing database connections, designing object-oriented computer code for the database, and establishing an efficient development methodology. These issues will now be discussed in turn.

7.4.1 Design considerations

The current database has evolved over many years and through several changes in structure and technologies. In many ways, its design has been dominated by available technologies with early versions using the *Microsoft Access* database and *Borland Database Engine*. After many attempts, it was decided that the flat file and relational database methodologies were not well suited to the progressive disclosure objectives of the software design. That is, the primary purpose of this database is not the traditional store/search/filter methodology of large data repositories. Rather its purpose is to embed itself into the object-oriented framework of the decision support system, with traditional tables and databases being replaced by intelligent objects that help form the program structure. In the current version, *XML* and *XML Schema* technologies are used

to facilitate these requirements, serving as communication and storage medium for disk access, and integration with external software and libraries.

One implication of this is that the database is not "live". That is, the data structure and values are loaded into memory when the user opens a project, which initiates a copy of the *XML* data into program memory. The original stored data remains unchanged until the user chooses to save the program data back over the original database. Forgetting to save the database will mean that all changes to the program data are lost. In comparison, the early attempts at using *Microsoft Access* database meant that data was constantly being read and rewritten to the database.

XML has many attractive features for database development including a standardised format allowing the data to be accessed by other applications and platforms. Saving the data in the *XML* format serves to preserve the structural relationship of the data as well as the data values. There is also little overhead in using *XML*, compared to having one of the commercial database engines installed on the user's computer. However, one disadvantage of using *XML* as a data store, is that the entire database structure is loaded into memory during initialisation[13]. However, given that the data requirements of a large irrigation database would only be in the order of megabytes (easily handled by today's computers), this should not be a problem.

Nevertheless, careful memory management was seen as a design objective to ease the burden on the system. *FIDO* ensures that while the database structure is always loaded in memory, the property data (other than the names and IDs and filenames) are not loaded into memory until required, where they will stay until the program is terminated. This is made possible by storing the property details separately from the database structure and using dynamic memory allocation techniques to assign memory for storage variables only when required. Both the record files and the database structure are stored in *XML* format.

In an early design using the *XML* technologies, relational tables for each record type were used to form the *FIDO* database with each set of data being saved in their own files. That is, each property, paddock, event and model data record was saved in its own individual file and stored in four separate data tables. The project file maintained data connections between the elements. From a practical point of view, this proved to be too complex with a typical database containing hundreds of *XML* data files. Backing up data and exchanging data with other users are important design considerations, and this was difficult to undertake with so many files in different directories. The software engineering required to manage the databases and import files was also complicated. Therefore, the idea was abandoned for the single property file format that is simple to copy and exchange.

7.4.2 Schema representation of the data.

Appendix 7.2 presents the schema representations of the *FIDO* database elements including property, paddock, event and model data. *XML*-based

[13] There are many exceptions to this.

property records represent the top level of these schemas, and are saved in their own separate property files. Paddock data, event data, and model data are then located inside each property file, and are pre-linked together by the *XML* structure. Each property record is then linked into the *FIDO* database by an *XML*–based project file that stores each record in a list.

In developing the schemas, common group data elements were identified to exist for all of the record types; a feature that will later be used in the programming implementation. These common data elements include:

- Identification information;
- Table specific data;
- Linked documents;
- Images;
- Performance summary;
- Infiltration summary; and
- Tally Summary (counts).

Specific information relating to each record type will now be discussed.

Property data

The property database has the responsibility of storing the property related information such as owner details and property location (Figure A7.2.1). This data type represents the highest level in the *FIDO* database structure. Summary statistics are recorded for child (paddock) records, however these results are processed no further as summarising across properties has no practical benefit.

Paddock data

The paddock database holds key paddock information (Figure A7.2.2) including basic field measurements for "auto-insertion" into newly created model records. This data type represents the second level of the *FIDO* database structure.

Event data

The event database holds information specific to the day on which an irrigation occurred including climate data and management options (Figure A7.2.3). Although this record is intended to represent an actual irrigation event, it is not designed to store any simulation model input data. This is because more than one set of input data can exist for an irrigation depending on how many furrows were monitored. Therefore, storage of this data was left to the model data type.

Model data

The model data type is the most complicated of the irrigation records (Figure A7.2.4). The "data" component of the record contains the model parameters required to run the simulations (which were discussed in Chapter 3.3.4). The furrow parameters may be stored in either physical or empirical form (included for input of furrow profile-meter results).

There are also some structural differences between this record type and the rest including:

- irrigation performance is represented by only one results-field instead of three. That is, we cannot calculate high, low and average performance from a single set of simulation results;

- calibration and optimisations results are stored here; and
- no tally or infiltration summary fields are included.

7.4.3 Database connections

The *FIDO* project object (`TFIDOProjectTreeobject`) is essentially a database object, which links all of the property data, and facilitates searching and retrieving requirements. Only the property filenames are initially stored in the project object until the user requests more information. The connections between the different properties are made externally through a project file (in *XML* format) that contains the entire database structure. Figure 7.6 represents the schema structure of this *XML* file showing how the record name and filename are the only information stored along with the structural information. The "0...∞" symbol appearing in the figure indicates that there is no limit on the number of property records that the database may contain.

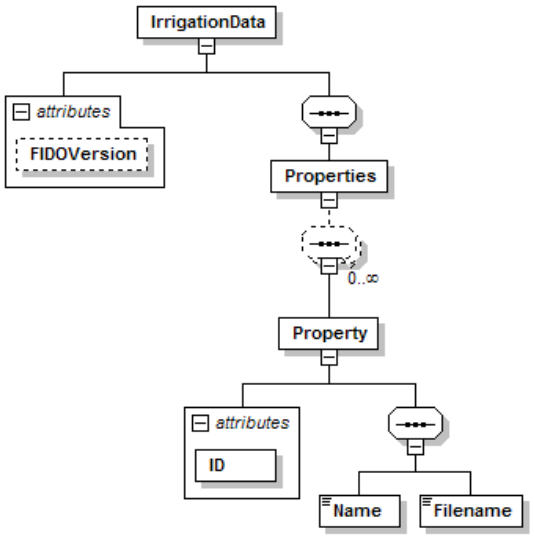


Figure 7.6: Schema representation of main *FIDO* database connections.

7.4.4 Programming implementation of the database.

The programming implementation of the *FIDO* database uses an object-oriented structure closely resembling the schema structure presented in section 7.4.2. It consists of a project object (`TFIDOProjectTreeObject`), a series of record objects for property, paddock, event and model data (derived from `TFIDOCustomDataTreeObject`), and many parameter objects for storing data-field values. These will now be discussed in turn

Project object

First presented in Section 7.3.2, the project object (`TFIDOProjectTreeObject`) contains a list of property objects, as well as temporary lists for active (selected) property, paddock, events and model data records. It contains routines for reading and writing to the XML files, and standard database functionality for adding, deleting, and modifying records.

Record objects

A series of record objects for property, paddock, event and model data have been derived from a common base class (`TFIDOCustomDataTreeObject`). Figure 7.7 displays the type and functionality of these object-classes and their common ancestors.

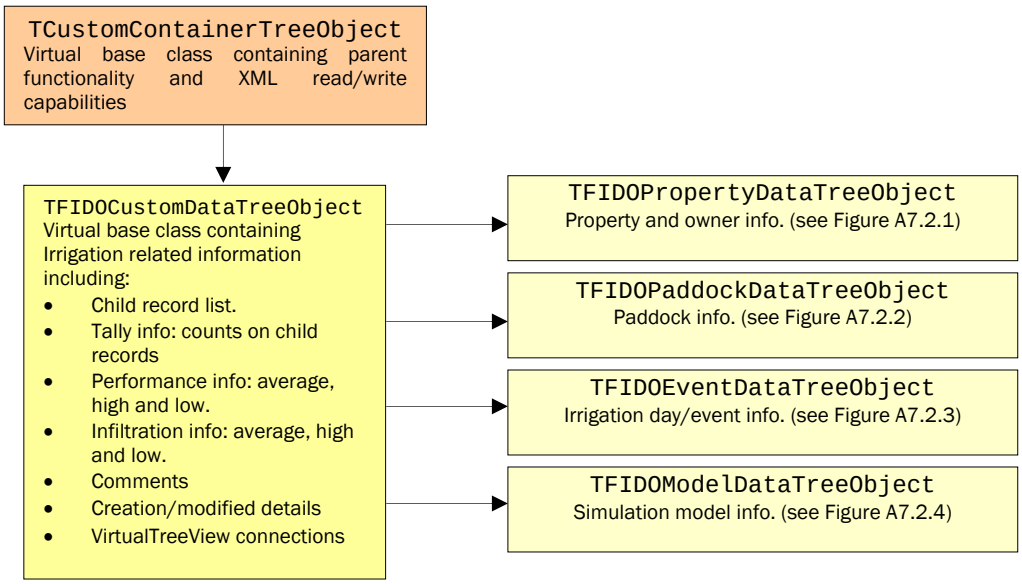


Figure 7.7: Derivation of record object classes in *FIDO*.

The structural organisational levels of the database are maintained through lists contained in each record object (defined in `TFIDOCustomDataTreeObject`). That is, paddock objects are stored in lists located in each property object, event objects are stored in lists located in each paddock object, and so on. These lists are updated during loading of the *XML* data files, and through modifications initiated by the program manager.

The `TFIDOCustomDataTreeObject` also defines statistical storage types for tally, performance and infiltration summaries, which are required on all database levels. Because these summary components are common to each child class, statistical operations can be called recursively and efficiently, updating each level in succession (Figure 7.8). For example, a property-object can request that its infiltration statistics be updated. This will spawn a cycle whereby each paddock-object listed for that property will request a similar action. Then every event-object listed for each paddock will again repeat the request. When the request filters down to the model data objects, the required information will then be passed backwards and be summarised at each level.

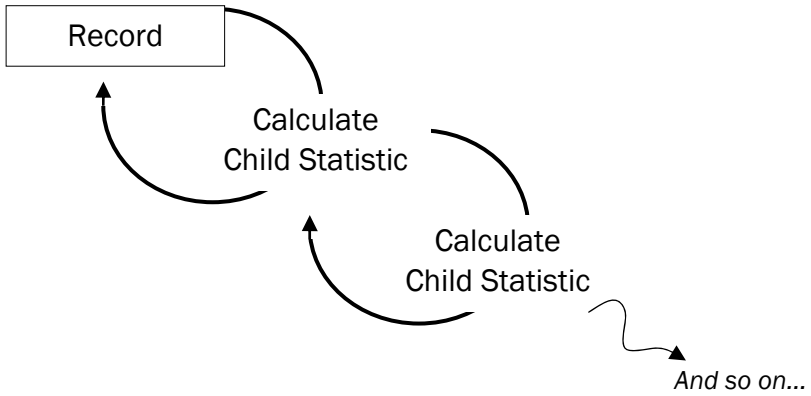


Figure 7.8: Sequence of summary calculations performed by `TFIDOCustomDataTreeObject` children.

Parameter objects

Powerful parameter objects have been developed to store and manipulate data-field information. These objects have the ability to:

- Display themselves in the *Virtual TreeView* control (see Appendix 7.1).
- Load/save themselves to disk using *XML* format
- Send themselves to *XML* Report
- Allow the user to edit them in the tree
- Allow comments to be added and stored
- Store “author” and “last-modified” information
- Allow user filtering of data.

Over thirty different parameter object types are used in *FIDO*. Figure 7.9 shows the design hierarchy of those relating to the data-field information.

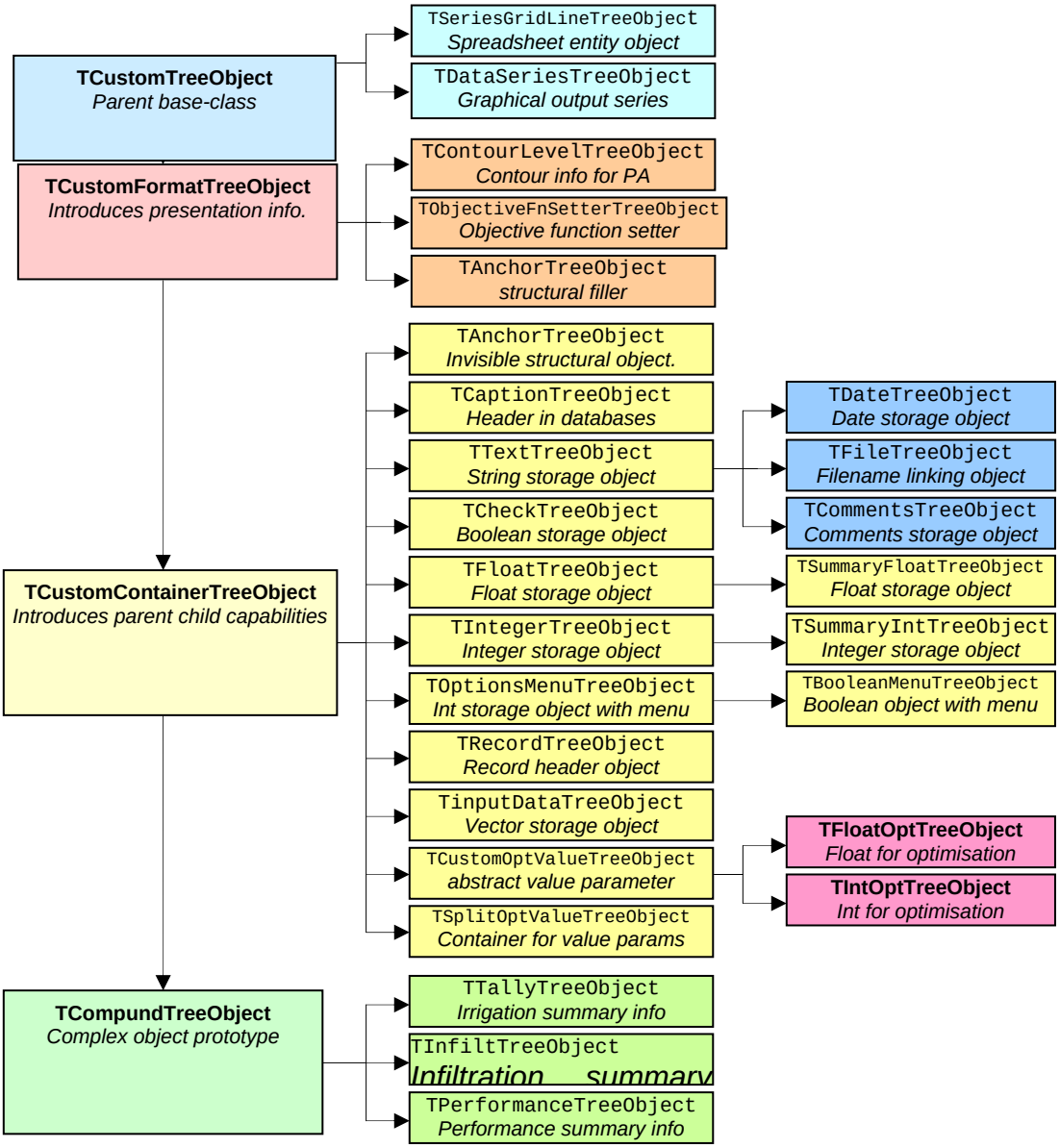


Figure 7.9: Parameter object design hierarchy as used in *FIDO*.

7.4.5 Database development methodology

Defining the structures of the property, paddock, event and model elements was a complicated process, and the final designs are the result of continual refinement and evolution over a period of months. The *XMLSpy Suite* software was indispensable for this purpose as it quickly and easily facilitated the synchronisation between the *XML Schemas* and *C++* record classes.

This involved a three step process of: (a) designing/modifying an *XML-Schema* for each database element using the *XMLSpy Suite* software; then (b) writing the equivalent *C++* code to generate "sample" output *XML* files; and then (c) validating the these files against each schema until the outputs and design structures agree. The *C++* code is in the form of specialised database classes or

"record and database objects" that contains the storage variables, and also methods to input and output the desired *XML* file.

7.5 Graphical user interface

Modern design conventions have been used to develop the graphical user interface for the *FIDO* decision support system. Lack of adherence to these conventions in the past has arguably been one of the leading contributors to the poor adoption rates of surface irrigation design and management software. Frustrations born out of trying to use a poorly designed interface do nothing to foster confidence in the effectiveness of the software and the complexities inherent in these types of mathematical models have traditionally manifested themselves in the interfaces.

The concept of the *FIDO* decision support system is complex, with its five main functional requirements and a large range of input and output parameters. Hiding this complexity from the user is the one of the primary goals of the design of the graphical user interface, which acts as the communication medium between human and computer. The effectiveness of the interface can be judged by how simply and intuitively the user can perform the tasks of surface irrigation database management, simulation, calibration, optimisation and parameter analysis. Therefore careful consideration has been given towards user issues and software design procedures in developing the *FIDO* graphical user interface.

7.5.1 Principles of graphical user interface design

Numerous texts and articles exist defining the principles of graphical user interface design (Tognazzini 1992; Cooper and Reimann 2003, Tidwell 2005). Although software technology is quickly outdated these days, the general principles behind graphical user interface design remain the same. Ten commonly used principles that have been adhered to in the design of the *FIDO* user interface include:

1. **Keep it simple and transparent**: Allow the user to concentrate on the task and not be distracted by the interface. Design the interface to show only useful and relevant information, and hide elements that compete with this. Don't clutter the interface and overload the user with too many buttons and options. Use "progressive disclosure" techniques to limit what the user sees at any given moment.
2. **Maintain consistency with appearance and behaviour**: The application should be consistent with itself and with other applications. Consistency allows the user to apply their existing knowledge of other applications and environments to understand the new application. This includes using common commands, controls and layout.
3. **Provide appropriate user feedback**: Maintain a sensible level of feedback to the user to keep them informed of the current program status. For each action a user makes, provide feedback that the system has received input and is operating on it. Feedback can include text or status bar messages, cursor changes, progress indicators, simple animations, audio alerts, and sometimes popup messages.
4. **Design the application to be self-evident:** Comprehensive online help and manuals are automatic requirements for any application, but the program

should be designed so that the user need not use them. The goal is to design an interface that requires no further explanation through careful use of labels, buttons, controls, hints and messages.

5. **View warning and errors messages as potential flaws in the application**: Users dislike being told something is wrong. The goal of the program should be to prevent these errors from occurring, rather than complaining about them. Design the interface to guide the user to enter appropriate data by constraining formats, presenting valid options and disabling dependant steps until the dependencies are satisfied.
6. **Create a safe environment for exploration**: Good programs allow the user to investigate features and functions without fear of getting lost or loosing/changing information. This involves making actions reversible, supporting "undos", creating a good sense of "home", and providing various paths for completing tasks.
7. **Design controls to be intuitive**: Users should be able to anticipate a control's behaviour from its visible properties. Metaphors should be used whenever possible to describe the controls behaviour (i.e. trashcan icon in *Windows*).
8. **Minimise "modal" behaviours:** Avoid locking users into situations by presenting multi-tasking capabilities with escape options.
9. **Allow user customisation:** Design to allow users to simplify tasks that they repeat often, and present outputs in a format of their choosing.
10. **Use multimedia effects sparingly**: Sound, colour, and animation can make an application look professional, but a balance is required to maximise their effectiveness. They should be used only as a secondary mean of communication. Remember that many users may be colour blind or deaf.

7.5.2 Evaluation of existing interfaces

In developing the graphical user interface for the *FIDO* decision support software, the interfaces of the two leading surface irrigation software packages, *SIRMOD* and *SRFR,* were studied to determine their strengths and weaknesses. Both of these programs evolved through several revisions during the course of this study; from *DOS*-based programs into *Windows*-based software. However, it was the versions that were available between 1999 and 2005 that have had the greatest influence on the design of the *FIDO* user-interface.

Both software packages have gone through several evolutions of interface design during their development cycle. Both have similar interface capabilities with animated graphical outputs, and dialog based inputs. Both packages operate with a distinct modal behaviour segregating input, output, and simulation operations. It was found that while the outputs from both tools were useful and attractive, the interfaces were awkward to use and lacked flexibility.

SIRMOD has evolved over several versions primarily as a *DOS*-based research tool, and was converted to Windows in 1997/98, in an effort to simplify and modernise the interface. In *SIRMOD*, only one set of data can be open at a time. It makes extensive use of tabbed dialog windows to enter the simulation parameter data (Figure 7.10a), with results being presented in a range of graphical and tabular outputs (Figure 7.10b-d) including an animation of water flowing down and infiltration into the furrow (Figure 7.10b).

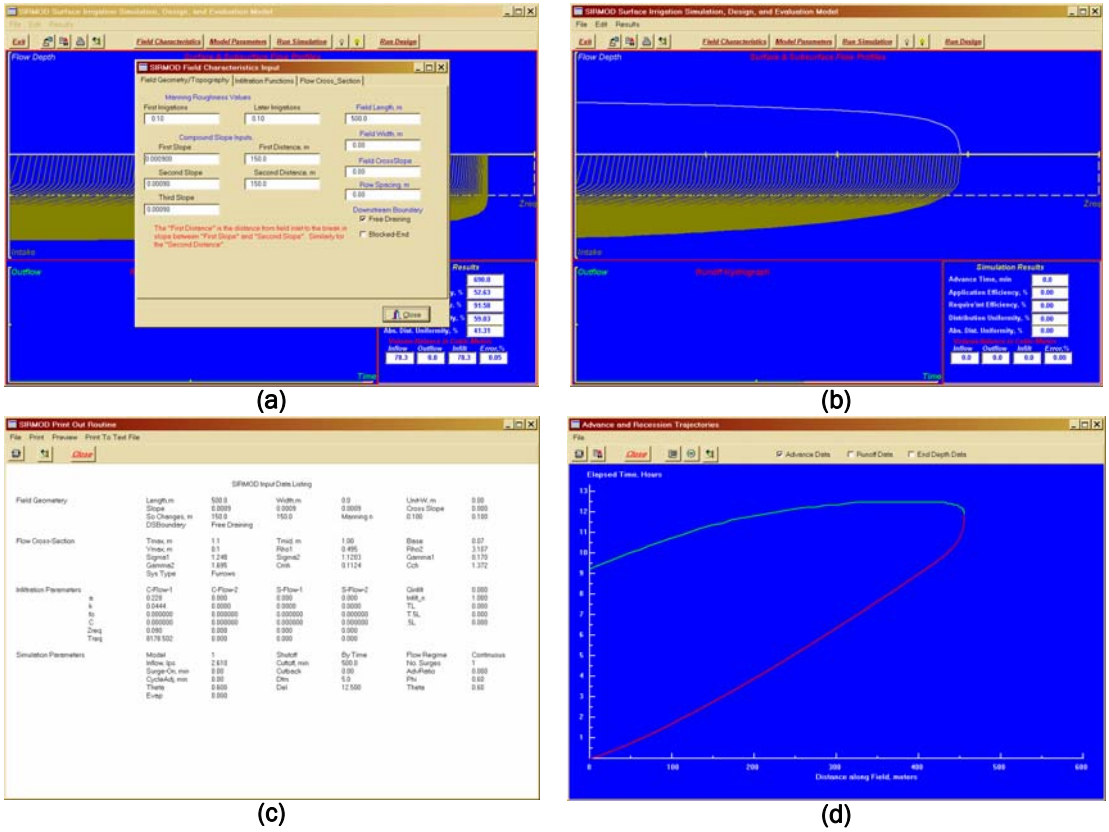


Figure 7.10: *SIRMOD* user-interface screenshots: (a) shows one of the many input dialogs; (b) animation of water flowing along and infiltrating into furrow; (c) tabulated input parameters; and (d) plotted output of advance and recession characteristics.

The *SIRMOD* interface appears simple on first viewing with relatively few buttons and menu options. However, the complexity of the interface soon becomes apparent once the "Field Characteristics" dialog or the "Model Parameters" dialog is opened. While parameters in these dialogs are cleverly grouped in common categories, one of the downfalls of this presentation is that every parameter is shown, independent of which modelling options are selected. No attempt has been made to hide parameters that are not required for the current set of management options. From experience gained through *SIRMOD* training workshops at the NCEA (National Centre for Engineering in Agriculture) in Toowoomba, this has caused considerable confusion among users. Many of the parameters are empirical and poorly labelled, while some of the physical parameters listed are not required for the simulation, but the user has no way of knowing this.

Unlike *SIRMOD*, *SRFR* has remained a *DOS*-based program for most of its development cycle (until just recently) featuring a "*Windows*-like" graphical user interface and well organised operational structure. In the most recent *DOS*-based version, inputs are entered into a dialog that is much less cluttered than in *SIRMOD*, but offers fewer input options (Figure 7.11a). A flow animation output is available (Figure 7.11b) along with other graphical and textural outputs (Figure 7.11c-d).

While the *SRFR* interface is relatively uncluttered, a major criticism of the interface is that it feels "unfamiliar" to use, given the *DOS*-based interface that

fails to maintain consistency with other Window's software. The controls aren't intuitive, and there is distinct "modal" behaviour with little sense of "home". Being developed in a non-*Windows* environment also introduces compatibility problems with printing and importing/exporting data and results.

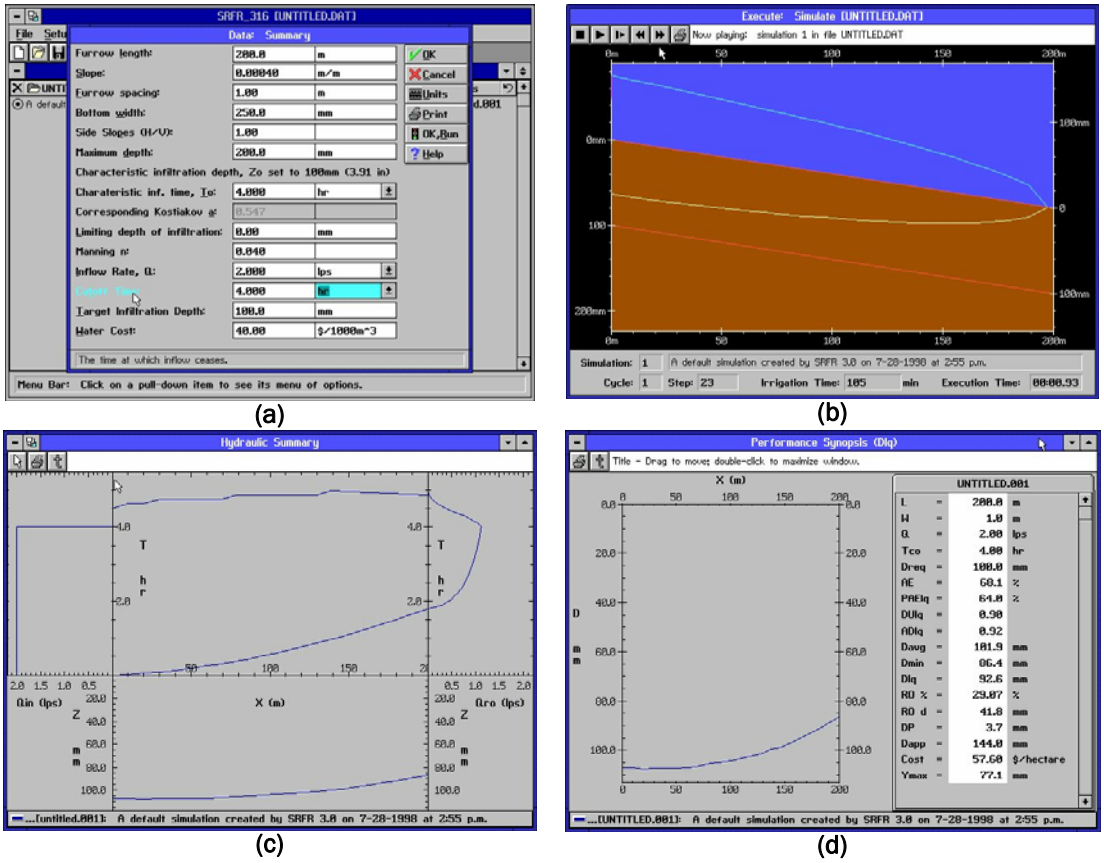


Figure 7.11: Screenshots of the *SRFR* interface: (a) shows the main parameter input dialog; (b) represents the animation of water flowing along and infiltrating into the furrow; (c) demonstrates curves of advance, recession, inflow, runoff, and infiltration; and (d) shows the infiltration distribution again, along with the performance summary figures.

Probably the most attractive feature of both programs' interfaces is the simulation animation of the water flowing down along the furrow. Experience has found that users are initially intrigued by this and its potential to demonstrate water infiltrating below the root-zone. All of the other graphical outputs are also useful, but are inflexible in terms of exploration and presentation; that is, the user cannot "zoom" or "pan" around the charts or compare them directly with other outputs. They lack the potential flexibility that professional quality third-party charting packages offer.

At the time of writing this, new versions of both products had only just been released, so there was no time for them to have any real influence on the *FIDO* interface design. *SIRMOD* has been redeveloped into an internal program for the US Department of Agriculture called *NRCS_Surface* (USDA, 2006) and was not available for evaluation, although it appears that the interface has changed very little to that reviewed earlier. Of more significance, are the substantial changes to the *SRFR* software with the introduction of a *Windows* version called "*WinSRFR*" in 2006.

The developers of *WinSRFR* (AARC 2006) have gone to great lengths to provide a user-friendly interface to encapsulate the combined functionality of the *SRFR*, *BORDER* (Clemmens *et al*. 1996) and *BASIN* (Clemmens *et al*. 1995) software. Nevertheless, *WinSRFR* still uses a very traditional form of interface architecture with many tabs, buttons, information elements, and modal dialogs all mixed together. To simply its operation, it is embedded with an extensive amount of textural guidance and suggestions to help the user navigate through the program. On most screens, the user will find instructions on how to proceed, with many "question and answer" type scenarios.

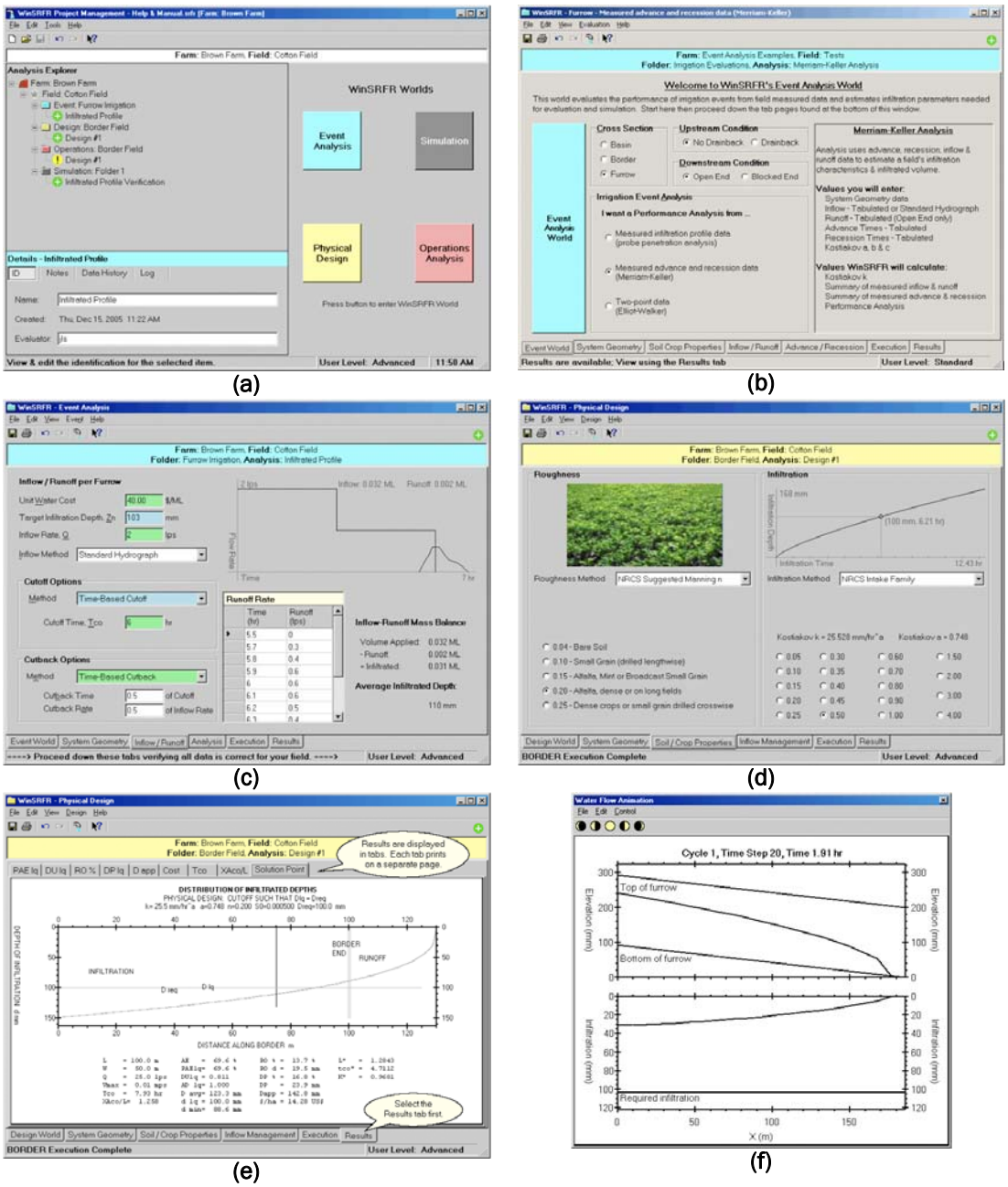


Figure 7.12: Screenshots of the *WinSRFR* interface: (a) Project Management Window; (b) Event Analysis World; (c) Inflow management screen; (d) hydraulic roughness and infiltration characteristics; (e) infiltration outputs; and (f) simulation animation. (Screenshots sourced from the *WinSRFR* 1.0 User Manual, 2006)

WinSRFR uses an analogy of "Worlds" to separate the tasks of event analysis, simulation, physical design, and operational analysis. These represent four separate interface pages that are linked by a central database (Figure 7.12a). Many tabs exist at the bottom and top of each page to switch between different inputs and outputs. A prominent feature is the amount of text on each page to guide the user through the process (Figure 7.12b). Separate pages exist for many of the input variables and parameter such as inflow (Figure 7.12c) and infiltration and roughness (Figure 7.12d). Outputs are very similar to the DOS-based version and still lack the flexibility of a modern graphics library (Figure 7.12e,f).

While the design of *WinSRFR* is a big improvement over the previous versions, it is now a much more complex and "daunting" piece of software, reflecting the diversity of new decision support tasks that it is required to perform. There is always a large amount of information being presented to the user at any one time, despite attempts at both grouping and segregating functionalities. Only time will tell how users will react to the complexity of the interface, and whether it will encourage new users to adopt this technology.

A common limitation of all of software packages is the modal behaviour of operation. That is, the user is forced to be in either "input mode", "output mode", or "action mode", with many steps required to progress from one stage to the next. For comparison, recent programming practices would encourage inputs and outputs to coincide and harmonise, while the "actions" would be designed to run "multi-threaded" in the background.

7.5.3 Prototyping the interface

The current version of the *FIDO* interface has evolved over many different configurations. Appendices 7.3 to 7.7 demonstrate the range of designs that have been developed and tested during the prototyping. Initially, many poor design choices were made while encapsulating the complexity and volume of tools required by the decision support system. Many of these problems were present because of the limitations of the software engineering tools, underdeveloped programming skills, and a lack of clarity in getting away from the simulation-centred design philosophy of the *SIRMOD* and *SRFR* software. Therefore, the initial prototypes of the decision support software were characterised by four main problems.

Firstly, initial versions of the software tried to provide general access to all operations of the decision support system from the main interface screen. This was done by configuring different analyses on separate pages accessible from the "tabs" located at the top and/or left hand side of the screen. In practice, this was found to clutter the interface and provoked confusion from evaluators. This also required that the user go to a particular page before conducting an operation such as simulation or calibration, adding extra unnecessary steps to the process. In later versions, a single button with a drop down menu linking the analyses was added to the toolbar to remove the tabs, and provoke the "action" (simulate, calibrate etc) before going to the analysis page. In the current version, this feature was removed in favour of "hyperlinks" in the database report. In this

way, actions are only presented if the action can be performed. This naturally provides progressive disclosure of the actions, and hyperlinks to the actions coincide seamlessly into the report. In effect, actions are now performed in a web-browsing-like manner.

Secondly, the first few versions of this software presented the focus of the interface on the currently loaded set of irrigation data. While a database was present, its main role was to provide alternative sets of data for this focus. However, this limited the use of the tools to analysing only a single set of data at a time, even though multiple sets of simulation results could be loaded simultaneously. It was only in the most recent version of the software that this central focus was shifted to the database itself, whereby the main focus or "home" is the entire database, and the secondary focus is any of the property, paddock, event or furrow records. This also allows progressive disclosure of analyses that are compatible with the selected record type.

Thirdly, the initial software versions were composed of too many types and number of interface components. This had two pronounced effects. The first effect was that the interface became cluttered with too many controls, and too many options presented simultaneously for the user. A good example of this is the initial database functionality, which contained an embedded version of *Microsoft Access* to maintain records. With this database came a range of controls, tables, and tools for navigating, editing, displaying and searching the database, when only a limited amount of functionality was actually required. There were also flexibility issues with using this feature as most of the functionality was built in and couldn't be easily customised. It also added to the size of the software and required the *Access* database engine to be installed on the computer along with software. The second effect of having too many components was that the software engineering became increasingly difficult to maintain. Typically, these components were employed using the RAD (rapid application development) functionality of dragging and dropping the components onto forms. In software engineering terms, this "individualises" the component rather than existing in a generalised form of array that could be treated as a group. Extra programming was then required to achieve this "group" functionality. The final result was that the initial software versions were plagued by excessive amounts of interface code that was difficult to maintain.

Finally, the object-oriented program structure was poorly developed in the initial versions of the software. This was primarily caused by immature programming skills at that stage of development. By utilising more object-oriented programming features in the later versions, a more powerful and less complex graphical user interface could be achieved. For example, using polymorphic "Tree Parameters" provided the means to simultaneously develop the database structure, a visual database-navigator tool, database editing facilities, database reporting facilities, and file input/output operations. Also, a consistent look and feel to the analyses was achieved throughout the program by using an object-oriented structure to derive each analysis from the same parent class.

7.5.4 Current interface functionality

The current version of *FIDO* is designed to behave with web-browser-like functionality. Built around the irrigation database, the user is able to navigate through the program, exploring data and analyses similar to browsing an Internet site. However, in this case, the information that the user sees comes from the database, rather than the Internet[14]. This is achieved through displaying all of the information, data, results and hyperlinked-actions inside a HTML report (created through *XML /XSLT* transformations) for the selected database record. Familiar web-browser controls such as “home”, ”back”, “next” and bookmarks exist, with the report for the selected record serving as the homepage for the user to return to after performing any actions.

Built closely around progressive-disclosure techniques, few options and information are presented to the user at “start-up”, with more becoming available as the user progresses through different tasks. A basic-level user may spend all of their time just viewing and updating the ”reports”, while an advanced-level user may probe deeper into more complex analyses. Menus are also context sensitive to the current task being undertaken with currently unavailable commands being hidden until required.

7.5.5 Interface layout

The current version of the *FIDO* software is characterised by a graphical interface designed using a three-panel layout (Figure 7.13):

- a top panel contains menus, toolbars and status bar;
- a left panel contains user input information including a database navigator control, and simulation input data for the analyses; and
- a right (main) panel contains output information (and inputs as well when viewing the database). This includes both HTML reports and other more complex analyses.

A splitter bar is used between the input and output windows for resizing and presentation purposes. The user can alternate between hiding and showing the input panel by clicking the splitter bar.

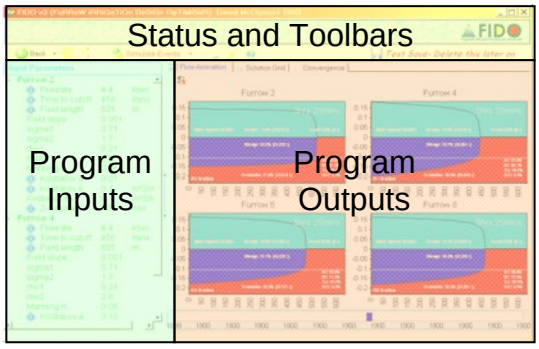


Figure 7.13: Layout of the *FIDO* graphical user interface.

[14] The potential exists for later versions of the software to store and access irrigation data from an internet server.

There are many advantages of this layout. *Firstly*, the user need not be concerned with arranging "windows" and "dialogs", as with "multiple document interfaces". In this example, graphical information is always automatically arranged, maximised and centred. Preference is given to displaying data "wholly" rather than trying to display too much information at once.

Secondly, task-inputs are always available to the user at their location in the left hand panel. This allows for instantaneous and simultaneous updating of the program outputs. The alternative to this is to display input options in dialogs (such as $SIRMOD$) where the outputs are quite often not updated until these dialogs are closed.

Finally, the large status bar continually keeps the user informed of the program's state and progress. Menu and tool bars are placed in their usual position at the top of the screen in accordance with the "principle of consistency". Other toolbars are used in the program and are located inside the input and output panels. These are duplicate commands designed to be located conveniently next to the objects that they act upon, but are also located in the main menu or main toolbar.

7.5.6 Modules

From the developer's perspective, the $FIDO$ user interface can be divided into five main sub-modules coinciding with the five main requirements of the decision support system, including:

- Database module;
- Simulation Analyses;
- Calibration Analyses;
- Optimisation analyses; and
- Response-surface generation and parameter analyses.

From the user's perspective, this modularisation is not apparent as it is hidden by the progressive disclosure and encapsulation techniques applied. These five interface modules will now be discussed in turn.

Database module

The database and its interface form the "centrepiece" of the decision support system. The database is not only a source of input data, but it is also a repository of results with associated analyses to post-process and summarise irrigation and infiltration performance. The database interface serves as a "home" location for program navigation, and presents the principal user-interface components of the software.

Early versions of the database interface (Appendix 7.7) employed the Microsoft Access database engine, and a series of data tables and tab fields. The current version has replaced this methodology with an XML database which opens up the interface to a progressive disclosure oriented design. It consists of four main components including:

- the database navigator panel;

- the database reporting window;
- the database editing window; and
- performance and infiltration summary analyses.

The *database navigator panel* (Figure 7.14) allows the user to navigate, select (and to some extent, edit) the property, paddock, event and model data using the *VirtualTreeView* component. The tree-like structure permits a "progressive disclosure" of the data greatly simplifying the interface. The record names displayed in the tree are complex "tree-parameters" (derived from `TCustomTreeParameter`), which communicate with the database and contain file input/output capabilities. They have an associated drop-down menu with special options for the selected record.

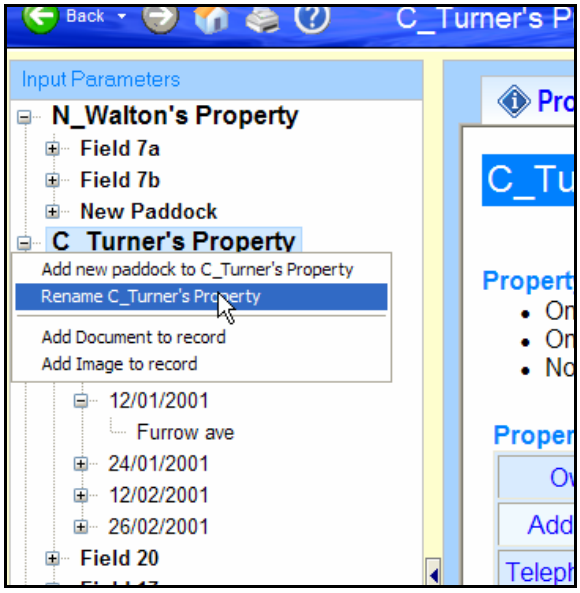
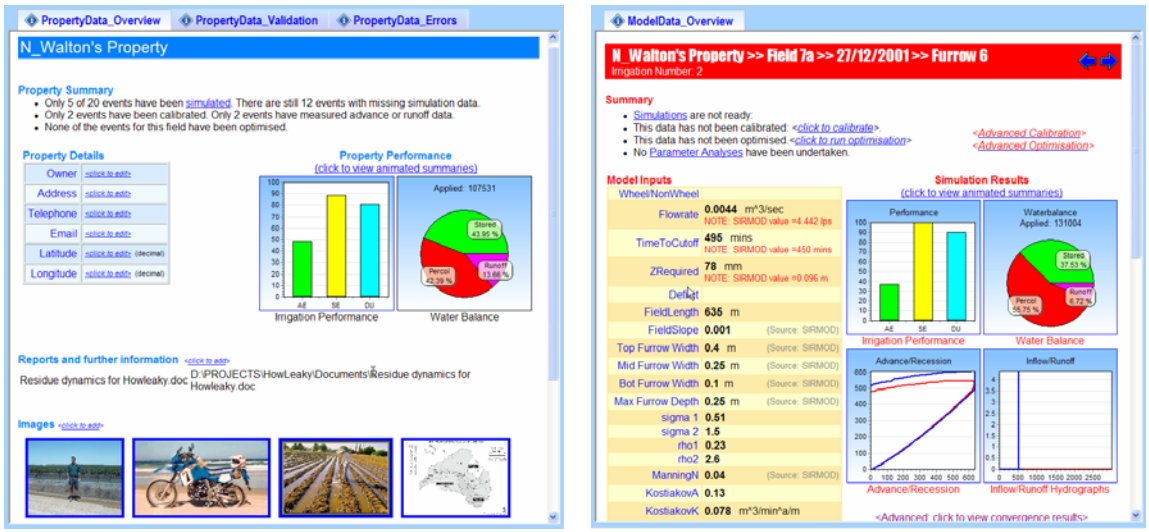


Figure 7.14: Database navigation panel showing drop down menu of commands.

The *database reporting window* (Figure 7.15(a) and (b)) is the central reporting tool in the decision support system. All program inputs and outputs can be presented in this window using customisable reporting templates. The reports are automatically generated in *HTML* format after selecting a record, through transforming the record's *XML* data-file using a *XSLT* stylesheet (template). A key feature of this technology is that the stylesheets are external to the decision support software, and can be redesigned or modified at any time by people other than software developer. In an ideal situation, a graphic designer could be employed to develop new eye-catching reports for the software.

The software has been designed to load any number of report stylesheets at one time, so that the user can switch between different reports and treatments of the same data-set. Because the resulting reports are in *HTML* format, hyperlinks can be added to the reports, to initiate actions such as editing the data, running the simulation and performing a calibration/optimisation/parameter-analysis. As the source data is in *XML* format, any type of attribute information can be associated with the data and displayed in the reports, including comments, units and "date-modified".



(a) (b)

Figure 7.15: Sample database reports displayed in the *database-reporting window*. (a) Property record report showing property statistics, and linked images. (b) Model record report showing simulation input data and results.

The *database-editing window* (Figure 7.16) hides behind the reporting window and is invoked if the user selects any of the hyperlinked parameters. This editor is required as the user is not able to directly edit the data in the transformed *HTML* report[15]. When editing the database, the selected record is displayed using *VirtualTreeView* component that provides editors for the different variable types. The user is returned back to the database report window when the "enter" key is pressed, or the "return to report view" button is clicked at the top of the editor.

[15] Several technologies could have been used to directly edit the transformed *HTML* report. However, each have associated limitations and complexities and were not included in this version of the software. For example, edit dialogs could have been embedded into the stylesheet, and would appear in the *HTML* report to allow editing. However, to get the edited values back into the database, a "post" button would be required on the form, and extra programming needed to do the processing. Also the visual appearance of the report is diminished when cluttered with edit boxes.

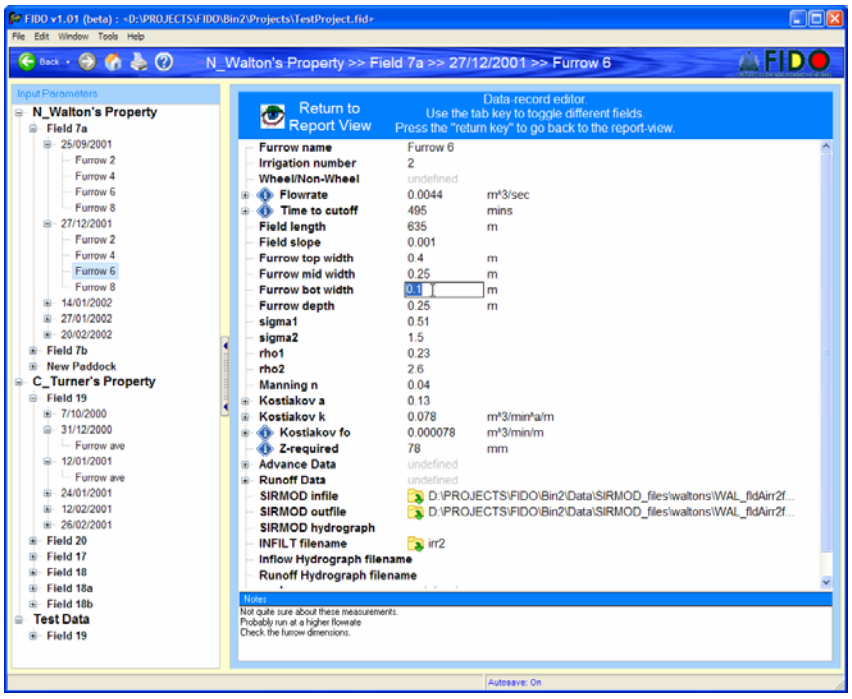


Figure 7.16: Database editing window showing the editing of the "Furrow bot width" parameter. The database report window will be revoked once the user presses the "enter" key.

Performance and infiltration summary analyses (Figure 7.17a,b) have been added to the database module to provide a direct summary of property, paddock, event and furrow data. This interface component is presented using the `TBrowerForm` component to provide an interactive analysis for the user, who is able to select and explore a range of summary information at each level from the paddock through to the furrow.

(a) (b)

Figure 7.17: (a) Performance and (b) infiltration summary analyses. (note: screenshots are from an older version of *FIDO*, but the functionality is the same)

Simulation module

The primary interface for displaying simulation information is through the database report window whereby simulation outputs including advance/recession trajectories, inflow/outflow hydrographs, and performance information are displayed. The user selects the hyperlink for "simulate" in the

report of the selected record, and the report is automatically updated with the new results. Multiple simulations are conducted if the selected record relates to a property, paddock, or event.

The simulation module also contains other more detailed analyses including an interactive animation of the simulation. Three main analyses exist for displaying advanced simulation information including:

- a simulation summary analysis;
- multiple record analyses; and
- a simulation convergence analysis;

The *simulation summary analysis* is a highly graphical interface for presenting simulation outputs. Like *SIRMOD* and *SRFR*, an animation of the simulation surface water and infiltration profiles is available, although this version is interactive with a slider-bar control at the top of the screen to navigate through the animation. Also present are the advance and recession trajectories, and the inflow and runoff hydrographs.

When this analysis is activated, the database navigation panel is replaced with the list of selected simulation data, and subsequent field and management parameters. These parameters can be edited through this facility with the simulated results being updated automatically (in the background in a separate operating thread) when the "return key" is pressed.

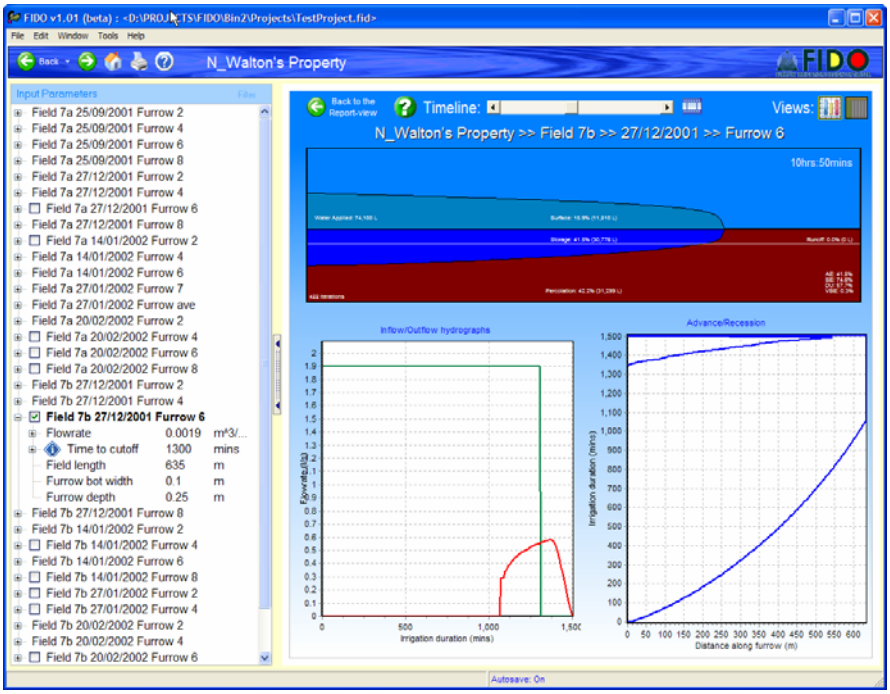


Figure 7.18: Simulation Summary Analysis.

One of the powerful analysis features of the *FIDO* decision support system is its ability to directly compare outputs from different irrigations. A range of *multiple record analyses* have been created for this purpose including a flow animation analysis (Figure 7.19a) and solution grid analysis (Figure 7.19b). Graphical

results can be compared side-by-side, overlaid, or stacked on top of each by setting the display format.

Figure 7.19: Advanced comparison of (a) animated flow profiles, and (b) simulation solution grid (advance/recession trajectories).

Another advanced analysis that is available is a *simulation convergence analysis* (Figure 7.20). This analysis was developed to study and debug the simulation engine and to optimise its performance. Convergence plots of the simulation solution parameters (*Q*, *A*, *dx* and/or *dt*) are displayed along with the simulation animation, and iterations history. A slider bar allows the user to explore convergence at different times during the simulation.

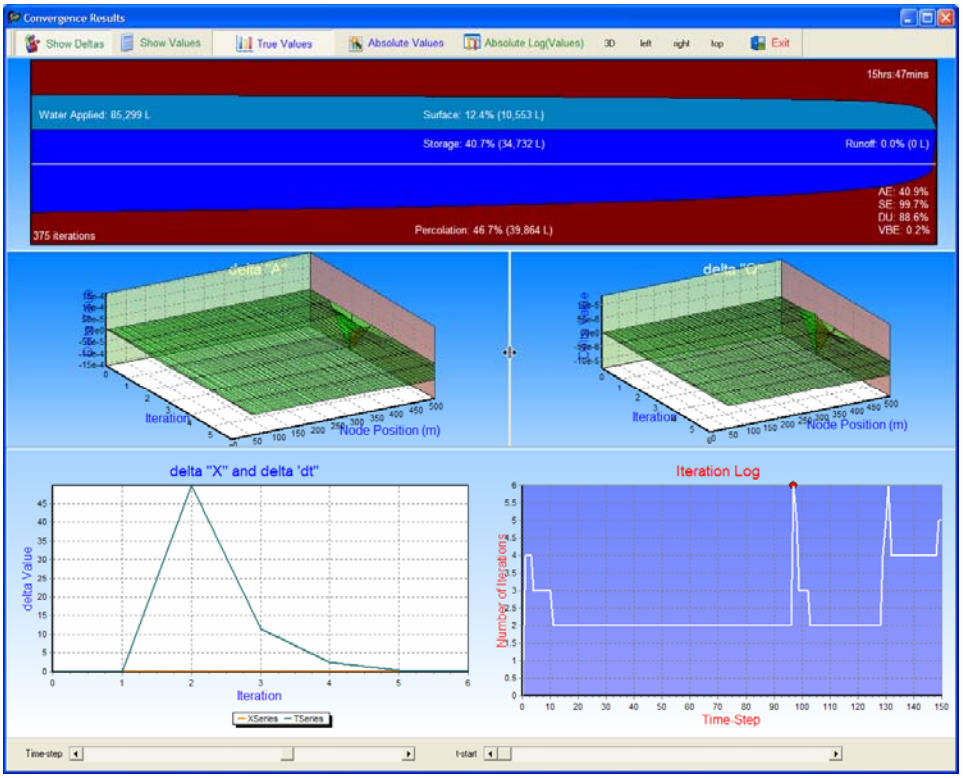


Figure 7.20: Advanced simulation convergence analysis. This is presented as a popup dialog.

Calibration module

The graphical user interface for the calibration component of the decision support system is minimal. If data is available, calibration can be initiated by clicking on the action hyperlink in the database report window for the selected record. The calibration parameters and objective-function will be determined automatically based upon the type of measured input data. Because calibration can take several minutes, a progress bar is displayed in the main title panel. When the calibration is completed, the report is updated with the calibrated results.

An *advanced calibration analysis* (Figure 7.21) can optionally be activated to monitor parameter changes during the optimisation process. Based upon several prototype versions (Appendix 7.4), graphs are displayed showing changes in the calibration parameters, objective-function value, and measured and predicted advance trajectory and/or runoff hydrographs.

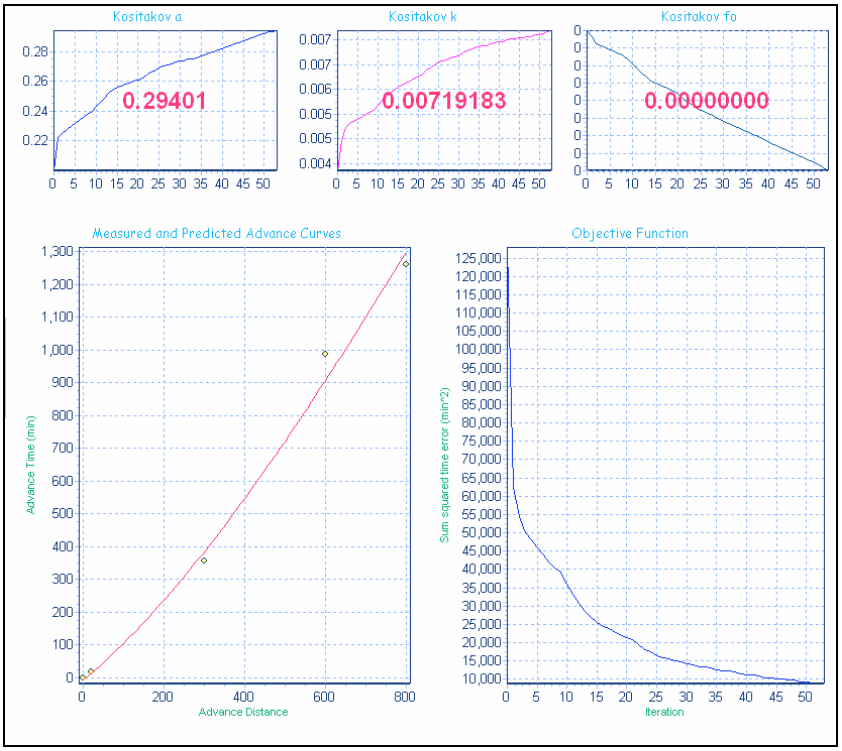


Figure 7.21: Advanced calibration-monitoring analysis. (note: screenshot is from an older version of *FIDO*)

Optimisation module

Like the calibration module, the graphical user interface for the optimisation component of the decision support system is also minimal. Optimisation can be initiated by clicking on the action hyperlink in the database report window for the selected record. Different hyperlinks can be added to the report to initiate different optimisation options. An "objective-function setter" dialog (Figure 7.22) is then presented to the user to specify priority weightings for the objective-function before a progress bar updates the user to the state of the optimisation.

This dialog is a unique graphical tool that has been created especially for the user to set the design weightings for the irrigation performance object function. The "objective-function setter" is in the form of a pie-chart with "user-sizable" pie pieces representing the weighting factors in the objective-function. "Handles" appear on the pie pieces when the user passes the mouse over the pie-chart in order to resize the proportions.

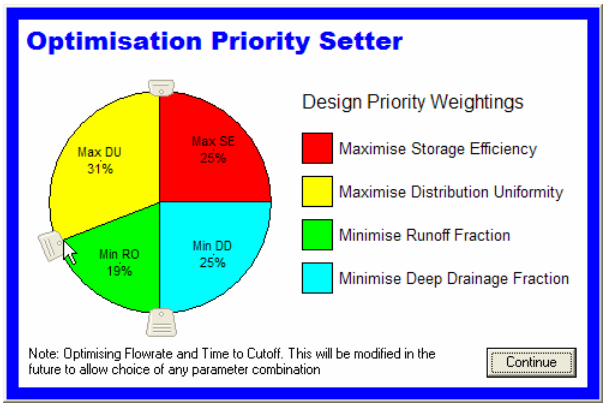


Figure 7.22: Optimisation objective-function priority setter.

Several versions of this tool were created before selecting this final design (see Appendix 7.5). Originally the tool consisted of four linear slider bars, where the user would adjust each in relation to each other to achieve the weighting-split. However, this caused several problems relating to the proportional nature of the task. Changing the setting of one bar inadvertently changed the other weightings automatically as the relative position between the bars changed. Also several different setting arrangements could be used to achieve the same result. For example, an equal twenty-five percent weighting split could be achieved by setting all sliders at the same setting, regardless of the value of this setting. In a later version, a pie chart was added above the sliders to try and provide a visualisation of the proportions. Unfortunately, this did little to alleviate the problems. Therefore, this new component was developed to allow the user to directly manipulate the pie settings.

An advanced optimisation monitoring analysis also exists using the same interface as shown in Figure 7.21. In this analysis, irrigation performance values are displayed instead of the measured data quantities.

Parameter analysis module

An advanced graphical user interface was developed for the parameter analysis module, to simplify the process of generating response-surfaces, and for generating guidelines for irrigation design and management. The interface consists of several components including:

- A parameter analysis setup dialog for designing the analysis;
- A response-surface generation analysis;
- A user-defined contouring analysis for generating guidelines for design and management; and
- A filter tool for modifying the surface and contour series;

This interface was developed after several prototyping iterations (Appendix 7.6) with the current version reflecting the need for complex analysis capabilities, and a simple and manageable code base. Earlier versions proved difficult to maintain and extend and were plagued by stability problems.

In the current version, parameter analyses can be initiated from the database report window by clicking on an appropriate action hyperlink. A *parameter analysis setup dialog* (Figure 7.23) is then presented to the user to define:

- the objective-function to investigate;
- the parameters to include in the analysis;
- the type of infiltration property to use (measured, or highest, lowest or average for site); and
- The step-count for each parameter in generating the response-surface.

Analyses are generated and automatically saved. A hyperlink to the saved file is added to the current record, and displayed in the database report window.

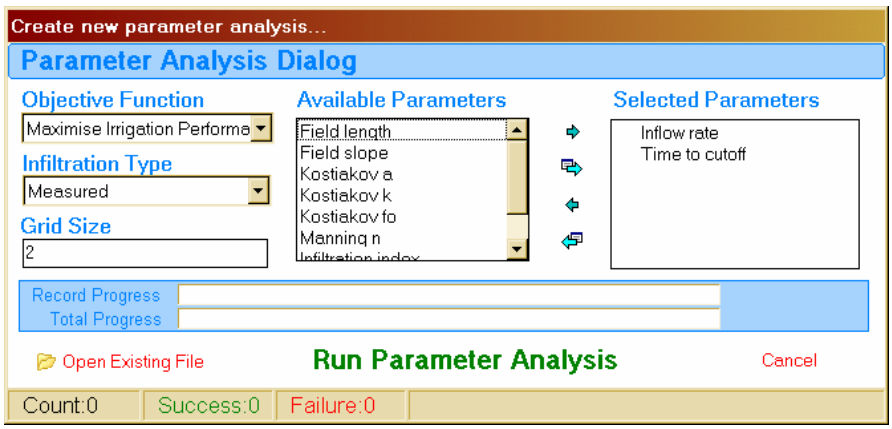
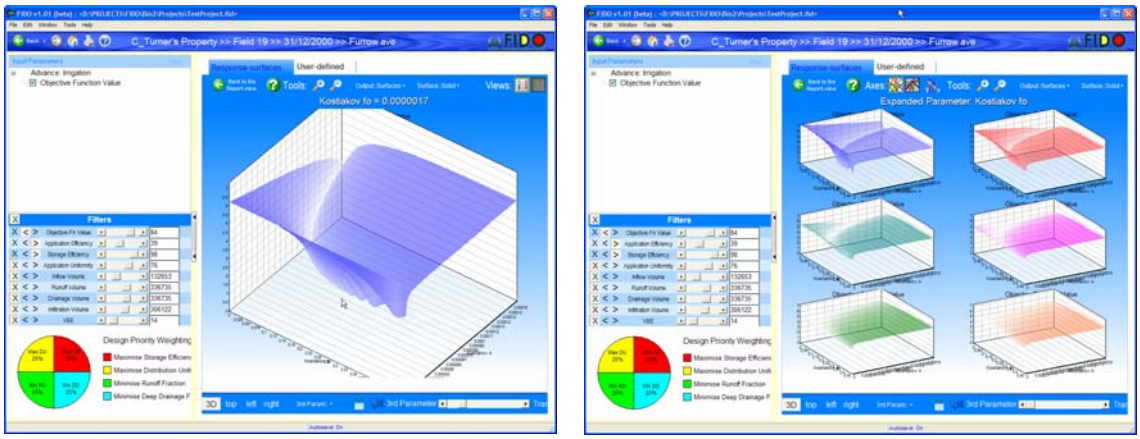


Figure 7.23: Parameter analysis configuration dialog.

The *response-surface analysis* (Figure 7.24a,b) was developed to simplify the investigation of multi-parameter interactions. When three design parameters are included in the analysis (for example, flowrate, time-to-cutoff and field-length), the third parameter is represented by the scrollbar at the bottom of the window. Changing the position of the scrollbar updates the value of the third parameter and hence, changes the active response-surface (Figure 7.24a). The third parameter can also be expanded, so that several graphs appear representing different values of the third parameter (Figure 7.24b). At any stage, the parameter order can be interchanged so that other parameters can be expanded.



(a) (b)

Figure 7.24: Response-surface generation for three design parameter. (a) Third parameter is represented by setting of scroller-bar. (b) third parameter is expanded, showing a separate response-surface for each value of the parameter.

The *user-defined contouring analysis* (Figure 7.25) is the primary tool for generating guidelines for design and management. It has already been briefly introduced in Chapter 6. Once response-surfaces have been generated, they can be dragged and dropped onto any of a predefined number of charts to generate new contours. Any number of response-surfaces can be superimposed to build up layers of contours. The axis-types are defined corresponding to the available parameters. Once a configuration has been created, it can be saved for later retrieval. As with the response-surface analysis, the value of the third parameter can be adjusted by setting the position of the scrollbar at the bottom of the window.

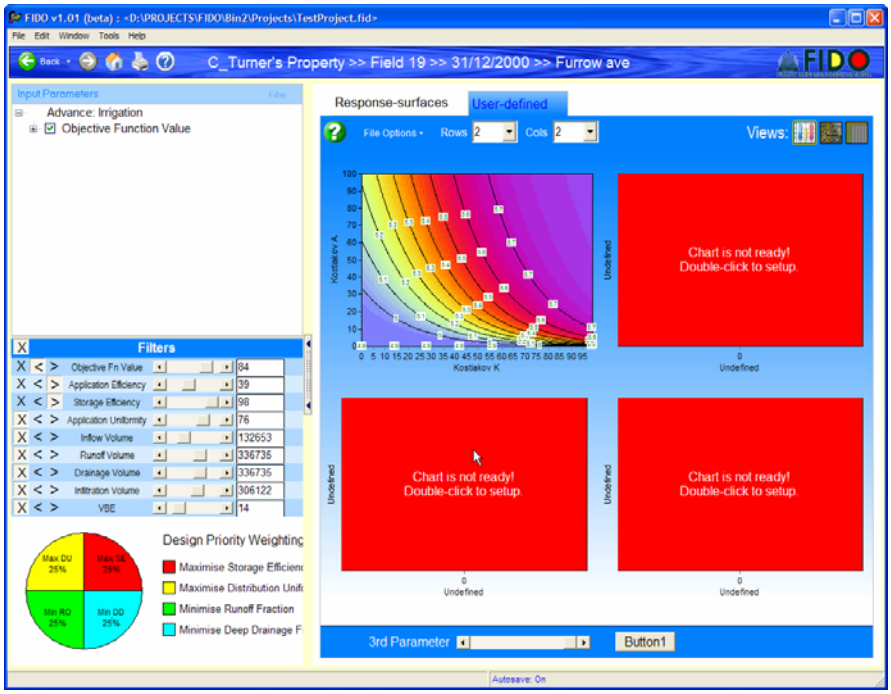


Figure 7.25: Design and management guideline analysis showing setting up of guideline grid.

For both of these analyses, a *filter tool* (Figure 7.26) has been developed to hide or show parts of the different response-surfaces depending upon a set criterion. For example, the user could hide parts of a surface or contour series where performance was greater than a designated value. This creates a visible working envelope of design parameters where the irrigator should aim to operate the irrigation system.

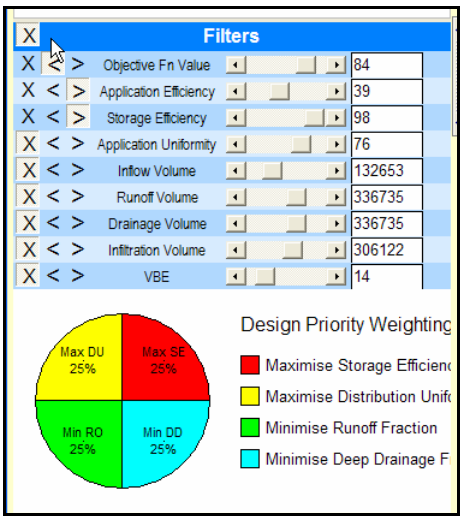


Figure 7.26: Response-surface filters for hiding/showing objective-function parameter ranges. Objective-function weightings can be assigned using the setter at the bottom of the dialog.

7.6 Using the decision support system

A typical usage of the decision support system would be:

1. Update property, paddock, event and model information in the database;
2. Select a property, paddock, event or furrow (model) record from the selector menu at the left of the screen.
3. Calculate the soil infiltration parameters for each model record by calibrating using the measured advance data. Clicking the "Calibrate" option from the action-menu starts the process for the selected record and its children.
4. Simulate these records to assess the irrigation performances using the "Simulate" option.
5. Assess optimum performance by selecting the "Optimise" option and prioritising irrigation management objectives.
6. Develop a series of design charts to assess likely performance over a range of infiltration conditions using the "Analyse" option.

7.7 Conclusions

A decision support system for furrow and border irrigation was developed by combining the tools developed in Chapters 3 to 6 with a database and graphical user interface. There were three focus areas during this marriage of components; *firstly*, an object-oriented structure was developed to accommodate program elements concentrating on separating the graphical user interface components from other task related objects for flexible future development; *secondly*, a

database was developed using *XML*-based technologies to store property, paddock, event and model information; and *thirdly*, a user-friendly graphical user interface was created with web-browser-like functionality. The software has evolved through many different prototypes versions with its current design being heavily influenced from the successes and mistakes of the previous attempts.

Chapter 8 Conclusions, implications and recommendations

8.1 Introduction

The objective of this dissertation was to develop a new decision support system for furrow and border irrigation aimed at increasing the usability of the technology to improve decision-making capabilities. Specifically the research hypothesis stated: **"That calibration, optimisation, and parameter analysis capabilities can be developed and integrated with an accurate and robust simulation model into a decision support system to improve furrow and border irrigation performance."**

This chapter presents the conclusions and implications of this research along with recommendations for future research. The discussion in this chapter has five main objectives: *(1)* it will provide a summary of the work undertaken in the previous chapters in order to highlight the logical progression of ideas and issues addressed in answering the research questions; *(2)* conclusions are presented for each of the research questions; *(3)* practical implications of this research are discussed; *(4)* the limitations of this research are acknowledged; and *(5)* recommendations are presented for future research and development.

8.2 Overview of previous chapters

Chapter 1 introduced the hypothesis and research objectives before explaining the relevance of the research problem. Justification for the research focused upon three interrelated problem areas: *firstly*, existing surface irrigation models aren't being used due to problems inherent in the software. These problems included over-complexity, poor software engineering, and reliability problems. *Secondly*, there is a need to aggregate different surface irrigation model components into a single decision support system. These components included a database, simulation engine, calibration (parameter estimation) capabilities, optimisation capabilities, and parameter analysis components (design curves). *Thirdly*, there was a need to combine modern software engineering concepts and tools with existing surface irrigation simulation technology.

The main purpose of **Chapter 2** was to investigate the history, performance and potential of existing surface irrigation decision support technology. The background theory of surface irrigation and computer modelling of surface irrigation was initially presented as a basis for this research. Then a literature review was undertaken into the three main surface irrigation research areas of: simulation modelling; solution of the "inverse problem"; and optimisation of design and management practices. It was found that gaps exist in the literature, especially with: simulating the later stages of the irrigation cycle; converting the mathematical model into computer code form; ensuring simulation robustness;

calibrating using the complete hydrodynamic simulation model; parameter analysis of system responses; and automating the optimisation process. Independent testing of existing surface irrigation models showed that they can accurately simulate surface irrigation, with the *SIRMOD* and *SRFR* models being identified as the most prominent simulation models. *SIRMOD* was then evaluated in a case study and found to satisfactorily simulate all phases of an irrigation, but was sometimes subject to stability problems. Functionality requirements for the development of a new decision support system were identified as simulation, calibration, optimisation, parameter-analysis and data management.

Chapter 3 presented the development of a new simulation engine based upon the refinement and modification of existing surface irrigation modelling technology: notably, a complete hydrodynamic model and the Preismann double sweep solution technique. The first goal of the chapter was to revise the solution technique into a simpler and more generic form. An object-oriented structure was then developed to encapsulate the "model" and solution technique, and provide input/output capabilities. The simulation engine was then developed around this structure focusing upon memory management, exception handling, simulation reliability, code clarity and future expansion. Potential sources of convergence problems were identified in the simulation and several techniques were developed to improve simulation robustness. The complete simulation engine was validated against output from the *SIRMOD* model.

Chapter 4 highlighted the need for using the same model for both simulation and soil infiltration and hydraulic roughness parameter estimations. Two different inverse methodologies for parameter estimation were developed; *firstly*, an optimisation-based volume-balance model (*INFILT*) was developed. This technique can determine any of the Kostiakov-Lewis infiltration parameters using inflow-rate and irrigation advance measurements. *Secondly*, a more complex hydrodynamic inverse technique was developed for determining any of the soil infiltration and/or the hydraulic roughness parameters. This second technique drew upon the optimisation-methodology developed and incorporated into *INFILT*. Because of long calculation times associated with the hydrodynamic method, starting estimates were obtained from the output of the *INFILT* method. Three objective-functions were developed based upon advance measurements, runoff measurements, and combined advance and runoff measurements. These methodologies were then encapsulated into the decision support system using an object-oriented structure. Parameter response-surfaces were generated for the advance-based objective-function to identify a unique solution for up to four optimisable parameters. A validation was performed through analysis of the simulation results based upon the calibrated infiltration parameters.

Automated design and management optimisation capabilities were developed for the decision support system in **Chapter 5**, with the added benefit of automated benchmarking of performance potential. This involved using the optimisation technique developed in the previous chapter, and applying it to the case of optimising irrigation performance. A user-defined objective-function was developed based upon different weightings of maximising storage efficiency,

minimising runoff, minimising deep drainage, and maximising application uniformity. Investigation of typical response-surfaces revealed that a single "best" set of design parameters (flowrate, time-to-cutoff and field-length) rarely exists in practice due to a compensating effect by the individual components of the objective-function. This manifests itself in a near horizontal parabolic ridge that maintained constant values for various combinations of the decision variables. In fact, for nearly every possible value of each decision variable, maximum performance could be achieved through a unique combination of the other two variables. Therefore, an infinite number of potential solutions exists for a given irrigation system. Hence, the optimisation problem simplifies down to solving for one decision variable: time-to-cutoff. Solving for more than one parameter would have been impaired due to small variations in the volume-balance errors across the response-surface. This was due to discretisation process used in the simulation solution technique. The method was validated by comparing the measured (simulated) results against the optimised results, with rapid and reliable convergence on the solution when optimising on only time-to-cutoff. The optimised results demonstrated a marked increase in application efficiency accompanied by a small reduction in storage efficiency and application uniformity.

In **Chapter 6**, field design and management guidelines were investigated in the process of developing an automated procedure for the decision support system. As a preliminary study, guidelines were prepared based upon a range of infiltration functions from seventeen surface irrigations. This involved presenting guidelines composed of low, high and average infiltration functions for the site. Charts were generated using a fixed management strategy of minimising runoff and presented iso-curves of flowrate and time-to-cutoff, with irrigation performance and field-length being represented on the chart axes. However, the potential benefits of visualising a large number of dimensions were negated by a limited range of response outputs for a fixed management objective. Nevertheless, this provided a direction for developing the guideline-generating capabilities. This included representing system outputs as contours and iso-curves, rather than by the chart axes; representing different infiltration conditions in separate design charts; allowing the user to assign variables to each chart axis; and representing only two decision variables in each chart. Automation facilities were then created based upon an object-oriented structure. The generated guidelines are presented as contours of performance values accompanied with a user-defined envelope of working ranges.

Chapter 7 focused upon the software engineering components of the decision support system. The main goal of this work was to combine the tools developed in the previous four chapters with a database and a simple user interface to develop a new decision support system for furrow and border irrigation. An object-oriented program structure was then developed to accommodate program elements. An *XML*-based surface irrigation database was presented along with a simple graphical user interface created using advanced third-party libraries. The resulting interface is based upon a web-browser-like design, and relies on progressive disclosure techniques to present advanced analyses. The evolutionary process of developing the final user-interface design was discussed with different prototype interfaces being presented.

8.3 Conclusions about the research problem

The hypothesis for the research problem was successfully tested through the completion of the individual research objectives, and an understanding of the practical implications resulting from the research. Specifically the research hypothesis stated: **"That calibration, optimisation, and parameter analysis capabilities can be developed and integrated with an accurate and robust simulation model into a decision support system to improve furrow and border irrigation performance."**

Testing of this hypothesis has resulted in the development of a decision support system for furrow and border irrigation featuring an automation-capable hydrodynamic simulation engine, automated full-hydrodynamic inverse-solution, automated optimisation of design and management variables, and automated user-definable real-time generation of system response. This has been combined with a highly flexible object-oriented program structure and web-browser-like graphical user interface. This represents a unique holistic development and integration of components into a research and practitioner tool, with competing products offering alternate non-automated and non-optimising capabilities. The demonstrated successful validation of the intended functionalities was a prerequisite for supporting the hypothesis.

Key conclusions from this research are that:

- Only minor enhancements to the existing numerical technologies were required for automating the simulation engine, with the major focus of the study placed upon the operational algorithms to enhance robustness and reliability;
- The inverse-solution using the full-hydrodynamic model is a viable and robust methodology for the unique identification of at least three infiltration/roughness parameters;
- Automated optimisation of design and management practices is limited to the selection of one solution variable (time to cut-off) due to the identification of non-unique multi-variable solutions caused by the interdependency of key decision variables in relation to irrigation performance;
- The automated optimisation facilities provide a unique benchmarking of irrigation performance potential; and
- Automated system response evaluation facilities provide a useful research and practitioner tool, capable of multidimensional analysis of irrigation systems subject to temporal and spatial infiltration variation.

In summary, the research into the development of the system supports the first part of the hypothesis: that is, that the decision support system can be developed through the suggested integration of components. While field-evaluation of the decision support system was not a part of this research, there is much evidence to support the second part of the hypothesis that suggests that the decision support system can improve furrow and border irrigation performance though better decision-making. Six principal pieces of evidence addressed through the Research Objectives that support this includes: *(1)* that the improved robustness and flexibility of the simulation engine allows it to be used in a variety of decision support roles, to accurately and reliably provide

irrigation performance measures to the user; *(2)* that the parameter estimation (inverse method) facilities based upon the hydrodynamic model minimises errors associated with the traditional volume-balance methods, improving the accuracy of the simulated and optimised results, and ultimately, management decisions; *(3)* that the automated optimisation capabilities can quickly and easily provide guidance for optimising water use efficiencies for a range of strategies; *(4)* that the automated optimisation capabilities also provide a measure of "performance potential" for benchmarking practices; *(5)* that site-specific design charts can quickly and easily be developed to identify the irrigation system response, and define design and management strategies; and *(6)* that the decision support components combined with a database and simple graphical user interface enhances the flexibility and utility of the product, while simplifying its operation.

Further evidence supporting the hypothesis, through successful completion of the research objectives and identification of practical implications of the research, are discussed in turn below.

8.3.1 Discussion of the research objectives

***Research Objective 1*: Investigate existing surface irrigation modelling technology to determine a model and solution technique structure suitable for incorporating into a decision support system.** Through a literature review and case study using the surface irrigation model $SIRMOD$, it was established that existing surface irrigation simulation technology is sufficiently accurate to use as the basis of a decision support tool for irrigation design and management. However, complexity, stability and usability issues were also identified with the technology, while gaps in the literature exist relating to strategies to address these problems.

It was recognised that the $SIRMOD$ and $SRFR$ packages are the most important decision support tools in the industry. However, these tools remain relatively unused and offer limited functionality in the overall decision support context. Both tools were identified as having complexity, stability and usability problems. It was concluded that the sources of these problems are interrelated, with the primary source being at the software engineering level. For example, little documentation was found on how to transfer the mathematical algorithms and solution techniques into their computer code form. This is further complicated by a lack of information on how to simulate the later phases of the irrigation cycle including simultaneous advance and recession. Nevertheless, $SIRMOD$ and $SRFR$ are capable of simulating all phases of the irrigation cycle. Given that they have both proved to be sufficiently accurate, and are very similar in design, their underlying model structure and solution techniques were chosen as a basis for a new improved simulation engine.

***Research Objective 2*: Develop a robust, reliable simulation engine for furrow and border irrigation for automation within a decision support system under optimisation and systematic response evaluation.** Based upon the findings of Research Objective 1, a new hydrodynamic simulation engine was developed through combination and simplification of the $SIRMOD$ and $SRFR$ implicit solution techniques, development of an object-oriented structure, and identification and treatment of convergence and stability problems.

This research has revised the Preismann double-sweep solution technique for simulating furrow and border irrigation. It was found that the published equations of the *SIRMOD* and *SRFR* solution techniques were not reduced into their simplest form, so a new set of solution equations were rederived from first principals. These equations provide the same solution as the published methods, and probably have little effect on the efficiency of the solution. However, they differ from the published methods through the calculation of "intermediate values", which significantly simplifies the algebra. The benefit is a much easier-to-understand implementation of the solution leading to both simpler translation into computer code, and a better basis for modification to the solution technique (for example, solving explicitly for simultaneous advance and recession). A generalised model structure was presented allowing different modes of operation including switching between using a constant time-step size and solving for the advance node locations, or having fixed node locations and solving for advance times. Both Eulerian and Langrangian solution methodologies can be implemented and new treatments for the runoff phase and lateral flow conditions were also developed.

The new simulation engine was developed using an object-oriented computer algorithm offering a flexible and stable platform for future research and development. It appears that this is the first attempt at using an object-oriented structure for surface irrigation modelling with previous attempts (including the *SIRMOD* and *SRFR* software) using procedural code. As well as providing a reference point for other researchers to follow, this work has simplified the process of implementing the simulation engine into other decision support software. This has already been demonstrated with the *FIDO* simulation engine being used in other research projects at both undergraduate and postgraduate levels (Dulin 2004; Gillies *pers.comm.* 2005-2006).

Stability and convergence problems were identified and documented. Very little research has been published into circumventing these problems during the simulation, which can largely be attributed to the software algorithm rather than the underlying mathematics. Methods developed to improve simulation robustness include; using small time steps at the start of the simulation to locate more nodes at the top end of the furrow; parameter monitoring during iterations to avoid repeated mirrored oscillations with certain nodes; automatic time-step adjustment to avoid divergence; and forecasting node collapse at the commencement of a time-step to reduce the stress on the solution technique.

The simulation engine proved to be accurate when validated against *SIRMOD* output, and reliable and robust when used in the subsequent optimisation tasks (Research Objectives 3 and 4).

Research Objective 3: ***Investigate and develop parameter estimation (calibration) capabilities for the decision support system.*** The literature review and initial case study identified that using volume-balance methods as a means to calibrate the (hydrodynamic) simulation model can introduce errors into the system. Therefore parameter estimation capabilities were developed for the decision support system using optimisation-driven simulations of the hydrodynamic model. Because of long convergence times involved with running

hydrodynamic simulations, parameter estimates from a simpler volume-balance method (*INFILT*) were used as starting estimates into the hydrodynamic method.

The *INFILT* method was developed as a preliminary study for this research, and is based upon a time-of-advance volume-balance equation linked to a specially developed optimisation algorithm. The optimisation algorithm adjusts the infiltration parameter values in order to minimise the error between the measured and predicted advance data. The method has proven to be fast, robust and reliable and has set a performance benchmark that has already been referenced by other authors (Hornbuckle *et al* 2005; Khatri *et al* 2005; Gillies *et al* 2005; Walker, 2005).

The more complex and computationally intensive hydrodynamic method uses the same optimisation methodology as the *INFILT* method, but requires simulations to be performed instead of simple advance-time estimations. However, a benefit of this is that different objective function formulations can be used, based upon different field measurements including advance and runoff data or a combination of both. This was facilitated using an object-oriented algorithm to allow swapping of objective-functions and simulation parameters (and also optimisation algorithms, and even simulation engines).

Validation of the hydrodynamic method (as well as the *INFILT* method) was performed for the advance-based objective-function and the modified Kostiakov infiltration parameters, and showed that after calibration, the simulated outputs aligned closely with the measured advance. Simulated outputs corresponding the *INFILT*-calibrated infiltration parameters did not align as closely to the measured advance, indicating the error that exists when mixing simulation and calibration model structures. A spike in the objective function output also demonstrated this during the optimisation, when the objective function is switched from the volume-balance to the hydrodynamic method.

As well, response-surfaces were generated for these objective-functions demonstrating that there is a unique solution of the inverse problem. This has typically been inferred by other researchers but never proven due to the dimensionality constraints of investigating three or four parameters. By developing several response-surfaces in "a-k space" for combinations of the remaining design parameter (fo), it was shown that the surface minimum moves around in the in "a-k space", although a true global minima exists overall. It was also shown that the general form of the response-surfaces is simple and regular, without defined local minima.

Research Objective 4: ***Investigate and develop optimisation capabilities for the decision support system.*** Optimisation capabilities were added by combining the optimisation algorithm (developed in Research Objective 3) with the simulation engine and a user-defined objective-function within an object oriented structure. However, the system was found to be unsuitable for solving more than one decision variable (time-to-cutoff is recommended) due to the nature of the response-surface exhibiting multiple solutions and pronounced surface roughness due to discretisation errors in the simulation.

The research involved developing a new user-defined objective-function for optimising the performance of furrow and border irrigation. The novelty of this function is that it allows the user to set predefined weightings on each of the performance components in order to match the irrigation priorities of maximising storage efficiency, minimising runoff, minimising deep drainage, and maximising application uniformity. An optional penalty function was also implemented into this objective function in order to ensure that the advance reaches the end of the field.

An investigation into the response-surfaces for different configurations of this objective-function found that a unique design solution does not exist, and that many combinations of the decision variables (flowrate, time-to-cutoff, and field-length) can be used to achieve a similar level of performance. Therefore, it was recommended to limit the optimisation to solving only for time-to-cutoff. Furthermore, this has implications for irrigation design and management whereby it has traditionally been understood that a unique set of design variables exist to provide maximum system performance.

This research has also identified that the objective-function response-surfaces are "roughened" because of variations in the volume-balance errors between different model runs. This can impede the optimisation algorithm from locating the surface maxima or minima. It was deduced that these inconsistencies are a result of the discretisation process used in the solution technique during each simulation. Even minute differences between different sets of model input data can introduce slight variations in the corresponding volume-balance errors because of the differences in solution grid structure.

Validation was performed by comparing the measured and optimised results when optimising on time-to-cutoff. This demonstrated that a gain in application efficiency was easily attained accompanied by a small reduction in storage efficiency and application uniformity. This situation is representative of irrigators who typically over-water their field in order to completely fill the root zone. This analysis demonstrates the utility of the methodology to quickly and easily present an alternate management solution. It was found that optimisations ran quickly and robustly without user-intervention. Interestingly, it was found that the penalty function was not required to ensure that the field is not under-watered.

An added benefit of the use of this tool is that is allows users to automatically benchmark the potential performance of an irrigation event. This will be a useful feature even for those who don't find this sort of automated tool appealing.

Research Objective 5: *Investigate and develop parameter response (design-charts) capabilities for the decision support system.* Parameter response evaluation capabilities were added by developing a detailed object-oriented structure to automate the process of running the simulation. The development process was guided by the recommendations from an initial case study focusing upon design-chart development.

This study presented design-charts combining the effects of variable infiltration and three decision variables using a fixed management strategy of minimising

runoff. However, a limited range of response outputs for a fixed management objective negated the potential benefits of visualising a large number of dimensions. Nevertheless, the benefit of presenting a range of charts for different infiltration characteristics was recognized and recommended for the decision support software. Furthermore, it was recommended that system outputs be represented as contours with decision variables represented on the charts axes. Decision variables should be kept to two per chart while the user should be able to define which variables are assigned to each chart axis.

Based upon these findings, parameter analysis capabilities for the decision support system were developed allowing the user to generate different guideline configurations for a range of soil infiltration properties. Low, average, and high infiltration characteristics are determined from the paddock site history (based upon measured irrigation runs) to allow the user to identify an envelope of possible outcomes for an irrigation design. As well as providing practical field-specific guidelines, this also represents a powerful research tool for understanding irrigation parameter relationships. The resulting charts can integrate large amounts of data into a simple, easy to follow format that offers great insight into irrigation performance.

Research Objective 6: Develop an object-oriented framework to combine the components developed in Research Objectives 2 to 5 with database facilities and a graphical user interface. An object-oriented framework was developed for combining the decision support components based upon separating the graphical user interface from the decision support algorithms. A major feature of this work is that all components of the system have been developed from first principles and were guided by this framework, with the primary goal of implementation into a decision support system.

This involved splitting components into managers, tools, analyses and interface elements. The open structure ensures interchangability of parts, and new analyses can be added without structural modification to the user interface. An XML-based database was also developed to store decision support inputs and outputs. Data and results are presented using $XSLT$ stylesheet technology allowing an infinite range of interface configurations. This culminated in the software having a web-browser like functionality with progressing disclosure capabilities. A key feature of the user-interface is that multiple irrigation events (or paddocks or furrows) can be analysed and evaluated simultaneously, and with a minimum of user input.

8.3.2 Practical implications of this research

The primary practical implication of this research is that it has contributed to the development of a professional-quality software package, capable of use in many market segments, including researchers, irrigation consultants, and also irrigators who have undertaken some training. While the goal of this research was not to present a commercially ready package, the software is well poised to be further developed to this level. Five specific implications for practice may be suggested as a result of this work.

(1) This research has laid the foundations to make surface irrigation decision support technology accessible to more users. It was identified early in the research that a likely reason why the technology hasn't been widely adopted is because the existing technology is complex and difficult to use. Therefore, a key goal of this research was to make the software as simple as possible to improve accessibility. This has been addressed using an *HTML*-browser-like interface combined with modern programming and interface design conventions. This provides multi-level and transparent functionality for different user types. Data requirements have been minimised and presented using progressive disclosure techniques, while the mathematical operation of the model has been hidden from the user. The five major tools required for surface irrigation decision support (database, simulation, calibration, optimisation and parameter analysis) have been encapsulated under a single familiar web-like interface. Actions required to initiate operations have been minimised. In theory, once the user has entered the irrigation input data, operations such as calibration, simulation, and optimisation processes can occur automatically and simultaneously with a single click of a button, and without user-intervention. Conveniently, the software is customisable through its *XML/XSLT* interface with an infinite range of interface configurations possible to account for different users.

(2) It has been recognised that using surface irrigation simulation technology encourages measurement and accountability of current practices. To use this technology, irrigators are required to measure how much water they apply, and to determine losses through estimation of infiltration and possibly runoff. The decision support system developed as a part of this research has the potential to further encourage this monitoring, through the inclusion of an inbuilt database for surface irrigation. This provides a simple tool to monitor irrigations in many paddocks over different irrigation seasons to measure the performance of current practices and improvements.

(3) The provision of the database in the decision support system provides a secondary benefit in helping to account for spatial and temporal variations of the field parameters in decision-making. Irrigations monitored over a season can be analysed to determine the range of infiltration variation and corresponding irrigation performance. The envelope of conditions can then be used as input in the software to generate guidelines for planning subsequent irrigations. This is undertaken through the inbuilt automatic guideline generation facilities. A benefit of this feature is that it could be used by consultants or extension officers to develop a set of simple to use, paper-based management charts to give to irrigators who may not have the capabilities to run the software.

(4) This research has provided tools to implement real-time management of furrow or border irrigation. For example, guidelines generated from irrigations over the previous season could be used to determine suitable estimates of flowrate and time-to-cutoff to perform the irrigation. Irrigation advance can then be monitored during the irrigation, and input into the decision support system while the irrigation is still occurring. The infiltration parameters can then be determined using the inbuilt parameter estimation facilities. The irrigation can then be terminated at a time suggested by the software to achieve optimum performance. This would typically involve cutting off the irrigation before the advance reaches the end of the field. While this type of analysis could be

undertaken with the existing modelling tools such as *SIRMOD* and *INFILT*, it would require considerable manipulation of the software in paddock. The new software would be able to perform these actions with a minimal effort – potentially by just entering the measured advance and inflow into the pre-configured database record, and clicking one button to estimate both the soil infiltration parameters and optimise the irrigation.

(5) The software provides a useful training tool for irrigators without having to resort to experimentation in the field. It offers a "computer game" like experience with graphical simulation animations, and automatic calculation and estimation facilities. Training courses for the *SIRMOD* and *INFILT* software run by the National Centre for Irrigation in Agriculture (http://www.ncea.org.au) have attracted considerable interest over the past few years. A major portion of this time was spent teaching users how to use the software. The simpler interface of the *FIDO* software would allow a greater focus upon management and design issues and utilise the inbuilt parameter estimation, optimisation and guideline generation facilities.

8.4 Limitations

Due to the range of subject areas presented in this dissertation, not all aspects of furrow and border irrigation decision support could be thoroughly treated. As explained in Chapter 1, this research has only focused upon the simplest forms of furrow and border irrigation in the context of Australian irrigation practices. However, provision has been made in the software for the future inclusion of more complex irrigation practices such as variable inflow irrigation, surge flow, and cutback irrigation. The spatial variability of field parameters such as furrow slope, furrow shape, hydraulic roughness, and soil infiltration has also been accounted for in the code but has not been implemented and tested.

The current version of the software (at the time of writing) has not been operationally released. While the program structure and mathematical algorithms have been repeatedly refined and validated, program navigation, database management, and presentation of results still needs further refinement. The operational efficiency of the algorithms could also be improved, with simulations often resorting to stability control measures to complete successfully, and optimisations often slower than desired. More extensive software documentation and tutorials also need to be developed before the software is released. Finally, the completed decision support system still requires rigorous testing and debugging.

8.5 Recommendations for further research and development

Hopefully, this research will lead to further interest in decision support development for surface irrigation. The software developed provides a stable and flexible platform from which many directions of research could be undertaken. This includes research in each of the major decision support component areas addressed in this dissertation: simulation engine development; solution of the inverse problem; automatic optimisation of design and management practices;

development of guidelines for design and management of furrow and border irrigation; and software engineering aspects of the decision support system.

Recommended areas of further research include:

- Measuring the usability, accessibility and performance of the *FIDO* decision support system under real field practice.
- Further investigation of parameter response and sensitivity analysis of furrow and border irrigation field parameters using the response-surface generation tools.
- Accounting for the spatial and temporal variation of the soil properties in terms of simulation, calibration and optimisation. This could also include investigating the aggregation of irrigation performance response-surfaces for different irrigation properties as a means for improving management and design practices.
- Including variable inflow methodologies (used in *SIRMOD* and *SRFR*) into the simulation engine in a robust, reliable manner, capable of implementation into the calibration, optimisation and response-surface generation tools.
- Improving the performance of the simulation engine in terms of robustness and iterative efficiency. That is, to ensure robustness without having to resort to emergency procedures to achieve success, and to decrease the number of iterations required to achieve convergence.
- Improving calibration and optimisation efficiency to reduce calculation times.
- Increasing the flexibility of the automatic guideline generation facility to plot iso-curves of design variables (with performance measures located on the axes) instead of performance measures.
- Investigating variations of field and furrow properties including non-linear furrow slopes, variable furrow shapes, and gradients of soil compaction as a means to improve irrigation performance.

8.6 Concluding postscript

This chapter has presented the key conclusions arising from this research in relation to the research hypothesis and research objectives. Both practical implications and limitations of this research have been identified, and recommendations for future research have been suggested.

In summary, this research has drawn from the efforts of numerous researchers over the last twenty years and cumulated in the creation of a stable flexible platform from which more advanced furrow and border irrigation research can be initiated. As well, this research has contributed to the development of a professional quality decision support system, which will hopefully improve the accessibility of decision support technologies to more users.

List of References

AARC / USDA / ARS 2006, *WinSRFR 1.0 User Manual*. U.S. Department of Agriculture, Agricultural Research Service, Arid-Land Agricultural Research Center 21881 N. Cardon Lane Maricopa, AZ 85239

Abbasi,F, Shooshtari, MM & Feyen, J 2003, 'Evaluation of Various Surface Irrigation Numerical Simulation Models', *Journal of Irrigation and Drainage Engineering*, 129(3): 208-213

Al-Azba, A & Strelkoff, T 1994, 'Correct form of Hall technique for border irrigation advance', *Journal of Irrigation and Drainage Engineering*, 120(2):292-307

Anthony, D 1995, 'On-farm water productivity, current and potential: options, outcomes, costs', *Irrigation Australia* 10,20-23

ASCE Task Committee 1993, 'Unsteady-flow modelling of irrigation canals*', Journal of Irrigation and Drainage Engineering*, 119(4):615-630

Austin, NR & Predgergast, JB 1997, 'Use of kinematic wave theory to model irrigation on cracking soil', *Irrigation Science*, 18:1-10

Australian Bureau of Statistics 2005, '4618.0 Water Use on Australian Farms', Australian Bureau of Statistics, Canberra, viewed 26 November 2005, <http://www.abs.gov.au/Ausstats>

Azevedo, CAV 1992, *Real-time solution of the inversion furrow advance problem*. Ph.D Dissertation, Utah State University

Azevedo, CAV, Merkley, GP, & Walker, WR 1996, 'The SIRTOM Software – A real-time surface irrigation decision support system', *Proc. Sixth International Conference of Computers in Agriculture*, Cancun, Mexico pp.872-884

Bassett, DL & Fitzsimmons, DW, 'Simulating overland flow in border irrigation', *American Society of Agricultural Engineers*, 19(4) 666-671

Bautista, E & Wallender, WW 1992, Hydrodynamic furrow irrigation model with specified space steps', *Journal of Irrigation and Drainage Engineering*, 118(3):450-465

Bautista, E & Wallender, WW 1993, 'Identification of furrow intake parameters from advance times and rates*', Journal of Irrigation and Drainage Engineering*, 119 (2):295-311

Bautista, E & Wallender, WW 1993, 'Numerical calculation of infiltration in furrow irrigation simulation models*', Journal of Irrigation and Drainage Engineering*, 119 (2):286-294

Bautista, E & Wallender, WW 1993, 'Optimal management strategies for cutback furrow irrigation*', Journal of Irrigation and Drainage Engineering*, 119 (6):1099-1113

Bautista, E & Wallender, WW 1993, 'Reliability of optimized furrow-infiltration parameters*', Journal of Irrigation and Drainage Engineering*, 119 (5):784-800

Boonstra, J & Jurriens, M 1988, BASCAD A mathematical model for level basin irrigation, ILRI pub. 43, ALTERRA, Wageningen, The Netherlands. 30p.

Bouwer, H 1957, 'Infiltration patterns for surface irrigation', *Agr. Eng.*, 38:662-664, 676

Brakensiek, DL, Heath, AL & Comer, GH 1966, 'Numerical technique for small watershed flood routing', *Agricultural Research Science* 41-113, USDA,

Burt, CM, Robb, GA & Hanon, A, 1982, 'Rapid evaluation of furrow irrigation efficiencies', *ASE Paper No, 82-2537*. Presented at the 1982 Winter Meeting of ASAE, Chicago, III

Cahoon, J 1998, 'Kostiakov infiltration parameters from Kinematic wave model*', Journal of Irrigation and Drainage Engineering*, 124(2);127-130

Calejo, MJ & de Sousa, PL 1996, 'Computer-aided evaluation of surface irrigation systems', *Proc. of Agricultural Engineering Conferenc*e, Madrid, Spain, 1996, 96C-057:10pp

Camacho, E, Perez-Lucena, C, Roldan-Canas, J & Alcaide, M 1997, 'IPE: Model for management and control of furrow irrigation in real time*', Journal of Irrigation and Drainage Engineering*, 123(4):264-269

Chadwick, A & Morfett, J 1986, *Hydraulics in Civil and Engineering*, Harper Collins Academic, London

Chaudhry, MH, Mays, LW 1993, *Computer modelling of free-surface and pressurized flows*, NATO ASI series, Series E, Applied Sciences

Chen, CL 1970, 'Surface irrigation using kinematic wave method', *J. Irrig. Drain. Div, ASCE* 96(IR1):39

Chen, CL, McCann, RC & Singh, VP 1981, 'Numerical solutions to the kinematic model of surface irrigation', *Tech. Rep. MSSU-EIRS-CE-81-1, Engineering and Industrial Research Station*, Mississippi State University, Mississippi, Stat, MS

Christiansen, JE, Bishop, AA, Kiefer, FW & Fok YS, 1966, 'Evaluation of Intake Rate Constants and Related to Advance of Water in Surface Irrigation*', Transactions of the ASAE*, Vol.9(5) pp 671-674

Clemmens, AJ 1979, 'Verification of the zero-inertia model for surface irrigation*', Transactions of the ASAE* 22(6 1306-1309

Clemmens, AJ 1981, 'Evaluation of infiltration measurements for border irrigation*', Agricultural Water Management*, 3:251-267

Clemmens, AJ 1982, 'Evaluating infiltration for border irrigation', *Agric. Water Management*, 3(4):251-267

Clemmens, AJ 1991, 'Direct solution to the surface irrigation advance inverse problem', *Journal of Irrigation and Drainage Engineering*, 117(4): 578-594

Clemmens, AJ 1992, 'Feedback control of a basin irrigation system', *Journal of Irrigation and Drainage Engineering*, 118(3):480-496

Clemmens, AJ & Keats, JB 1992, 'Bayesian inference for feedback control. I: Theory", *Journal of Irrigation and Drainage Engineering*, 118(3):416-432

Clemmens, AJ, Dedrick, AR & Strand, RJ 1995, 'WCL Report 19, BASIN – a computer program for the design of level-basin irrigation systems, version 2.0', U.S. Water Conservation Laboratory, Phoenix, Arizona. 55 pp. WCL# 1824

Coad, P & Nicola, J 1993, *Object-Oriented Programming*, Pearson Education; 1 edition

Cooper, A & Reimann RM 2003, *About Face 2.0: The essentials of interaction design*, Wiley, 1st edition

Courant, R Friedrichs, KO & Lewy, H 1928, 'Ueber die partiellen Differenzgleichungen der mathematische Physik', *Math Ann*, 100 pp 32-74.

Courant, R Friedrichs, KO & Lewy, H 1956, 'On the partial difference equations of mathematical physics', *NYO-7689*, Inst. Math. Sci. New York University (Translated from German)

Courant, R & Friedrichs, KO 1948, 'Supersonic flow and shock waves, *Interscience*

DeTar, WR 1989, 'Infiltration function from furrow stream advance*', Journal of Irrigation and Drainage Engineering*, 115(4):722-730

Dholakia, M, Misra, R & Zamam, MS 1998, 'Simulation of border irrigation system using explicit MacCormack finite difference method', *Agricultural Water Management*, 36:181-200

Elliott, RL & Eisenhauer, DE, 1983, 'Volume-balance techniques for measuring infiltration in surface irrigation', *ASAE Paper Non 83-2520*, Presented at the 1983 Winter Meeting of the ASAE Chicago III

Elliott, RL & Walker, WR 1982, 'Field evaluation of furrow infiltration and advance functions*', Transactions of the ASAE*, 396-400

Elliott, RL, Walker & WR, Skogerboe, GV 1983a, 'Furrow irrigation advance rates: a dimensionless approach*', Transactions of the ASAE*, 1722-1731

Elliott, RL, Walker, WR & Skogerboe, GV 1982, 'Zero-inertia modelling of furrow irrigation advance*', Journal of Irrigation and Drainage Engineering*, 108(IR3):179-195

Elliott, RL, Walker, WR & Skogerboe, GV 1983b, 'Infiltration parameters form furrow irrigation advance data*', Transactions of the ASAE*, 1726-1731

Enciso-Medina, J, Martin, D & Eisenhauer, D 1998, 'Infiltration model for furrow irrigation*', Journal of Irrigation and Drainage Engineering*, 124(2):73-80

Evans, RG, Smith, CJ, Mitchell, PD & Newton, PJ 1990, 'Furrow infiltration on nontilled beds with cracking soils*', Journal of Irrigation and Drainage Engineering*, 116(5):714-733

Fangmeier, DD & Ramsey, MK 1978, 'Intake characteristics of irrigation furrows*', Transactions of the ASAE*, 696-705

Fangmeier, DD & Strelkoff, T 1979, 'Mathematical models and border irrigation design', *American Society of Agric. Engineers*, 22(1) pp93-99

Finkel, HJ & Nir, D 1960, 'Determining infiltration rates in an irrigation border', *Journal of Geophysical Research*, 65(7):2125-2131

Fonken, DW, Carmody, T, Laursen, EM & Fangmeier, DD 1980, 'Mathematical model of border irrigation', *Journal of the Irrigation and Drainage Division*, *American Society of Civil Engineers*, 106(IR3):203-220

Garcia-Navarro, P & Saviron, JM 1992, 'McCormack's method for the

numerical simulation of one-dimensional discontinuous unsteady open channel flow*', Journal of Hydraulic Research*, 30(1):95-105

Garcia-Navarro, P, Priestley, A & Alcrudo, F 1994, 'An implicit method for water flow modelling in channels and pipes*', Journal of Hydraulic Research*, 32(5):721-742

Garcia-Navarro, P, Sanchez, A, Clavero, N & Playan, AM 2004, 'Simulation model for level furrows. II: Description, validation, and application', *Journal of Irrigation and Drainage Engineering*, 130(2):113-121

Gillies, MH, Smith, RJ & Raine, SR 2006, 'Accounting for temporal inflow variation in the inverse solution for infiltration in surface irrigation*', Irrigation Science* DOI 10.1007/s00271-006-0037-9

Gillies, MH, Smith, RJ 2005, 'Infiltration parameters from surface irrigation advance and run-off data', *Irrigation Science* 24:25-35

Gray, DM & Ahmed, M 1965, 'Rational Approach applied to the design of border Dyke systems', *Canadian Agricultural Engineering* (Ottawa), Vol 7:1 pp30-33, 44.

Green, WH & Ampt, CA 1911, 'Studies on soil physics, 1. Flow of air and water through soils', *J. Agric. Sci*, 4 1-24

Hall, WA 1956, 'Estimating irrigation border flow', Agr Engrg., 37(4), 263-265

Hall, WA 1960, 'Performance parameters of irrigation systems', *Transactions of the ASAE*, 3(1):75-76,81

Hanson, BR, Prichard, TL & Schulbach, H 1993, 'Estimating furrow infiltration*', Agricultural Water Management*, 24:281-298

Harder, JA & Armacost LV 1966, *Wave propagation in rivers*. Hydraulic *Engineering Laboratory Report No 1, Series 8*, University of California, Berkeley

Hardie, M, Newell, G & Raine, S 2002, 'Effect of Sugarcane trash retention systems on furrow irrigation performance*', Proc. National Conference, Irrigation Association of Australia*, 21st–24th May, Sydney. pp311-320

Haverkamp, R, Parlange, JY, Star, JL, Schmitz, G & Fuentes, C 1990, 'Infiltration under ponded conditions. 3: A predictive equation base on physical parameters', *Soil Science*, 149(4), 292-300

Henderson, FM & Wooding, RA 1964, 'Overland flow and groundwater flow from a steady rainfall of finite duration', *J. of Geophysical Research, Am. Geophysical Union*, 69(8):1531-1540

Hibbs, RA, James, LG & Cavalieri, RP 1992 A furrow irrigation automation system utilizing adaptive control*', Transactions of the ASAE*, 35(3):1063-1067

Hodgson, AS, Constable, GA, Duddy, GR & Daniells, IG 1990 A comparison of drip and furrow irrigated cotton on a cracking clay soil: 2. Water use efficiency, waterlogging, root distribution and soil structure*', Irrigation Science*, 11:143-148

Hoffman, GJ & Martin, DL 1993 Engineering systems to enhance irrigation performance*', Irrigation Science*, 14:53-63

Holden, AP & Stephenson, D 1995 Finite difference formulations of kinematic equations', *Journal of Irrigation and Drainage Engineering*, 121(5):423-426

Holden, AP & Stephenson, D 1998 Improved four-point solution of the kinematic equations*', Journal of Hydraulic Research*, 26(4):413-423

Holzapfel, EA & Marino, MA 1987 Surface irrigation nonlinear optimization models*', Journal of Irrigation and Drainage Engineering*, 113(3 379-392

Holzapfel, EA, Marino, MA & Chavez-Morales, J 1984 Border irrigation model selection*', Transactions of the ASAE*, 1811-1816

Holzapfel, EA, Marino, MA & Chavez-Morales, J 1984 Comparison and selection of furrow irrigation models*', Agricultural Water Management*, 9:105-125

Holzapfel, EA, Marino, MA & Chavez-Morales, J 1986 Surface irrigation optimization models', *Journal of Irrigation and Drainage Engineering*, 112(1 pp1-19

Holzapfel, EA, Marino, MA, Valenzuela, A & Diaz, F 1988 Comparison of infiltration measuring methods for surface irrigation*', Journal of Irrigation and Drainage Engineering*, 114(1):130-142

Hornbuckle, JW, 1999, *Modelling furrow irrigation on heavy clays, high water tables, and tiled drain soils using SIRMOD in the Murrumbidgee irrigation area.* Undergraduate Thesis. University of New England. Armidale, NSQ. Australia.

Hornbuckle, JW, Christen, EW & Faulkner, RD 2003, 'Improving the efficiency and performance of furrow irrigation using simulation modelling in south-eastern Australia*', Proc. International Workshop on Improved Irrigation Technologies and Methods: Research, Development and Testing*, 18-19 September, Montpellier, France.

Hornbuckle, JW, Christen, EW & Faulkner, RD 2005, 'Use of SIRMOD as a Quasi Real Time Surface Irrigation Decision Support System, *Proc. MODSIM International Conference*, Melbourne, 12-15 December

Horton, RE 1940, 'An approach towards a physical interpretation of infiltration capacity', *Soil Sci. Sol Am., J*. 5 pp399-417

Hume, IH 1993, 'Determination of infiltration characteristics by volume-balance for border check irrigation*', Agricultural Water Management*, 23:23-29

Ismail, SM & Depeweg, H 2005, 'Simulation of continuous and surge flow irrigation under short field conditions', *Irrigation and Drainage*, 54(2):217-230

Ito, H, Wallender, WW & Raghuwanshi, NS 1999, 'Economics of furrow irrigation under partial infiltration information*', Journal of Irrigation and Drainage Engineering*, 125(3 pp105-110

Izadi, B, Heermann, DF & Duke, HR 1988, 'Sensor placement for real time infiltration parameter evaluation', *Trans. Am. Soc. Of Agric. Engrs*, 31(4 1159-1166

Jain, SK & Singh, VP 1989, 'A numerical kinematic wave model for border irrigation*', Irrigation Science*, 10:253-263

Jurriëns, M 2001, *SURDEV: surface irrigation software : design, operation, and evaluation of basin, border, and furrow irrigation*. Wageningen : International Institute for Land Reclamation and Improvement/ILRI, 194 p

Kafshgiri, VMR 1984, *A non-linear zero-inertia model for surge flow furrow irrigation*. Dissertation submitted towards Ph.D, University of California, 186pp

Katopodes, ND & Strelkoff, T 1977, 'Dimensionless solutions of border-irrigation advance*', Journal of Irrigation and Drainage Engineering*, 103(IR4):401-417

Katopodes, ND & Strelkoff, T 1977, 'Hydrodynamics of border irrigation - complete model*', Journal of Irrigation and Drainage Engineering*, 103(IR3):309-323

Katopodes, ND & Tang, JH 1990, 'Self-adaptive control of surface irrigation advance*', Journal of Irrigation and Drainage Engineering*, 116(5):697-713

Katopodes, ND 1990, 'Observability of surface irrigation advance*', Journal of Irrigation and Drainage Engineering*, 116(5):656-675

Katopodes, ND 1994, 'Hydrodynamics of surface irrigation: vertical structure of the surge front*', Irrigation Science*, 15:101-111

Katopodes, ND, Tang, JH & Clemmens, AJ 1990, 'Estimation of surface irrigation parameters*', Journal of Irrigation and Drainage Engineering*, 116(5):676-696

Khatri, KL & Smith, RJ 2005, 'Evaluation of methods for determining infiltration parameters from irrigation advance data', I*rrigation and Drainage*, 54(4 pp467-482

Khatri, KL, Smith, RJ & Raine, SR, 2006, 'Real time control of surface irrigation: managing infiltration variations and enhancing furrow irrigation performance*', Proc. National Conference, Irrigation Association of Australia*. 9-11 May, Brisbane. P73-4

Knight, JH 1980, 'An improved solution for the infiltration-advance problem in irrigation hydraulics*', Proc. 7th Australasian Hydraulics and Fluid Mechanics Conference*, Brisbane, 18-22 August, 1980, 258-261

Kostiakov, AN 1932, *On the dynamics of the coefficients of water percolation in soils*. Sixth Commission, Int. Soc. *Soil Science*, Part A 15-31.

Lal, R & Pandya, AC 1972, 'Volume-balance method for computing infiltration rates in surface irrigation*', Transactions of the ASAE*, 69-72

Latimer, EA & Reddell, DL 1989, 'A real time automated control system for surface irrigation', *Land and Water Use*, 613-620

Latimer, EA & Reddell, DL 1990, 'Components for an advance rate feedback irrigation system (ARFIS)*', Transactions of the ASAE*, 33(4):1162-1170

Levien, SLA & de Souza, F 1987, 'Algebraic computation of flow in furrow irrigation', *Journal of the Irrigation and Drainage Division*, Vol 113, 367-377

Lewis MR 1937, 'The rate of infiltration of water in irrigation practice', *Trans Am Geophys Union 18th Annual meeting*, 361-368.

Lewis, MR & Milne, WE 1938, 'Analysis of border irrigation', *Agr. Eng*., 19:267-272

Liggett, A & Cunge, JA, 1975, 'Numerical methods of solution of the unsteady flow equations', Chapter 4 of *Unsteady Flow in Open Channels*, Water Resources Publication, Fort Collins, Colorado.

Lighthill, MH & Whitman, RB 1955, 'On kinematic Waves: I. Flood movement in long rivers', *Proc. Royal Soc London, Series A*, Vol. 229:201-316

Ma, AD 2004, *Computer aided mapping of spatial and temporal infiltration variability, Undergraduate Thesis*, University of Southern Queensland

MacCormack, RW & Warming, RF 1973, 'Survey of computational methods for three-dimensional supersonic inviscid flows with shocks. *Advances in numerical fluid dynamics, AGARD lecture series 64*, Brussels, Belgium, ppp5.1-5.19

Maheshware, BL & Patto, MJ 1990, 'Present status of border irrigation design in Australia*', Proc. Conference on Agricultural Engineering*, Toowoomba. Institution of Engineers, Australia. 156-159

Maheshwari, BL & Jayawardane, NS 1992, 'Infiltration characteristics of some clayey soils measured during border irrigation*', Agricultural Water Management*, 21:265-279

Maheshwari, BL & McMahon, TA 1991, 'BICADM. A software package for border irrigation computer aided design and management', Dept. Cvil and Ag. Engr. University of Melbourne, Melbourne Australia. 32p.

Maheshwari, BL & McMahon, TA 1993, 'Performance evaluation of border irrigation models for south-east Australia: Part 1, advance and recession characteristics', *J. agric. Engng Res*., 54:67-87

Maheshwari, BL & McMahon, TA 1993, 'Performance evaluation of border irrigation models for south-east Australia: Part 2, overall suitability for field applications', *J. agric. Engng Res*., 54, 127-139

Maheshwari, BL 1994, 'Development of a regression-based model of border irrigation on cracking soils*', Agricultural Water Management*, 25: 167-178

Maheshwari, BL, Turner, AK & McMahon, TA 1986, 'Mathematical modelling of border irrigation*', Proc. Conference on Agricultural Engineering*, Adelaide 24-28 August 1986, 215-218

Maheshwari, BL, Turner, AK, McMahon, TA & Campbell, BJ 1988, 'An optimization technique for estimating infiltration characteristics in border irrigation*', Agricultural Water Management*, 13:13-24

Maihol, JC & Gonzalez, JM 1993, 'Furrow irrigation model for real-time applications on cracking soils*', Journal of Irrigation and Drainage Engineering*, 119(5):768-783

McClymont, DJ & Smith, RJ 1996, 'Infiltration parameters from optimisation on furrow irrigation advance data', *Irrigation Science*, 17(1): 15-22.

Merriam, JL & Clemmens, AJ 1985, 'Time rate infiltration depth families', Proc. Development and Management Aspects of Irrig & Drainage,Spec. Conf Proc. Irrig. And Drain. Div.., ASCE, San Antonio, Texas, pp67-74

Merriam, JL & Keller, J 1978*. Farm irrigation system evaluation: A guide for management.* Department of Agricultural and Irrigation Engineering, Utah State University, Logan, Utah.

Norum, DI & Gray, DM 1970, 'Infiltration equations from rate-of-advance data', *Journal of Irrigation and Drainage Division, Proceeding of the American Society*

of Civil Engineers, 96(IR2):111-119

Or, D & Silva, HR 1996, 'Prediction of surface irrigation advance using soil intake properties*', Irrigation Science*, 16:159-167

Ottoni, TB & Warrick, AW, 1983, 'Identification of infiltration parameters using border irrigation tests', *ASAE* Pap 83-2519

Oweis, TY & Walker, WR 1990, 'Zero-inertia model for surge flow furrow irrigation*', Irrigation Science*, 11:131-136

Philip, JR & Farrell, DA 1964, 'General solution of the infiltration-advance problem in irrigation hydraulics', *J. Geohpys. Res*. 69 pp. 621-631.

Philip, JR 1957a, 'Theory of infiltration, 4. Sorptivity and algebraic infiltration equations', *Soil Science*, 84,257-264

Philip, JR 1957b, 'Theory of infiltration, 5. The influence of the initial moisture content', *Soil Science*, 84,329-339

Philip, JR 1957c, 'Theory of infiltration, 2. The profile at infinity', *Soil Science*, 83,434-448

Philip, JR 1969, *Theory of infiltration. Advances in hydroscience, vol. 5*, K.T.Choe, ed., Academic Press, New York, 215-296

Raes, D, Sahli, A, Van Looij, J, Ben Mechlia, N & Persoons, E 2000, 'Charts for guiding irrigation in real time', *Irrigation and Drainage systems*, 14(4 pp343-352

Raghuwanshi, NS & Wallender, WW 1997, 'Economic optimisation of furrow irrigation*', Journal of Irrigation and Drainage Engineering*, 123(5):377-

Raghuwanshi, NS & Wallender, WW 1999, 'Forecasting and optimising furrow irrigation management decision variables*', Irrigation Science*, 19(1) pp1-6

Raine, SR & Bakker, D 1996, 'Increased furrow irrigation efficiency through better design and management of cane fields", *Proc. Aust. Soc. Sugar Cane Technol.* 18:119-124

Raine, SR & Shannon, EL 1996, 'Increasing the irrigation efficiency of Burdekin canegrowers through participatory action learning*', Proc. Irrigation Australia Conference "Australian Solutions,"* Adelaide May 14-16, 1996, 36:1-6

Raine, SR & Walker, WR 1998, 'A decision support tool for the design, management and evaluation of surface irrigation systems*', Proc. National Conference, Irrigation Association of Australia*, 19-21 May, Brisbane. pp117-123

Raine, SR, Holden, JR & Shannon, EL 1996, 'Getting the message across in the battle for irrigation efficiency*', Proc. Conference on Engineering in Agriculture and Food Processing* 1996, University of Queensland, Gatton, 25(3):SEAg96/092

Ram, RS & Singh, VP 1985, 'Application of kinematic wave equations to border irrigation design', *J. agric. Engng Res.*, 32:57-71

Ram, RS, Singh, VP & Prasad, SN 1986a, 'A quasi-steady state integral model for closed-end border irrigation*', Agricultural Water Management*, 11:39-57

Ram, RS, Singh, VP & Prasad, SN 1986b, 'A quasi-steady state integral model

for border irrigation*', Irrigation Science*, 7(2) pp 113-141

Rasmussen, WO 1994, 'Infiltration-advance equation for finite linear source*', Journal of Irrigation and Drainage Engineering*, 120(4):796-812

Rayej, M & Wallender, WW 1985, 'Furrow irrigation simulation time reduction*', Journal of Irrigation and Drainage Engineering*, 111(2):134-146

Rayej, M & Wallender, WW 1987, 'Furrow model with specified space intervals*', Journal of Irrigation and Drainage Engineering*, ASCE 113, 536-548

Rayej, M, Wallender, WW 1988, 'Time solution of kinematic-wave model with stochastic infiltration*', Journal of Irrigation and Drainage Engineering*, 114(4):605-621

Reddell, DL 1981, *Modified rate-of-advance method for an automatic furrow irrigation system. Paper No. 81-2552*, Am. Soc. Of Agric. Engrs. , St Joseph, Mich.

Reddell, DL & Latimer, EA 1987, Field evaluation of an advance rate feedback irrigation system, *Irrigation Systems for the 21st Century, Proc. if Irrigation and Drainage Divisions of the ASCE*, Portland, 317-324

Reddy, JM & Apolayo, HM 1991, 'Sensitivity of furrow irrigation system cost and design variables*', Journal of Irrigation and Drainage Engineering*. 117(2), pp201-219

Reddy, JM & Clyma, W 1989, 'Discussion on "Surface Irrigation nonlinear optimisation models" by Holzapfel, EA, and Marino, MA*', Journal of Irrigation and Drainage Engineering*, 115(5 897-898

Reddy, JM & Clyma, W, 1981, 'Optimal design of border irrigation systems*', Journal of Irrigation and Drainage Engineering*, 107(IR3) pp 289-306

Reddy, JM & Singh, VP 1994, 'Modelling and error analysis of kinematic-wave equations of furrow irrigation*', Irrigation Science*, 15:113-122

Reddy, JM 1994, 'Optimization of furrow irrigation system design parameters considering drainage and runoff water quality constrains*', Irrigation Science*, 15:123-136

Renault, D & Wallender, WW 1990, 'Alive (Advance Linear Velocity): Surface irrigation rate balance theory*', Journal of Irrigation and Drainage Engineering*, 118(1):138-155

Renault, D & Wallender, WW 1991, 'ALIVE- Advance Linear Velocity: Horton infiltration law from field water advance rate*', Transactions of the ASAE*, 34(4):1706-1714

Renault, D & Wallender, WW 1994, 'Furrow advance-rate solution for stochastic infiltration properties*', Journal of Irrigation and Drainage Engineering*, 120(3)617-633

Riel, AJ 1996, *Object-Oriented Design Heuristics*, Addison-Wesley Professional

Ross, PJ 1987, *Two numerical models for furrow and border irrigation*, Technical Memorandum, CSIRO Div. Soils, Davies Lab. Townsville, QLD, Australia, 8pp

Sakkas, JG, Bellos, CV & Klonaraki, MN 1994, 'Numerical computation of surface irrigation*', Irrigation Science*, 15:83-99

Scaloppi, EJ 1984, 'A method for evaluating infiltration parameters in surface irrigation', *Trans. 12 Congress on Irrigation and Drainage*, 28 May – 2 June 1984), Fort Connins, CO. International Commission on Irrigation and Drainage, New Delhi, Vol.1(B), pp 369-378.

Scaloppi, EJ, Merkley, GP & Willardson, LS 1995, 'Intake parameters from and advance and wetting phases of surface irrigation', *Journal of Irrigation and Drainage Engineering*, 121(1), 57-70

Schmitz, G, Haverkamp, R & Palacios, O 1985, 'A coupled surface-subsurface model for shallow water flow over initially dry soil', Proc, 21st Congr. IAHR, Int. Assoc. for Hydr. Res., 1, 23-30

Schmitz, GH & Seus, GJ 1989, 'Analytical model of level basin irrigation*', Journal of Irrigation and Drainage Engineering*, 115(1):78-95

Schmitz, GH & Seus, GJ 1990, 'Mathematical zero-inertia modelling of surface irrigation: advance in borders*', Journal of Irrigation and Drainage Engineering*, 116(5):603-615

Schmitz, GH & Seus, GJ 1992, 'Mathematical zero-inertia modelling of surface irrigation: Advance in furrows*', Journal of Irrigation and Drainage Engineering*, 118(1):1-18

Schwankl, LJ & Wallender, WW 1987, 'Furrow modelling with variable hydraulic characteristics', *Proc. Specialty Conference sponsored by the Irrigation and Drainage Division, ASCE*, pp753-760

Schwankl, LJ & Wallender, WW 1988, 'Zero-inertia furrow modelling with variable infiltration and hydraulic characteristics*', Transactions of the ASAE*, 31(5):1470-1475

Shayya, WH, Bralts, VF & Segerlind, LJ 1993, 'Kinematic-wave furrow irrigation analysis: a finite element approach*', Transactions of the ASAE*, 36(6):1733-1742

Shepard, JS, Wallender, WW & Hopmans, JW 1993, 'One-point method for estimating furrow infiltration*', Transactions of the ASAE*, 36(2):395-404

Sherman, B & Singh, VP 1978, 'A kinematic-wave model for surface irrigation', *Water Res Res* 14:357-364

Sherman, B & Singh, VP 1982, 'A kinematic-wave model for surface irrigation: an extension', *Water Res Res*, 18:659-657

Singh, P & Chauhan, HS 1973, 'Determination of water intake rate from rate of advance', *Transactions of the ASAE*, 16(6) June, 1081-1084

Singh, V & Bhallamudi, SM 1996, 'Complete hydrodynamic border-strip irrigation model', Journal of Irrigation and Drainage Engineering, 122(4) 189-197

Singh, V & Bhallamudi, SM 1997, 'Hydrodynamic modelling of basin irrigation*', Journal of Irrigation and Drainage Engineering*, 123(6):407-414

Singh, VP & He, YC 1988, 'Muskingum model for furrow irrigation*', Journal of Irrigation and Drainage Engineering*, 114(1):89-103

Singh, VP & Joseph, ES 1994, 'Kinematic-wave model for soil-moisture movement with plant-root extraction*', Irrigation Science*, 14:189-198

Singh, VP & Ram, RS 1983, 'A kinematic model for surface irrigation" verification by experimental data', *Water Res Res,* 19(6) 1599-1612

Singh, VP & Ram, RS 1984, 'Solution of the Kinematic-wave equations for border irrigation', *Agricultural Water Management*, Vol 9(2):127-138

Singh, VP & Sherman, B 1983, 'A kinematic study of surface irrigation: mathematical solutions', *Tech. Report WRR4*, Water Resources Program, Department of Civil Engineering, Louisiana State University

Singh, VP & Yu, FX 1987, 'A mathematical model for border irrigation I. Advance and storage phases', *Irrigation Science*, 8(3) 151-174

Singh, VP & Yu, FX 1987, 'A mathematical model for border irrigation II. Vertical and horizontal recession phases', *Irrigation Science*, 8(3) 175-190

Singh, VP & Yu, FX 1987, 'A mathematical model for border irrigation III. Evaluation of Models', *Irrigation Science*, 8(3) 191-213

Singh, VP & Yu, FX 1990, 'Derivation of infiltration equation using systems approach*', Journal of Irrigation and Drainage Engineering*, 116(6):837-858

Singh, VP, Aravamuthan, V & Joseph, ES 1994, 'Errors of kinematic-wave and diffusion-wave approximations for time-independent flow in infiltration channels*', Irrigation Science*, 15:137-146

Singh, VP, Feyen, J & Persoons, E 1987 A model for optimizing management variables in close end borders - development and evaluation'

Singh, VP, Scarlatos, PD & Prasad, SN 1990, 'An improved Lewis-Milne equation for the advance phase of border irrigation*', Irrigation Science*, 11:1-6

Singh, VP, Scarlatos, PD & Raudales, SA 1988, 'Muskingum model for border irrigation*', Journal of Irrigation and Drainage Engineering*, 114(2):266-280

Smerdon, ET, Blair, AW & Reddell, DL 1988, 'Infiltration from irrigation advance data. I: Theory*', Journal of Irrigation and Drainage Engineering*, 114(1):4-17

Smith, RE 1972, 'Border irrigation advance and ephemeral flood waves', *Journal Irrig. and Drain. Div., ASCE*, 98(2) 289-305

Smith, RJ, 1993, 'Infiltration parameters from irrigation advance data', Proc. Intern. Cong. Modelling and Simulation, University of Western Australia, Perth 4: 1569-1574

Souza, F 1981, *Nonlinear hydrodynamic model of furrow irrigation*, Ph.D. Dissertation, Department of Land, Air and Water Resources, University of California at Davis

Stephenson D & Meadows ME, 1986, *Kinematic Hydrology and Modelling*, Elsevier, Amsterdam, pp23-8

Strelkoff, T & Katopodes, ND 1977, 'Border-irrigation hydraulics with zero inertia', *Journal of the Irrigation and Drainage Division, American Society of Civil Engineers*, 103(IR3):325-342

Strelkoff, T & Shatanawi, MR 1984, 'Normalised graphs of border-irrigation performance*', Journal of Irrigation and Drainage Engineering*, 110(4):359-374

Strelkoff, T 1977, 'Computation of flow in border irrigation*', Journal of Irrigation and Drainage Engineering*. Vol 103, no3, pp357-377

Strelkoff, T 1985, 'Dimensionless formulation of furrow irrigation*', Journal of Irrigation and Drainage Engineering*, 111(4):380-394

Strelkoff, T 1992, 'EQSWP: Extended unsteady-flow double sweep equation solver', *Journal of Hydraulic Engineering*, 118(5):735-742

Strelkoff, T, Adamsen, FJ, Bautista, E, Hunsaker, DJ, Clemmens, AJ, Strand, RJ & Tabbara H, 2001, *Surface Irrigation Water Quality and Management*,

Strelkoff, TS & Clemmens, AJ 1994, 'Dimensional analysis in surface irrigation*', Irrigation Science*, 15:57-82

Strelkoff, TS, Clemmens, AJ, El-Ansary, M, and Awad, M 1999, 'Surface-irrigation evaluation models: applications to level basins in Egypt.' Transactions of the ASAE 42(4):1027-1036

Strelkoff, TS & Clemmens, AJ 2001, 'Data-driven organization of field methods for estimation of soil and crop hydraulic properties.' Paper number 012256, *ASAE Annual Meeting* 2001.

Strelkoff, TS & Falvey, HT 1993, 'Numerical methods used to model unsteady canal flow*', Journal of Irrigation and Drainage Engineering*, 119(4):637-655

Strelkoff, TS 1990, *SRFR. A computer program for simulating flow in surface irrigation. Furrows – Basins – Borders.* U.S. Water Conservation Laboratory, 4332 East Broadway, Phoenix AZ. WCL Report #17. 75pp. December.

Strelkoff, TS, Clemmens, AJ & Schmidt, BV & Slosky, EJ 1996, *WCL Report 21 – BORDER – A design and management aid for sloping border irrigation systems*. 1-44

Strelkoff, TS, Clemmens, AJ & Schmidt, BV 1998, *SRFR, Version 3.31 – A model for simulating surface irrigation in borders, basins and furrows*, USWCL, USDA/ARS, 4332 E. Broadway, Phoenix, AZ.

Tabauda, MA, Rego, ZJC, Vachaud, G & Pereira, LS 1995, 'Modelling of furrow irrigation. Advance with two-dimensional infiltration*', Agricultural Water Management*, 28:201-221

Tamimi, AH, Clemmens, AJ, Bautista, E & Strelkoff, T 2003, *SIPES – Surface Irrigation Parameter Estimation Software.* United States Committee of Irrigation and Drainage Engineering Conference P. 615-624

Tidwell, J 2005, *Designing Interfaces*, O'Reilly Media

Tognazzini, B 1992, *Tog on Interface*, Addison-Wesley Professional

Trout, TJ 1990, 'Furrow inflow and infiltration variability impacts on irrigation management*', Transactions of the ASAE*, 33(4):1171-1178

Turbak, AS & Morel-Seytoux, HJ 1988, 'Analytical solutions for surface irrigation. I: Constant infiltration rate', Journal of Irrigation and Drainage Engineering, 114(1) 31-47

Turner, AK & Clift, TR 1984, *Advance and recession in border check irrigation for flat, clayey soils.* Agricultural Engineering Report, University of Melbourne, 71/84

Turral, H & Malano, HM 1996, 'A recirculating infiltrometer to investigate surge flow in border irrigation on cracking soils - reconciling field and infiltrometer measurements*', Proc. Conference on Engineering in Agriculture and Food*

Processing 1996, University of Queensland, Gatton, 25(3):SEAg96/037

USDA 1986, *Surge flow irrigation field guide,* USDA Soil Conservation Service, 88pp

USDA 2006, *Part 623 Irrigation National Engineering Handbook*, USDA Natural Resources Conservation Service, 115pp

Valiantzas, JD 1993, 'Border advance using improved volume-balance model*', Journal of Irrigation and Drainage Engineering*, 119(6):1006-1013

Valiantzas, JD 1994, 'Simple method for identification of border infiltration and roughness characteristics*', Journal of Irrigation and Drainage Engineering*, 120(2):233-249

Valiantzas, JD 1997a, 'Surface irrigation advance equation: variation of subsurface shape factor*', Journal of Irrigation and Drainage Engineering*, 123(4):300-306

Valiantzas, JD 1997b, 'Volume-balance irrigation advance equation: Variation of surface shape factor*', Journal of Irrigation and Drainage Engineering*, 123(4):307-312

Valiantzas, JD 1999, 'Explicit time of advance formula for furrow design', *Journal of Irrigation and Drainage Engineering*, 125(1):19-25

Valiantzas, JD 2000a, 'Discussion of "estimation of surface volume in hydrological models for border irrigation" by J. Monserrat and J. Barragan' *Journal of Irrigation and Drainage Engineering*, 126(2), 131-133

Valiantzas, JD 2000b, 'Surface water storage independent equation for predicting furrow irrigation advance', Irrigation Science, 19:115-123

Valiantzas, JD 2001a, 'Optimal furrow design. I: Time of Advance Equation, *Journal of Irrigation and Drainage Engineering*, 127(4):201-207

Valiantzas, JD 2001b, 'Optimal furrow design. II: Explicit calculation of design variables', *Journal of Irrigation and Drainage Engineering*, 127(4):209-215

Valiantzas, JD, Aggelides, S & Sassalou, A 2001, 'Furrow infiltration estimation from time to a single advance point*', Agricultural Water Management*, 52: 17-32

Victorian DPI, 2004, *Shepparton Irrigation Region Implementation Committee, Research Reporting Day*. DPI Tatura, May 2004

Vogel, T & Hopmans, JW 1992, 'Two-dimensional analysis of furrow infiltration', *Journal of Irrigation and Drainage Engineering*, Vol 118, no. 5, pp791-806

Walker, WR 1993, *SIRMOD – Surface Irrigation Simulation Software*, Utah State University, Logan Utah

Walker, WR 1989, 'Guidelines for designing and evaluating surface irrigation systems', *Irrigation and Drainage paper 45*, Food and Agriculture Organization of the United Nations, Rome, Italy

Walker, WR & Busman, JD 1990, 'Real time estimation of furrow irrigation', *Journal of Irrigation and Drainage Engineering*, 116(3):299-318

Walker, WR & Humpherys, AS 1983, 'Kinematic-wave furrow irrigation model', *Journal of Irrigation and Drainage Engineering*, 109(4):377-392

Walker, WR & Skogerboe, GV 1987, *Surface Irrigation Theory and Practice,* Prentice-Hall, New York

Walker, WR 2005, 'Multi-level calibration of furrow infiltration and roughness', *Journal of Irrigation and Drainage Engineering*, 131(2) pp129-135

Walker, WR 2003, *SIRMOD III – Surface irrigation simulation, evaluation, and design. Guide and technical documentation,* Dept. of Biological and Irrigation Engineering, Utah State University, Logan, Utah

Wallender, WW 1986, 'Furrow model with spatially varying infiltration', *Trans. The Amer. Soc. Of Agric. Engineers*, 29(4), 1012-1016

Wallender, WW & Rayej, M 1985, 'Zero-inertia surge model with wet-dry advance', *Transactions of the ASAE*, 28(5);1530-1534

Wallender, WW & Rayej, M 1990, 'Shooting method for Saint Venant equations of furrow irrigation', *Journal of Irrigation and Drainage Engineering*, 116(1):114-122

Wallender, WW & Rayej, M, 1987, 'Economic optimization of furrow irrigation with uniform and nonuniform soil', *Transactions of the ASAE* 30(5):1425-1429

Wallender, WW & Yokokura, J 1991, 'Space solution of Kinematic-Wave model by time iteration', *Journal of Irrigation and Drainage Engineering*, 117(1):140-144

Wallender, WW (1986), 'Furrow model with spatially varying infiltration', *Transactions of the ASAE*, 29(4):1012-1016

Wallender, WW 1989, 'Economic surface irrigation within environmental constraints', *Proc. Southampton Conference*, 5.9:507-513

Wallender, WW, Ardila, S & Rayej, M, 1990, 'Irrigation optimization with variable water quality and nonuniform soil', *Transactions of the ASAE* 33(5): pp1605-1611

Wilke, OC & Smerdon, ET, 1961, 'A solution of the irrigation advance problem', *Journal of Irrig. Drain. Div. ASCE*, 91(5),2334

Wilke, OC, 1968, 'A hydrodynamic study of flow in irrigation furrows', *Report TR-13*, Texas Water Resources Institute, Texas A&M University

Wilson, BN & Elliott, RL 1988, 'Furrow advance using simple routing models', *Journal of Irrigation and Drainage Engineering*, 114(1):104-117

Wood, M, Malano, H & Turral, H, 1998, *Real-time monitoring and control of on-farm surface irrigation systems*', Final Report, Department of Civil and Environmental Engineering, University of Melbourne.

Wooding, RA & Henderson, FM 1965a, 'A hydraulic model for the catchment-stream problem. 1. Kinematic wave theory', J. of Hydrology, 3(3):254-267

Wooding, RA & Henderson, FM 1965b, 'A hydraulic model for the catchment-stream problem. 2. Numerical solutions', J. of Hydrology, 3(3):268-282

Wu, I & Bishop, AA 1970, 'Graphic relation of intake, length-of-run and time', *Journal of the Irrigation and Drainage Division, American Society of Civil Engineers*, 96(IR3):233-240

Yost, SA & Katopodes, ND 1998, 'Global identification of surface irrigation

parameters', *Journal of Irrigation and Drainage Engineering*, 124(3):131-139

Yu, FX & Singh, VP 1989, 'Analytical model for border irrigation', *Journal of Irrigation and Drainage Engineering*, 115(6):982-999

Yu, FX & Singh, VP 1990, 'Analytical model for furrow irrigation', *Journal of Irrigation and Drainage Engineering*, 116(2):154-171

Zerihun, D, Feyen, J 1992 *FISDEV A software package for design and evaluation of furrow irrigation systems*. Centre for irrigation Engineering, Katholieke Universiteit, Lueven, Belgium. 54p

Zerihun, D, Feyen, J & Reddy, JM 1996, 'Sensitivity analysis of furrow-Irrigation performance parameters', *Journal of Irrigation and Drainage Engineering*, 122(1):49-57

Zerihun, D, Mohan Reddy, J, Feyen, J & Breinburg, G 1993, 'Design and management nomograph for furrow irrigation', *Irrigation and Drainage Systems*, 7(1) pp29-41.

Appendix 2.1 Derivation of the Saint Venant Equations with lateral inflows and outflow

The following one-dimensional unsteady flow equations are derived through application of the principles of conservation of mass and momentum to an elemental control volume of fluid. This derivation is given for completeness as many texts fail to state the assumptions underlying the derivation, are not thorough in the derivation and neglect to include lateral inflows and outflows.

A2.1.1 Assumptions

- Flow is a translatory wave motion of long wavelength and low amplitude (Chadwick and Morfett, 1986 implying that the streamlines can be considered parallel.
- The water surface profile varies gradually implying a hydrostatic pressure distribution with negligible vertical accelerations.
- The velocity of the streamflow across the cross-sectional area of flow can be considered sufficiently uniform and can be represented by the average cross-sectional velocity.
- The effects of momentum for the lateral inflow and outflow can be considered negligible.
- Steady flow formula can be used to approximate resistance to flow.
- The channel is rectangular of unit width and its slope is small.

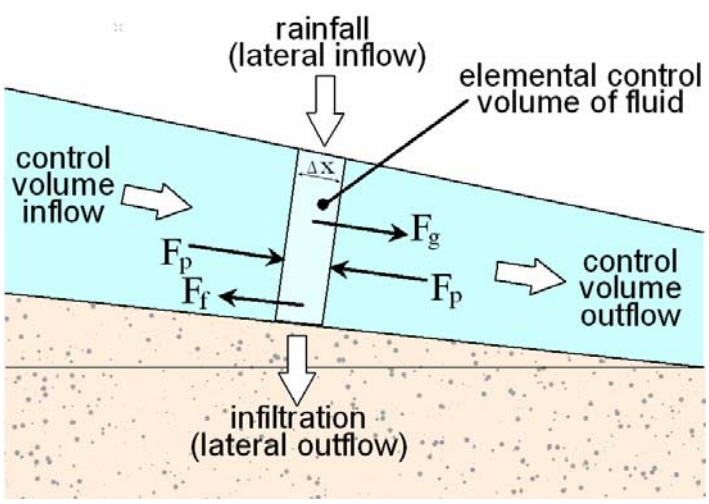


Figure A2.1.1: Elemental control volume in open channel flow

A2.1.2 Continuity Equation

The principle of conservation of mass implies that for any time interval, the difference between the mass flow entering and the mass flow exiting an elemental control volume, is equal to the change of mass within the control volume. In terms of flowrate, the difference between the inflow and outflow for a control volume is equal to the rate of change of storage over the time interval. Therefore, over a fixed time interval ∂t;

$$inflows - outflows = rate\ of\ \ change\ in\ storage\ with\ time \quad (1)$$

$$q_{in} + r\Delta x - q_{out} - i\Delta x = \frac{\partial y}{\partial t}\Delta x \quad (2)$$

where q_{in} (m^3/s) is inflow into the control volume, r (m^2/s) is rainfall intensity (lateral inflow), q_{out} (m^3/s) is outflow from the control volume, i (m^2/s) is the infiltration rate (lateral outflow), Δx (m) is the thickness of the control volume, $\partial q / \partial x$ is the change in inflow over the control volume width, and $\partial y / \partial t$ is the change of depth with time. We can relate q_{out} in terms of q_{in} using;

$$q_{out} = q_{in} + \frac{\partial q}{\partial x}\Delta x \quad (3)$$

Then substituting this back into Eqn.2 we have;

$$q_{in} + r\Delta x - \left(q_{in} + \frac{\partial q}{\partial x}\Delta x \right) - i\Delta x = \frac{\partial y}{\partial t}\Delta x \quad (4)$$

Expanding the brackets, the q terms cancel out to give;

$$r\Delta x - \frac{\partial q}{\partial x}\Delta x - i\Delta x = \frac{\partial y}{\partial t}\Delta x \quad (5)$$

Then dividing through by Δx and rearranging, we are left with the continuity equation;

$$\frac{\partial q}{\partial x} + \frac{\partial y}{\partial t} = r - i \quad (6)$$

A2.1.3 Momentum Equation

The principle of conservation of momentum implies that a moving body will neither gain nor lose momentum unless an external force is applied. This is classically known as Newton's second law of motion where the sum of the external forces on the body equals the rate of change of momentum;

$$Sum\ of\ \ external\ \ forces = rate\ of\ change\ of\ momentum \quad (7)$$

$$\sum F = \frac{dM}{dt} \quad (8)$$

Equating the left hand side of this equation, we have four external forces acting on the control volume. If forces acting in the downstream direction are positive, we have;

$$\sum F = F_{P_{left}} - F_{P_{right}} + F_g + F_f \quad (9)$$

where F_{Pleft} and $F_{\Pr ight}$ are the pressure forces acting on the left and right hand sides of the control volume respectively, F_g is the downstream component of the gravitational force, and F_f is the frictional resistance force. We can relate $F_{\Pr ight}$ in terms of F_{Pleft} using;

$$F_{P_{right}} = \left(F_{P_{left}} + \frac{\partial F_P}{\partial x} \Delta x \right) \quad (10)$$

where $\partial F_P / \partial x$ is the rate of change of pressure force across the thickness of the control volume. Substituting this into Eq. 9 we have;

$$\sum F = F_{P_{left}} - \left(F_{P_{left}} + \frac{\partial F_P}{\partial x} \Delta x \right) + F_g + F_f \quad (11)$$

The resultant pressure force acting on any section of the control volume (perpendicular to the channel bed) can be expressed using the relationship $F = (pgyA)/2$. In this case where we are dealing with unit width of channel, the cross-sectional area *A* is equal to the depth *y* giving;

$$F_p = \frac{1}{2} \rho g y^2 \quad (12)$$

The gravitational force acting in the downstream direction is given by;

$$F_g = \rho g y \Delta x S_o \quad (13)$$

where S_o is the channel slope and is in fact an approximation as theoretically the tangent of the channel angle should be used. This is called the small slope approximation (Stephenson and Meadows, 1986).

The friction force which retards the flow can be expressed in terms of a shear stress and the wetted area on which it is acting.

$$F_f = \tau P \Delta x \quad (14)$$

where τ is the shear stress and P is the wetted perimeter. By equating the energy loss by the work done by the shear force, the following relationship is established;

$$\tau = \rho g R S_f \quad (15)$$

where S_f is the slope of the energy line or friction slope and R is the hydraulic radius. Then substituting Eqn.15 into Eqn.14 and recalling that $R = A/P$, and for a unit width of channel $A = y$ we have;

$$F_f = \rho g y S_f \Delta x \quad (16)$$

Substituting Eqns.12, 13 and 16 into Eqn.11

$$\sum F = F_{P_{left}} - \left(F_{P_{left}} + \frac{\partial}{\partial x} \left(\frac{1}{2} \rho g y^2 \right) \Delta x \right) + \rho g y \Delta x S_o - \rho g y \Delta x S_f \quad (17)$$

Expanding the brackets, the F_{Pleft} terms cancel out. After differentiating *y* with respect to *x*, we have;

$$\sum F = -\frac{2}{2}\rho g y \frac{\partial y}{\partial x}\Delta x + \rho g y \Delta x S_o - \rho g y \Delta x S_f \quad (18)$$

Taking $\rho g y \Delta x$ out as a common factor, we can simplify this to obtain;

$$\sum F = \rho g y \Delta x \left(-\frac{\partial y}{\partial x} + S_o - S_f \right) \quad (19)$$

Now considering the RHS of Eqn.8, the change in momentum consist of two parts, a temporal momentum change and a spatial momentum change.

$$\frac{dM}{dt} = \frac{\partial M(x)}{\partial t} + \frac{\partial M(t)}{\partial x} \quad (20)$$

The momentum of the fluid is $M = \rho y \Delta x v$ and the temporal momentum change is its time derivative;

$$\frac{\partial M(x)}{\partial t} = \frac{\partial}{\partial t}\left(\rho y \Delta x v\right) \quad (21)$$

$$\frac{\partial M(x)}{\partial t} = \rho \Delta x \left(y \frac{\partial v}{\partial t} + v \frac{\partial y}{\partial t} \right) \quad (22)$$

The spatial momentum change is the space derivative of the momentum flux through the control surface $\rho y \Delta x v^2$;

$$\frac{\partial M(t)}{\partial x} = \frac{\partial}{\partial x}\left(\rho y \Delta x v^2\right) \quad (23)$$

$$\frac{\partial M(t)}{\partial x} = \rho \Delta x v \left(v \frac{\partial y}{\partial x} + 2y \frac{\partial v}{\partial x} \right) \quad (24)$$

Then substituting Eqns.22 and 24 into Eqn.20;

$$\frac{dM}{dt} = \rho \Delta x \left(y \frac{\partial v}{\partial t} + v \frac{\partial y}{\partial t} \right) + \rho \Delta x v \left(v \frac{\partial y}{\partial x} + 2y \frac{\partial v}{\partial x} \right) \quad (25)$$

To simplify this equation, we introduce another relationship for a channel of unit width;

$$q = vy \quad (26)$$

Differentiating with respect to *x* we obtain;

$$\frac{\partial q}{\partial x} = v \frac{\partial y}{\partial x} + y \frac{\partial v}{\partial x} \quad (27)$$

Then substituting this into our continuity equation (Eqn.6), we have;

$$v \frac{\partial y}{\partial x} + y \frac{\partial v}{\partial x} = r - i - \frac{\partial y}{\partial t} \quad (28)$$

Adding a $y\partial v/\partial x$ term to both sides of this equation, the LHS now resembles the second bracketed term of Eqn25;

$$v\frac{\partial y}{\partial x}+2y\frac{\partial v}{\partial x}=r-i-\frac{\partial y}{\partial t}+y\frac{\partial v}{\partial x} \quad \text{(29)}$$

Therefore, substituting Eqn.29 into Eqn.25 we have;

$$\frac{dM}{dt}=\rho\Delta x\left(y\frac{\partial v}{\partial t}+v\frac{\partial y}{\partial t}\right)+\rho\Delta xv\left(r-i-\frac{\partial y}{\partial t}+y\frac{\partial v}{\partial x}\right) \quad \text{(30)}$$

This equation can be simplified further as the $\rho\Delta xv\partial y/\partial t$ terms cancel out to give;

$$\frac{dM}{dt}=\rho\Delta xy\frac{\partial v}{\partial t}+\rho\Delta xv(r-i)+\rho\Delta xvy\frac{\partial v}{\partial x} \quad \text{(31)}$$

Taking $\rho\Delta xy$ out as a common factor, we simplify further to obtain;

$$\frac{\partial M}{\partial t}=\rho\Delta xy\left(\frac{\partial v}{\partial t}+\frac{v(r-i)}{y}+v\frac{\partial v}{\partial x}\right) \quad \text{(32)}$$

Then combining Eqns.19 and 32;

$$\rho\Delta xy\left(\frac{\partial v}{\partial t}+\frac{v(r-i)}{y}+v\frac{\partial v}{\partial x}\right)=\rho gy\Delta x\left(\frac{\partial y}{\partial x}+S_o-S_f\right) \quad \text{(33)}$$

Finally, by expanding the brackets, we obtain the second of the Saint Venant equations;

$$\frac{\partial v}{\partial t}+v\frac{\partial v}{\partial x}+g\left(\frac{\partial y}{\partial x}\right)=g\left(S_o-S_f\right)+\frac{v(i-r)}{y} \quad \text{(34)}$$

Appendix 2.2 Case study – evaluation of *SIRMOD*

During the early stages of this research, *SIRMOD* was seen as the industry standard software package for simulating furrow and border irrigation in Australia. It was therefore decided to undertake a preliminary study to evaluate the effectiveness of *SIRMOD* as a platform for decision support development.

A2.2.1 Outline of case study

Volume-balance, surface advance and recession data were collected from 3 different sugarcane farms in both the Burdekin Delta and Burdekin River Irrigation Areas. Over thirty individual irrigations were monitored with the results used to validate the performance of the surface irrigation model *SIRMOD* (Version 2.12).

Where furrow dimensions, surface flow characteristics and the modified Kostiakov intake parameters were measured for individual irrigations and applied in the model, results for the predicted advance and infiltration were generally found to range within 22.2% and 16.9% of the measured parameters, respectively. However, the usefulness of a model lies in its ability to predict irrigation performance where this level of input data is unavailable.

This study presents the results of a sensitivity analysis conducted to identify the relative effects on the model output of variations in the values of the input parameters. The use of infiltration parameters derived solely from the irrigation advance data and inflow is investigated along with the temporal variations in the measured infiltration functions.

A2.2.2 Rational for study

One of the perceived disadvantages of *SIRMOD* is its requirement for a substantial amount of measured data. This includes information on furrow profile shape and roughness, field size and slope, soil infiltration characteristics, irrigation inflow rates and event time. In practice this level of data tends to be difficult, time consuming and expensive to obtain. The question arises as to what effort is required or justified in data collection for the tool to be beneficial to the user. Therefore the case study was conducted to determine: (i) the ability of the model to simulate the physical event if all of the input data is available (model validation) and (ii) the usefulness of the model if some of the data is inaccurate or missing. The latter is undertaken in the form of a sensitivity analysis.

A2.2.3 Materials and methods

The individual performance of over thirty irrigations were monitored on three commercial furrow irrigated sugarcane farms in the Burdekin region during 1994-95 (Raine, 1995a). Sites (Table A2.2.1) were selected to be representative of the soils, irrigation design and management practices of the region. In each case, irrigation water was applied at low pressures from a head box through collapsible fluming and cut-off fluming cups at the end of each furrow. At each

site, up to twenty furrows were irrigated with measurements undertaken on up to three individual replicated furrows. Field slope and furrow length were measured manually while electronic water meters (Great Lakes Instruments, Wisconsin) were used to measure inflow and outflow rates (and hence, the total volumes) of water applied and lost as tailwater. The inflow meter was mounted in a length of 250 mm PVC pipe and installed in the layflat irrigation fluming between the headbox and the monitoring site. The outlet meters were mounted in 50 mm PVC tubing and sited within individual replicated furrows. The final intake rate for the furrow was determined as the difference between the inflow and outflow rates after the outflow rate had reached an effective maximum. Float sensors were sited at either 100 or 200 m intervals along the furrow length to measure water advance time, depth of flow and recession time.

To reduce errors due to non-uniformities in furrow shape, furrow geometries were measured using a profile meter at up to six locations within the trial site. This data was averaged and the program PCSv1.42 (Raine, 1995b) used to produce the empirically fitted profile parameters $\rho 1$, $\rho 2$, $\sigma 1$, $\sigma 2$, $\gamma 1$, and $\gamma 2$ which were used as input on the *SIRMOD* "Page 3" screen to describe the surface water storage. These values were also used in the calculation of the Manning n, as described on page 118 of Walker and Skogerboe (1987).

Table A2.2.1: Site information for irrigation trials.

Site	Surface texture	Subsurface texture	Average Slope	Field Length (m)	Inflow Range (l/s)
Home Hill	sandy clay loam	sandy clay loam	0.001825	400 m	0.99 to 3.50
Rita Island	sandy loam	sandy clay loam	0.001864	350 m	1.32 to 2.62
Jarvisfield	sandy clay loam	sandy clay	0.000914	450 m	2.04 to 3.31

The modified Kostiakov-Lewis infiltration equation (Equation 2.7) was used to described the infiltration process. The average hydraulic area of the furrow was calculated using special software and used to calculate the infiltration parameters a and k using a modified version of the two point method of Elliott & Walker (1982). If, as sometimes did occur, the value of the parameter a was calculated as a negative number using the two-point method, it was set equal to zero and k was recalculated. This implies a linear infiltration curve predominately influenced by a high final infiltration rate. However, it most likely means that the final infiltration rate (fo) was over determined.

A sensitivity analysis was undertaken using data from nine irrigation events, three tests from each site (Table A2.2.2). It was conducted by changing the measured input parameters individually by amounts larger than could be realistically encountered in practice. These were then used as input into *SIRMOD* and the output volume-balance recorded. In each case all other parameters were retained at their measured value.

Table A2.2.2: Irrigations tested in sensitivity analysis.

Sites	Home Hill	Rita Island	Jarvisfield
Test names	lv1120, lu56, lu139	j37, j74, j104	sh74, sh84, sh88

The full hydrodynamic model option was used preferentially when undertaking the simulations in *SIRMOD*. However, where this model was unable to complete a simulation due to either programming errors or mathematical non-viability, the zero inertia model and kinematic wave model options were used.

The program *INFILT* v3.01 (McClymont, 1995, McClymont and Smith, 1996) was used to generate alternative values of the three infiltration parameters and the cross-sectional area of flow from the measured advance data. The program does this by determining all four parameters in a single optimisation process. The empirical profile parameters were matched to the area parameter value generated from *INFILT* v3.01 using the profile program PCSv1.42. These results were then used as input into *SIRMOD* and the output compared with field measurements to determine the necessity of directly measuring the infiltration parameters. The effect of variation in the infiltration function of successive irrigations on the cumulative infiltration was also investigated.

A2.2.4 Validation of the model

To test the model's validity, the results for predicted advance, and runoff and infiltration volumes, were compared against the measured quantities for over 70 sets of individual furrow irrigation data (Table A2.2.3).

Table 2.2.3: Summary of validation results.

Output	slope of trendline	average error	r^2
Predicted versus measured advance (Figure A2.2.1)	0.91	22.2%	0.83
Predicted versus measured runoff volumes (Figure A2.2.2(a))	1.2	151%	0.2
Predicted versus measured infiltration volumes (Figure A2.2.2(b))	0.8	16.9%	0.63

Figure A2.2.1 shows the relationship between the measured advance times and those predicted by *SIRMOD*. The regression analysis shows a high correlation (r^2=0.83) while the slope (0.91) of the trendline indicates that *SIRMOD* is slightly underpredicting the advance times. The average error between the predicted and measured advance times was 22.2%.

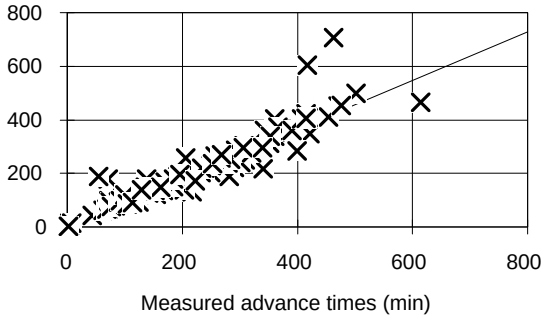


Figure A2.2.1: Comparison of measured and predicted advance times.

A significant correlation (r^2=0.63) was found between the measured and predicted infiltrated volumes with a regression coefficient of 0.8 (Figure A2.2.2b). This suggests that the model generally underpredicts the total infiltration during the irrigation. However, the average deviation was 16.9% of the measured infiltrated volume (or less than 10% of the total volume applied). Because of the direct relationship between the infiltrated and runoff volumes, an underprediction in infiltration resulted in an overprediction in runoff (Figure A2.2.2a). In this case the slope of the regression between the measured and predicted runoff was 1.2 and the correlation coefficient 0.2. The poor correlation coefficient reflects the proportionally small runoff volumes obtained in these trials.

The average deviation from the measured runoff volumes appears high at 151% although again it is less than 10% of the total volume applied. This result is greatly influenced by several extreme results with errors over 2000%. Excluding these extremes, the deviations from the measured runoff volumes remains under 40% for the majority of tests. The 16.9% average deviation between measured and predicated infiltration volumes provides us with some reassurance that the model simulates the physical process with some reliability. However, this is for the case where all input data has been measured with a reasonable degree of accuracy. The sensitivity analysis undertaken in the next section indicates the response of the model to changes in input data.

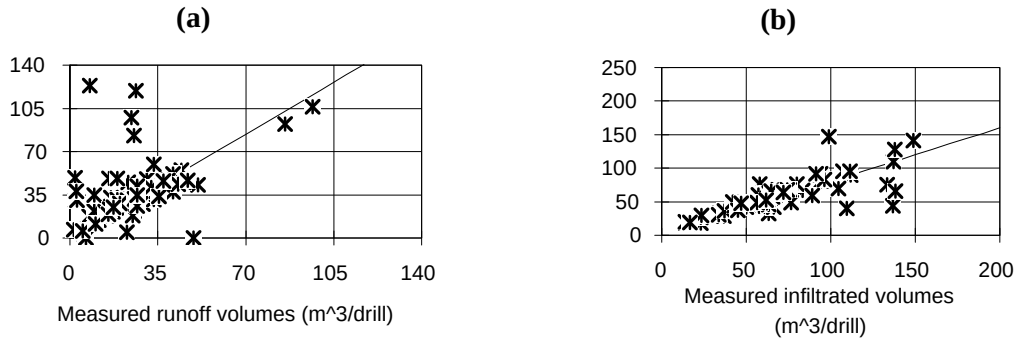


Figure A2.2.2: Comparison of measured and predicted (a) runoff volumes and (b) infiltrated volumes.

A2.2.5 Sensitivity analysis

Manning n

Figure A2.2.3(a) and (b) demonstrate the effect of changes in the Manning n parameter on the *SIRMOD* output where the Manning n was varied from 50% to 250% of the measured value. These graphs indicate that increasing the Manning n increases the simulated infiltrated volume. This was expected as increasing the Manning n effectively increases the roughness of the furrow, slowing the advance and allowing more time for infiltration. However, the error induced is a less than 4% deviation in the measured infiltrated volume for up to a 150% increase in the Manning n. The maximum deviation from the measured runoff volumes of 30% is a reflection of the proportionally small actual runoff volumes. The maximum volume-balance error was only 2.9% of the total volume applied.

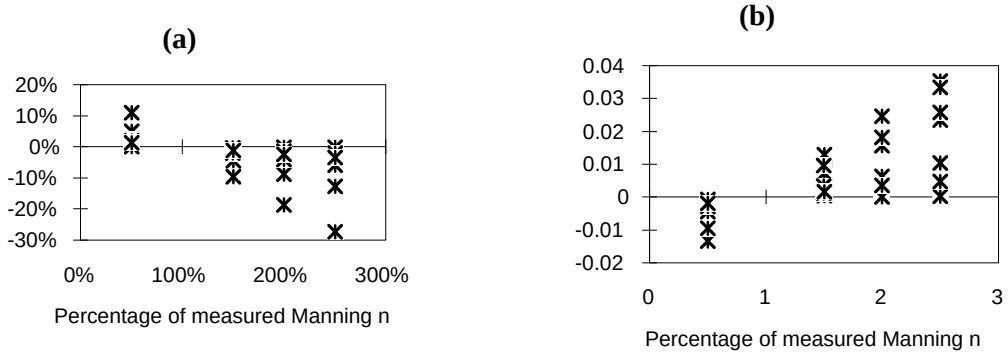


Figure A2.2.3: Effect of changes in Manning n on (a) runoff and (b) infiltrated volumes.

Field slope

The field slope parameter in *SIRMOD* was varied from 40% to 200% of the measured value. The results shown in Figure A2.2.4(a) and (b) indicate that by increasing the slope, we are effectively increasing the runoff and decreasing the infiltrated volume. Again, this was expected due to a faster advance.

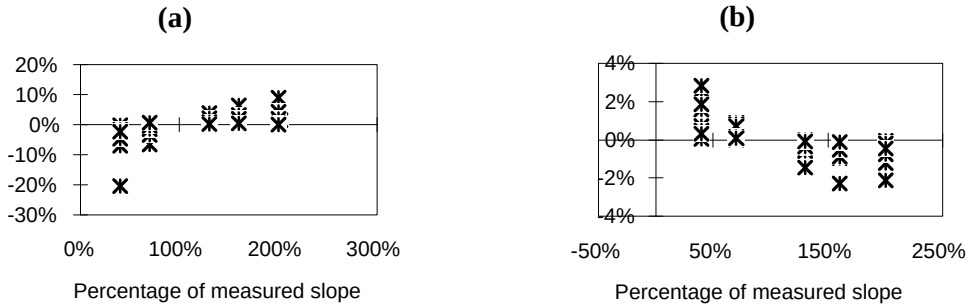


Figure A2.2.4: Effect of changes in field slope on (a) runoff and (b) infiltrated volumes.

The maximum errors induced by a reduction to 40% of the measured slope was a less than 3% deviation in infiltrated volume and 20% in the runoff volume. This

equates to 2.2% of the total inflow. Decreasing the slope had more effect on the simulated output quantities than an increase in slope.

Inflow

Inflow values ranging from 70% to 160% of the measured values were used as input to *SIRMOD*. The infiltration parameters and time to cutoff were entered as their measured values. This represents a situation which could occur if the infiltration parameters were calculated from a previous event, or using a point source method.

Figure A2.2.5(a) demonstrates the effect of inflow on the measured runoff volumes. The deviation from the measured runoff is up to 600% for a 160% change in inflow (less than 60% of the total volume applied). The large deviations in simulated runoff volumes can be explained by the fact that changing the inflow rate has little effect on infiltration of water through the soil, which is dominated by the soil properties. This is best visualized at large irrigation times where the final infiltration rate will be dominating the infiltration process.

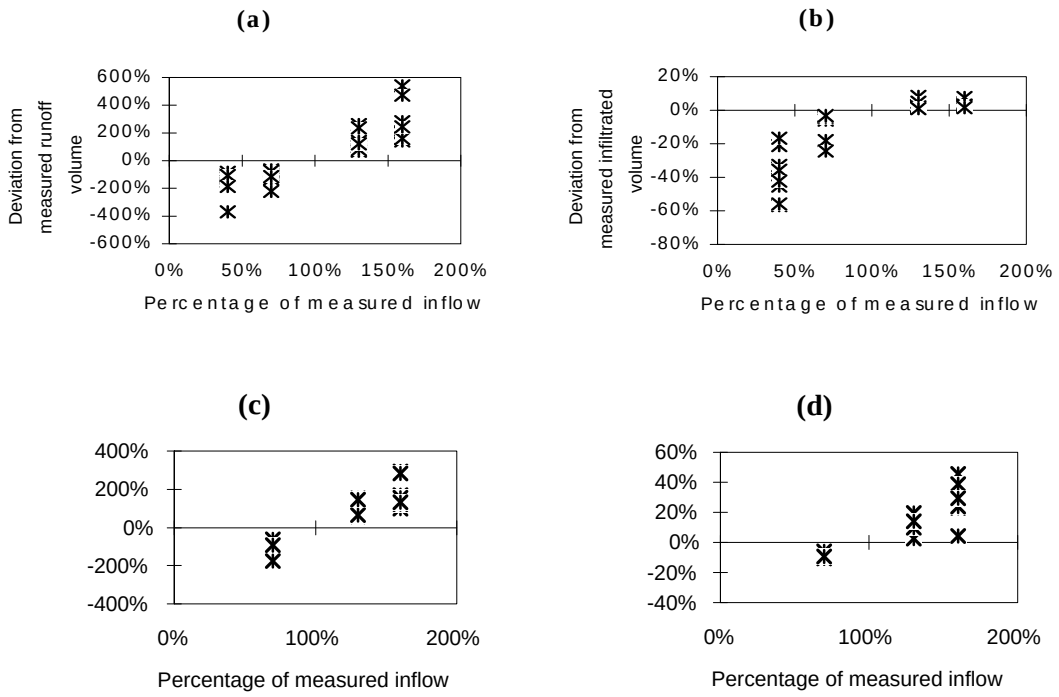


Figure A2.2.5: Effect of changes in inflow on; (a) runoff volume with constant infiltration parameters; (b) infiltrated volume with constant infiltration parameters; (c) runoff volume with recalculated infiltration parameters; and (d) infiltrated volume with recalcul

The effect on the infiltrated volumes of varying inflow (Figure A2.2.5(b)) is less dramatic with deviations ranging up to 60% of the measured infiltration volume. The large errors occurring at low inflow values arise when the simulated advance does not reach the end of the field and all of the water applied infiltrates with no runoff. For the cases tested, this usually meant that the simulated volume of water applied was less than the measured infiltrated volume. In practice, it

would be unlikely that the irrigator would apply an inflow rate below that required to reach the end of the field. Where runoff did occur, the largest infiltration error was only 7% of the total water applied.

When the Kostiakov-Lewis infiltration parameters a, and k are recalculated for the changes in inflow, the magnitude of the error between the measured and predicted infiltration volumes increases dramatically (Figure A2.2.5(d)). The maximum deviation between infiltrated volumes is now nearly 50% which accounts for 39% of the total water applied. The error between runoff volumes is in fact reduced (Figure A2.2.5(c)) to nearly 300% (44% of the total water applied) for the same inflow changes, though this is of little consequence as it is only compensating for the larger infiltration errors.

It should be noted that it was not possible to recalculate the infiltration parameters for the 40% inflow rate. This is not unreasonable as in practice it would be unlikely to find such a high advance rate from a low inflow rate.

The general trend is that an increase in inflow, without changing the cutoff time, will increase both predicted runoff and infiltrated volumes, whether or not the infiltration parameters are altered. However, it is the runoff volume that is most greatly affected by an increase in inflow.

Cross-sectional area of flow

Figure A2.2.6(a) and (b) demonstrate the effect of changes in the cross-sectional area of flow parameter on the output volumes. Again, the infiltration parameters were not recalculated for the new area. Both runoff and infiltration volumes remained relatively unaffected by changes of 30% to 250% of the measured area. The greatest infiltration error was only 1% while for runoff it was just under 3.5%. This corresponds to a maximum error of 0.62% of the total inflow volume. The scatter in the graphs is indicative of numerical rounding errors.

When the infiltration parameters were recalculated to be consistent with the wetted perimeter corresponding to the change in area, we once again see an increase in the output volume deviations. Figure A2.2.6 (c) demonstrates a maximum runoff error of 60% for a 70% reduction in cross-sectional area. Figure A2.2.6 (d) shows a maximum infiltration error of 16% for a 120% increase in the measured area parameter. There is an increase in the total volume-balance error to 9.0% of the inflow volume. The general trend of these graphs indicate that an increase in cross-sectional area of flow will increase the runoff, reducing the amount of water infiltrated. As the results of Figure A2.2.6 (a) and (b) indicate little change in the associated output volumes, Figure A2.2.6 (c) and (d) are in effect demonstrating the result of changes in the cross-sectional area parameters on the Kostiakov-Lewis infiltration parameters.

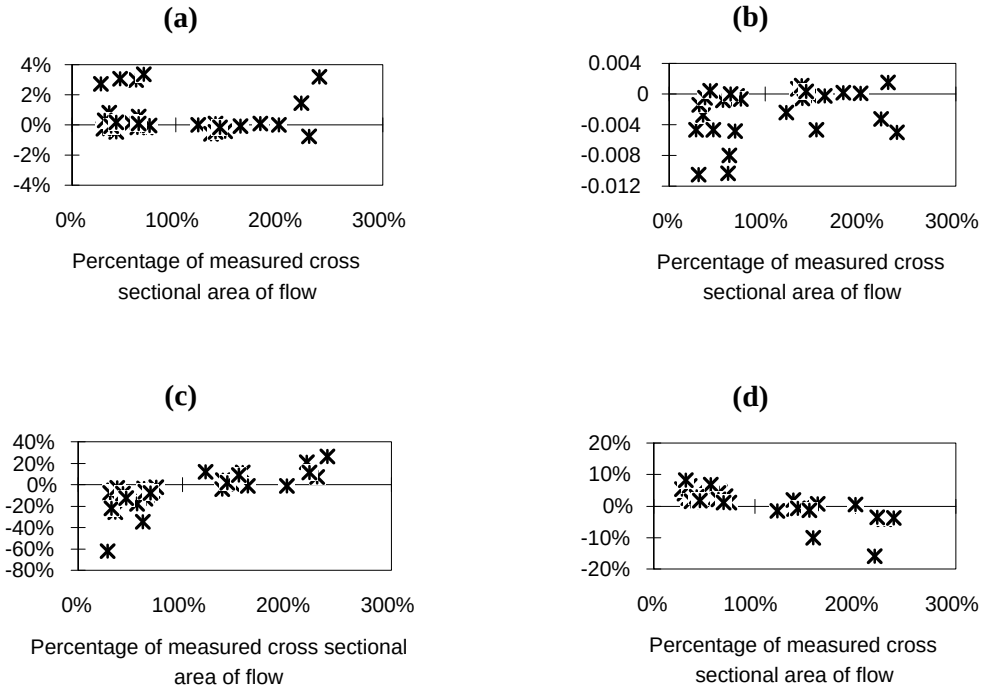


Figure A2.2.6: Effect of changes in cross-sectional area of flow on; (a) runoff volume with constant infiltration parameters; (b) infiltrated volume with constant infiltration parameters; (c) runoff volume with recalculated infiltration parameters; and (d) infiltrated volume with recalculated infiltration parameters.

Final infiltration rate

The final infiltration rate parameter (*fo*) was varied by ±100% of the measured value (Figures A2.2.7(a) and (b)). This involved the recalculation of a and k, although this was not always possible for very high final infiltration rates. The first of these figures shows a maximum deviation from the measured runoff volume of 360% at a zero final infiltration rate. Figure A2.2.7(b) similarly shows that the maximum error occurs at the zero infiltration rate, with a deviation of nearly 50% from the measured infiltration volume. This corresponds to a maximum error of 40% when expressed relative to the total inflow volume. Both figures demonstrate that an increase in the final infiltration rate leads to a reduction in runoff and hence, more water infiltrated into the soil.

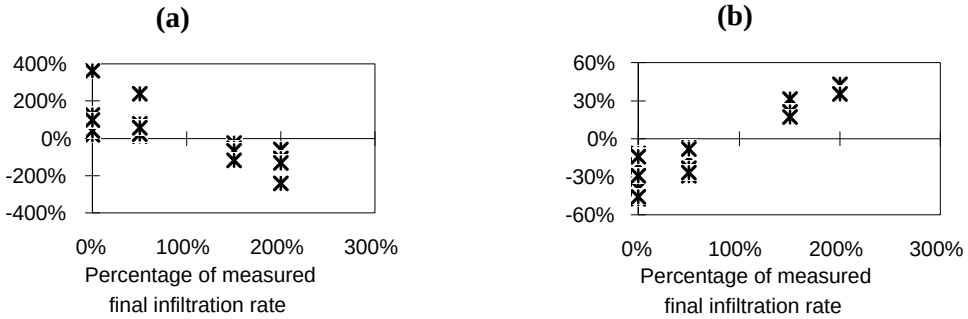


Figure A2.2.7: Effect of changes in final infiltration rate on (a) runoff and (b) infiltrated volumes.

A2.2.5 Mathematical convergence errors

The volume-balance error predicted by *SIRMOD* is a measure of the success of the mathematical convergence in the model. It was found that for most parameter combinations tested, the mathematical convergence error returned by *SIRMOD* was less than one percent. That is, the solution converged and the predicted infiltrated and runoff volumes added up to equal the volume of water applied to the field. The exception to this was when the predicted advance did not reach the end of the field. In this case, the error was still usually less than 20%.

A2.2.6 Results using empirically fitted infiltration parameters

Figures A2.2.8(a) and (b) show the magnitude of the volume errors incurred through using the output of *INFILT* v3.01 as input into *SIRMOD*. Five of the tests showed good agreement with the measured infiltration volumes with deviations of less than 12%. However, the three tests at the Jarvisfield site (j37, j74, j104) and the first test undertaken at the Rita Island site (sh74) showed a poor correlation between measured and predicted infiltration volumes. The maximum volume-balance error was for the event "j104" at 33.8% of the measured inflow volume.

The empirically fitted infiltration parameter based model under-predicted infiltration in all cases. However, Figure A2.2.2(b) demonstrated that *SIRMOD* generally under-predicts infiltration, so this result may not wholly be attributed to the empirically fitted infiltration parameters.

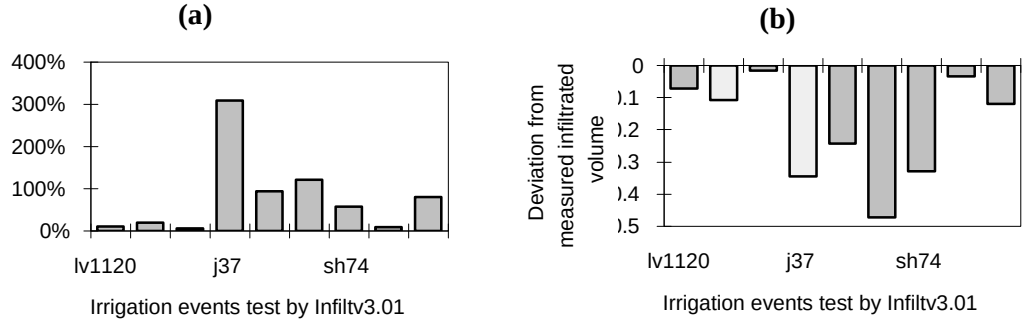


Figure A2.2.8: Results from *SIRMOD* using infiltration parameters from *INFILT* v3.01, showing effect on (a) runoff and (b) infiltrated volumes.

The optimisation undertaken in the *INFILT* v3.01 model is in effect a curve fitting exercise through the advance points (x,t). Therefore, you would expect the quality of the advance data to have an effect on the output parameter values. Table A2.2.4 shows the coefficients of variation for simple power curve regressions of the nine irrigations analysed. This data suggests that the irrigation data sets which produced poor correlations in the predicted volume-balance also

had high coefficients of variation and confirms that empirically fitted infiltration parameters are less reliable for "noisy" advance data.

Table A2.2.2: Coefficients of variation for power curve regressions of the nine irrigation events

Test	lv1120	lu56	lu139	j37	j74	j104	sh74	sh84	sh88
r^2	0.01	0.12	3.42	2.33	2.22	2.66	3.33	0.29	0.20

A2.2.7 Discussion of results of case study

The first part of this analysis showed that *SIRMOD* is able to simulate the surface irrigation process adequately when sufficient data are available. When used with measured data, *SIRMOD* showed a tendency to under-predict both the rate of advance and the volume infiltrated.

The underprediction by *SIRMOD* of the volumes infiltrated was also observed by Maheshwari and McMahon (1993b), a fact which they attributed to an uncertainty in the values they used for the infiltration parameters. Given that the same result occurred in the present study, this suggests that there could be a systematic error in *SIRMOD* which might be removed by an appropriate calibration procedure. Alternatively, this could be caused by the difference in structure between the two point method, and the simulation model leading to a "faulty" calibration.

Here calibration is defined as the process whereby the value of a parameter is adjusted until the predicted result matches the measured result. The infiltration parameters and the Manning n are the only data input to *SIRMOD* which are not measured directly and which therefore provide an opportunity for calibration.

In other applications of similar hydrodynamic models, for example in modelling river flows, the Manning n parameter is used as the calibration factor. One outcome of this is that it often results in unrealistically high values for *n*. A useful extension of the present study would be to explore the efficacy of attempting to calibrate *SIRMOD* against measured data prior to its use in optimisation of irrigation applications.

The *SIRMOD* program will be most useful if it can be used with confidence in a predictive role. The sensitivity analysis reported in this paper showed that for prediction to be successful, an accurate estimate of the infiltration characteristic of the soil is a necessity. It was the infiltration parameters which had far greatest effect on the model results. However, it is also the infiltration parameters which are the most difficult to measure or estimate. Point measurements are expensive and do not account well for the spatial and temporal variation in the soil properties, while techniques based on the irrigation advance cannot be applied before the first irrigation. Where possible, event based methods using data from the current or previous irrigation, should be employed. It is unlikely that an irrigator would measure the infiltration characteristic during each irrigation. This data would most likely come from some previous event. There is some slight risk in this practice as demonstrated by the temporal variation in infiltration at the Jarvisfield site. Using empirically fitted infiltration parameters may be a labour saving alternative to conventional event based methods although the process

should be undertaken with caution as poor advance data may influence the results. It was also shown that the model was relatively insensitive to changes or errors in the slope and the Manning n, while the cross-sectional profile of the furrow had little or no effect on the model results. Variation of the inflow had little influence on the volume infiltrated but had a significant effect on the runoff volume and hence the volumetric efficiency of an irrigation.

From the work reported in this paper it can be concluded that *SIRMOD* is a useful tool for surface irrigation design and management, provided an accurate description of the infiltration properties of the soil is available.

Appendix 3.1 Simulation engine source code

The following code listing represents the entire code base for the simulation engine, excluding that relating to the different “smart” parameter objects.

A3.1.1 C++ Header file

```
//---------------------------------------------------------------------------

#ifndef FIDOSimulationH
#define FIDOSimulationH

#include <Classes.hpp>
#include <vector>
#include <set>
#include "TeeGeometry.hpp"
#include "TeeGLCanvas.hpp"
#include "TeeOpenGL.hpp"
#include "SmartPointers.h"


//---------------------------------------------------------------------------
#ifndef SIMNAMES
    #define  MaxDepth   (*_MaxDepth)
    #define  TopWidth   (*_TopWidth)
    #define  MidWidth   (*_MidWidth)
    #define  BotWidth   (*_BotWidth)
    #define  sigma1     (*_sigma1)
    #define  sigma2     (*_sigma2)
    #define  rho1       (*_rho1)
    #define  rho2       (*_rho2)
    #define  ManN       (*_Manning_n)
    #define  KosA       (*_Kostiakov_a)
    #define  KosK       (*_Kostiakov_k)
    #define  KosFo      (*_Kostiakov_fo)
    #define  TotalTime  (*_TotalTime)
    #define  StageArray (*_StageArray)
    #define  DownstreamCell    (*_DownstreamCell)
    #define  UpstreamCell  (*_UpstreamCell)
    #define  Qin        (*_Qin)
    #define  So         (*_So)
    #define  dt         (*_dt)
    #define  Dist       (*_X)
    #define  Xinit      (*_Xinit)
    #define  A          (*_A)
    #define  Q          (*_Q)
    #define  Z          (*_Z)
    #define  D          (*_D)
    #define  P          (*_P)
    #define  WP         (*_WP)
    #define  VP         (*_VP)
    #define  dZdT       (*_dZdT)

    #define  DQ         _Q->FDeltaValues
    #define  DA         _A->FDeltaValues
    #define  DX         _X->FDeltaValues
    #define  DT         _dt->FDeltaValue

    #define  SOl        _So->L
    #define  Al         _A->L
    #define  Ql         _Q->L
    #define  Zl         _Z->L
    #define  Dl         _D->L
    #define  Pl         _P->L
    #define  WPl        _WP->L
    #define  ManNl      _Manning_n->L
    #define  rho1l      _rho1->L
    #define  rho2l      _rho2->L
    #define  sigma1l    _sigma1->L
    #define  sigma2l    _sigma2->L
    #define  VPl        _VP->L
    #define  dZdTl      _dZdT->L

    #define  SOr        _So->R
    #define  Ar         _A->R
    #define  Qr         _Q->R
    #define  Zr         _Z->R
    #define  Dr         _D->R
    #define  Pr         _P->R
    #define  WPr        _WP->R
    #define  ManNr      _Manning_n->R
    #define  rho1r      _rho1->R
    #define  rho2r      _rho2->R
    #define  sigma1r    _sigma1->R
    #define  sigma2r    _sigma2->R
    #define  VPr        _VP->R
    #define  dZdTr      _dZdT->R

    #define  SOj        _So->J
    #define  Aj         _A->J
    #define  Qj         _Q->J
    #define  Zj         _Z->J
    #define  Dj         _D->J
    #define  Pj         _P->J
```

```
    #define  WPj        _WP->J
    #define  VPj        _VP->J
    #define  dZdTj      _dZdT->J

    #define  SOm        _So->M
    #define  Am         _A->M
    #define  Qm         _Q->M
    #define  Zm         _Z->M
    #define  Dm         _D->M
    #define  Pm         _P->M
    #define  WPm        _WP->M
    #define  VPm        _VP->M
    #define  dZdTm      _dZdT->M
#endif

enum TPhaseElements {peAdvance,peRecession,peInflow,peRunoff,pePonding,peLateralFlow};
typedef Set<TPhaseElements, peAdvance,peLateralFlow>  TPhaseComponents;

typedef void __fastcall (__closure *TCalculateCellPositions)(int xpos);
typedef double __fastcall (__closure *TDerivativeCellFunction)(void);
typedef void __fastcall (__closure *TCalcAuxCoefficients)(void);
typedef void __fastcall (__closure *TUpdateParamEstimates)(void);
typedef void __fastcall (__closure *TResetSimulation)(void);



enum TInitialSolutionDirection {sdTopToBottom,sdBottomToTop} ;
enum TGridType {gtEulerian,gtLangrangian};

class TFIDOSimulation;
class TFIDOModelDataTreeObject;
class T1DInputParameter;
class T1DGridParameter;
class T2DGridParameter;
class TCustomGridParameter;
class ECustomFIDOException;
class TFIDOOutputTreeObject;
class TSimulationParametersObject;

SmartPointer<TFIDOSimulation> __fastcall CreateFIDOSimulation(void);

class TFIDOSimulation
{
public:
    __fastcall TFIDOSimulation(void);
    __fastcall ~TFIDOSimulation(void){};
    bool __fastcall RunSimulation(void);
    void __fastcall ResetSimulation(void);
    int x,t;
    bool stop;
    bool CutoffTimeExceeded;
    int lastcell,firstcell;
    double FieldLength;
    double ZRequired;
    double TimeToCutoff;
    double DefaultTimeStep;
    double SimulationTimeStep;
    double MaxZ;
    double An;
    int TotalIterations;
    int ReducedSteps;
    bool StopWhenRunoffOccurrs;
    SmartPointer<T1DInputParameter> _Qin;
    SmartPointer<T1DInputParameter> _So;
    SmartPointer<T1DInputParameter> _MaxDepth;
    SmartPointer<T1DInputParameter> _TopWidth;
    SmartPointer<T1DInputParameter> _MidWidth;
    SmartPointer<T1DInputParameter> _BotWidth;
    SmartPointer<T1DInputParameter> _sigma1;
    SmartPointer<T1DInputParameter> _sigma2;
    SmartPointer<T1DInputParameter> _rho1;
    SmartPointer<T1DInputParameter> _rho2;
    SmartPointer<T1DInputParameter> _Manning_n;
    SmartPointer<T1DInputParameter> _Kostiakov_a;
    SmartPointer<T1DInputParameter> _Kostiakov_k;
    SmartPointer<T1DInputParameter> _Kostiakov_fo;
    T1DGridParameter*_dt;

    T1DGridParameter*_TotalTime;
          T1DGridParameter*_DownstreamCell;
          T1DGridParameter*_UpstreamCell;
    T2DGridParameter*_X;
    T2DGridParameter*_A;
    T2DGridParameter*_Q;
    T2DGridParameter*_Z;

    SmartPointer<T2DGridParameter> _D;
    SmartPointer<T2DGridParameter> _P;
    SmartPointer<T2DGridParameter> _WP;
    SmartPointer<T2DGridParameter> _VP;
    SmartPointer<T2DGridParameter> _dZdT;
    SmartPointer<TList> GridParametersList;
    bool StopOnException;
    __property TResetSimulation OnResetSimulation = { read = FOnResetSimulation, write = FOnResetSimulation };
    __property  TCalculateCellPositions  CalculateCellPositions  =  {  read  =  FCalculateCellPositions,  write  =
FCalculateCellPositions };
    __property TGridType GridType={read=GetGridType,write=SetGridType};
    __property TSimulationParametersObject* CurrentSimData = { read = GetCurrentSimData, write = SetCurrentSimData
};
    __property TFIDOOutputTreeObject* OutputObject = { read = GetOutputObject, write = SetOutputObject };
    __property std::vector<int> Iterations={read=FIterations};
```

```
    void __fastcall LoadSimData(TSimulationParametersObject*SimData);
    void __fastcall SetTimeStep(void);
    void __fastcall EnableConvergenceLogging(void);
    void __fastcall DisableConvergenceLogging(void);
    __property int StopAtPoint = { read = GetStopAtPoint, write = SetStopAtPoint };

protected:
    void __fastcall UpdateA(const int&i, const int&j);
    void __fastcall UpdateA_LastCell(const int&i, const int&j);
    void __fastcall UpdateA_Runoff(const int& i);
    void __fastcall UpdateT(const int&i);
    void __fastcall UpdateQ(const int&i);
    void __fastcall UpdateX(const int&i);
    void __fastcall CombineLastTwoCells(void);
    void __fastcall ResetGridParameterDeltaValues(void);
    bool __fastcall AreThereOscillationsAtRecessionFront(void);
    bool __fastcall AreThereOscillationsAtAdvanceFront(void);

    __property int CurrentCell={read=GetCurrentCell,write=SetCurrentCell};


    __property bool AllowRunoff={read=GetAllowRunoff,write=SetAllowRunoff};
    __property bool CutoffTimeReached={read=GetCutoffTimeReached};
    __property bool CellFlowsAreNegligible={read=GetCellFlowsAreNegligible};

          __property bool StillConverging = { read = GetStillConverging };
    __property bool StillSimulating = { read = GetStillSimulating };
    __property TCustomGridParameter*GridParameter[int index]={read=GetGridParameter};
    __property int CellCount ={read=GetCellCount};

    __property                                                                    TCalcAuxCoefficients
CalculateAuxCoefficients={read=FCalculateAuxCoefficients,write=FCalculateAuxCoefficients};
    __property                                                                    TUpdateParamEstimates
UpdateParameterEstimates={read=FUpdateParameterEstimates,write=FUpdateParameterEstimates};
     __property TDerivativeCellFunction dRC_dT = { read = FdRC_dT, write = FdRC_dT };
    __property TDerivativeCellFunction dRM_dT = { read = FdRM_dT, write = FdRM_dT };
    __property TDerivativeCellFunction dRC_dAr= { read = FdRC_dAr, write = FdRC_dAr};
    __property TDerivativeCellFunction dRM_dAr= { read = FdRM_dAr, write = FdRM_dAr};
    __property TDerivativeCellFunction dRC_dAr_LastCell= { read = FdRC_dAr_LastCell, write = FdRC_dAr_LastCell};
    __property TDerivativeCellFunction dRM_dAr_LastCell= { read = FdRM_dAr_LastCell, write = FdRM_dAr_LastCell };
    __property TDerivativeCellFunction dRC_dParam= { read = FdRC_dParam, write = FdRC_dParam};
    __property TDerivativeCellFunction dRM_dParam= { read = FdRM_dParam, write = FdRM_dParam};
    __property   TDerivativeCellFunction   dRC_dParam_LastCell=   {   read   =   FdRC_dParam_LastCell,   write   =
FdRC_dParam_LastCell };
    __property   TDerivativeCellFunction   dRM_dParam_LastCell=   {   read   =   FdRM_dParam_LastCell,   write   =
FdRM_dParam_LastCell};

    __property bool SolveForT = { write = SetSolveForT };
    __property bool SolveForX = { write = SetSolveForX };
    __property bool SolveForRunoff = { write = SetSolveForRunoff };
private:
    bool NewTimeStep;
    int SimulationRepeats;
    int furtherestdownstreamcellindex;
    double dummy; //delete this
    double __fastcall dRMd(double&param);
    double __fastcall dRCd(double&param);
    bool FirstSim; //delete this
    bool Convergence;
    bool FieldLengthExceeded;
    bool IterationsExceeded;
    bool FieldLengthReached;
    bool LogConvergence;
    TGridType FGridType;
    int MaxIterations;
    TPhaseComponents Components;
    double DampeningFactor;
    double RMBodyTemp;
    double theta,phi,inv_phi,inv_theta;
//    intRemovedCellCount;
    void __fastcall ResizeMemory(const int&xcount,const int&tcount);
    double T1,T2,T3,T4,T5,T6,denom;
    int maxxcount;
    double a,b,c,d,e,g,p,q,r,u,w;
    double _s;
    double Xrl,Xmj,Xrm,Xlj;
    std::vector<double>E;
    std::vector<double>H;
    std::vector<double>F;
    std::vector<double>U;
    std::vector<double>V;
    std::vector<double>Y;
    std::vector<double>W;
    int FCurrentCell;
    TInitialSolutionDirection InitialSolutionDirection;
    double AMax;
    double drmdt;
    double LowLimitWeighting;
    bool MakeLastCellDerivativeEqualZero;


    std::vector<int>FIterations;
    TResetSimulation FOnResetSimulation;
    TCalculateCellPositions FCalculateCellPositions;
    TCalcAuxCoefficients FCalculateAuxCoefficients;
    TUpdateParamEstimates FUpdateParameterEstimates ;

    TDerivativeCellFunction FdRC_dT;
    TDerivativeCellFunction FdRM_dT;
    TDerivativeCellFunction FdRC_dAr;
    TDerivativeCellFunction FdRM_dAr;
    TDerivativeCellFunction FdRC_dAr_LastCell;
```

```
    TDerivativeCellFunction FdRM_dAr_LastCell;
    TDerivativeCellFunction FdRC_dParam;
    TDerivativeCellFunction FdRM_dParam;
    TDerivativeCellFunction FdRC_dParam_LastCell;
    TDerivativeCellFunction FdRM_dParam_LastCell;
    TSimulationParametersObject* FCurrentSimData;
    TFIDOOutputTreeObject* FOutputObject;
    int FStopAtPoint;

    void __fastcall CreateParameters(void);
    SmartPointer<T1DGridParameter> __fastcall Create1DGridParameter(SmartPointer<TList> List,AnsiString name,bool
XOrientated);
    SmartPointer<T1DInputParameter>    __fastcall    Create1DInputParameter(SmartPointer<TList>    List,AnsiString
name,bool XOrientated);
    SmartPointer<T2DGridParameter> __fastcall Create2DGridParameter(SmartPointer<TList> List,AnsiString name);

    TGridType __fastcall GetGridType();
    void __fastcall SetGridType(TGridType type);

    void __fastcall SetCurrentCell(int cell);
    void __fastcall SetSolutionFunctionPointers(void);
    void __fastcall SetSolutionParameters(void);
    int __fastcall GetCurrentCell(void);
    bool __fastcall GetStillConverging();
    void __fastcall CalculateHydraulicParameters(int CellSide);
    void __fastcall CalculateDerivativeValues(int cell) ;
    void __fastcall DealWithTheProblem(ECustomFIDOException & Problem);
    void __fastcall CalculateEulerianCellPositions(int xpos);
    void __fastcall CalculateLangrangianCellPositions(int xpos);
    void __fastcall RemoveUnwantedUpstreamCells(void);
    void __fastcall RemoveUnwantedDownstreamCells(void);
    void __fastcall SetupMemoryForParameters(void);
    void __fastcall DetermineIrrigationStage(void);
    void __fastcall DetermineSolutionCellRange(void);
    TCustomGridParameter*GetGridParameter(int index);
    void __fastcall RemoveEmptyCells(void);
    void __fastcall CheckForAbnormalities(void) ;
    void __fastcall CalculateCellParameters(void);
    void __fastcall UndoTimeStep(void);
    void __fastcall UpdateIterationCount(void);
    void __fastcall ResetChecksAndTolerences(void);
    bool __fastcall GetStillSimulating();
    void __fastcall ResetIterationCount(void);
    void __fastcall CheckConvergence(void);
    void __fastcall CheckForBreakInSimulation(void);
    void __fastcall IncrementTimeStep(void);

    void __fastcall SolveEquationsForFirstCellForDistance(void);
    double __fastcall ResidualOfContinuity(void);
    double __fastcall ResidualOfMomentum(void);
    double __fastcall CalculateHydrostaticPressure(int xpos);
    void __fastcall SetDerivativeFunctionPointers(void);
    double __fastcall CalculateDragForce(int xpos);
    double __fastcall CalculateWettedPerimeter(int xpos);
    double __fastcall CalculateWettedPerimeterDependantInfiltration(int xpos);
    double __fastcall CalculateKostiakovLewisInfiltration(int xpos);
    double __fastcall CalculateVelocityPressureFactor(int xpos);
    void __fastcall CalculateInfiltrationForLateralSurfaceFlow(void);
    void __fastcall DetermineIrrigationComponents(void);
    double __fastcall ZeroFunction(void);
    void __fastcall RefineGrid(const int& t);
    SmartPointer<TCurveFit>__fastcall CreateCurveFit(void);

    double __fastcall RCTip(void);
    double __fastcall RMTip(void);
    void __fastcall SaveParameterEstimates(void);
    void __fastcall UndoIteration(void);
    int __fastcall GetPosAtGreatestDepth(void);
    double __fastcall dRC_dX(void);
    double __fastcall dRM_dX(void);
    double __fastcall dZ_dA(const unsigned& xcoord,const unsigned& tcoord);
    double __fastcall dZ_dT(const unsigned& xi);
    double __fastcall dRC_dAl(void);
    double __fastcall dRC_dQl(void);
    double __fastcall dRC_dAr_Runoff(void);
    double __fastcall dRC_dAr_Normal(void);
    double __fastcall dRM_dAr_Normal(void);
    double __fastcall dRM_dAr_Runoff(void);
    double __fastcall dRC_dt(void);
    double __fastcall dRM_dt(void);
    double __fastcall dRM_dAl(void);
    double __fastcall dRM_dQl(void);
    double __fastcall dRC_dQr(void);
    double __fastcall dRM_dQr(void);
    double __fastcall dRM_dQr_Runoff(void);


    double __fastcall ResidualOfMomentum_Runoff(void);
    double __fastcall ResidualOfContinuity_Runoff(void);

    void __fastcall SetAllowRunoff(bool);
    bool __fastcall GetAllowRunoff(void);
    void __fastcall SetInitialParameterEstimatesForFirstCell(void);
    void __fastcall SetInitialParameterEstimates(void);
    bool __fastcall GetCutoffTimeReached(void);
    bool __fastcall GetCellFlowsAreNegligible(void);
    void __fastcall CalculateAuxCoefficients1(void);
    void __fastcall CalculateAuxCoefficients2(void);
    void __fastcall UpdateParameterEstimates1(void);
    void __fastcall UpdateParameterEstimates2(void);
    void __fastcall UpdateParameterEstimatesForFirstCell(void);
    int __fastcall GetCellCount(void);
```

```
    void __fastcall SetSolveForT(bool value);
    void __fastcall SetSolveForX(bool value);
    void __fastcall SetSolveForRunoff(bool value);
    void __fastcall SetCurrentSimData(TSimulationParametersObject* value);
    TSimulationParametersObject* __fastcall GetCurrentSimData();
    void __fastcall SetOutputObject(TFIDOOutputTreeObject* value);
    TFIDOOutputTreeObject* __fastcall GetOutputObject();
    int __fastcall GetActiveModelCount();
    TSimulationParametersObject* __fastcall GetActiveModelSimData(int index);
    void __fastcall UpdateOutputObjectProperties(void);
    double __fastcall GetDampeningFactor(void);
    void __fastcall SetStopAtPoint(int value);
    int __fastcall GetStopAtPoint();

};


#endif
```

A3.1.2 C++ Source File

```
#include <vcl.h>
#include "PreCompiledHeaders.h"
#pragma hdrstop

#include "Simulation.h"
#include "FIDOModelTreeObject.h"
#include "GridParameters.h"
#include "SimulationExceptionHandling.h"
#include "FIDOOutputTreeObject.h"
#include "SimulationParametersObject.h"

#include <math.h>

SmartPointer<TFIDOSimulation> __fastcall CreateFIDOSimulation(void)
{
    SmartPointer<TFIDOSimulation> Sim(new TFIDOSimulation);
    #ifdef ASSIGN_SMART_PTR_NAMES
    Sim.PtrName="CreateFIDOSimulation.Sim";
    #endif
    return Sim;
}

__fastcall TFIDOSimulation::TFIDOSimulation(void)
{
    CreateParameters();
    MakeLastCellDerivativeEqualZero=false;
    FOnResetSimulation=0;
    StopOnException=true;
    MaxIterations=10;
    DefaultTimeStep=600;
    InitialSolutionDirection=sdTopToBottom;
    theta=0.6;
    phi=0.6;
    inv_phi=0.4;
    inv_theta=0.4;
    GridType=gtEulerian;
    LogConvergence=false;
}

void __fastcall TFIDOSimulation::LoadSimData(TSimulationParametersObject*SimData)
{
    CurrentSimData=SimData; //CurrentSimData is a property, and many initialisations occur in the setter.
    ResetSimulation();
}

bool __fastcall TFIDOSimulation::RunSimulation(void)
{
    SimulationRepeats=0;
    SimulationTimeStep=DefaultTimeStep;
    do
    {
        try
        {
            if(NewTimeStep)
            {
                IncrementTimeStep();
                SetupMemoryForParameters();
                ResetIterationCount();
                DetermineIrrigationComponents();
                SetSolutionParameters();
                DetermineSolutionCellRange();
                SetSolutionFunctionPointers();
                SetInitialParameterEstimates();
                ResetGridParameterDeltaValues();

            }
            if((t==1||CellCount>1)&&stop==false)
            {
                do
                {
                    try
                    {
                        SaveParameterEstimates();
                        UpdateIterationCount();
                        if(FCalculateAuxCoefficients)
                            CalculateAuxCoefficients();
```

```
                        UpdateParameterEstimates();
                        CheckConvergence();

                        //think about constraining delta values to half the actual value
                    }
                    catch(ECustomFIDOException & Problem)
                    {
                        DealWithTheProblem(Problem);
                        UpdateOutputObjectProperties();
                        return false;
                    }
                    catch(...)
                    {
                        MessageDlg("Unexpected Error!", mtError, TMsgDlgButtons() << mbOK, 0);
                        throw;
                    }
                }
                while(StillConverging);
                    CheckForAbnormalities();
                RemoveEmptyCells();
                if(StopAtPoint>0&&Dist(t,lastcell)>StopAtPoint)
                {
                    UpdateOutputObjectProperties();
                    return true;
                }

            }
            else
                stop=true;
        }
        catch(...)
        {
            stop=true;
            if(!LogConvergence)
            {
                --t;
                --t;
            }
        }
    }while(StillSimulating);

    UpdateOutputObjectProperties();
    if((CellCount<=1&&t>1)||(StopWhenRunoffOccurrs&&(Components.Contains(peRunoff)
          ||Components.Contains(peLateralFlow))))
        return true;
    return false;
}

void __fastcall TFIDOSimulation::CreateParameters(void)
{
    SmartPointer<TList> List (new TList);
    GridParametersList=List;
    #ifdef ASSIGN_SMART_PTR_NAMES
    GridParametersList.PtrName="TFIDOSimulation.GridParametersList";
    #endif
    _Qin=           Create1DInputParameter(List,"Inflow rate",false);
    _So=            Create1DInputParameter(List,"Slope",true);
    _MaxDepth=      Create1DInputParameter(List,"Max Furrow Depth",true);
    _TopWidth=      Create1DInputParameter(List,"Top Furrow Width",true);
    _MidWidth=      Create1DInputParameter(List,"Mid Furrow Width",true);
    _BotWidth=      Create1DInputParameter(List,"Bot Furrow Width",true);

    _sigma1=        Create1DInputParameter(List,"Sigma1",true);
    _sigma2=        Create1DInputParameter(List,"Sigma2",true);
    _rho1=          Create1DInputParameter(List,"Rho1",true);
    _rho2=          Create1DInputParameter(List,"Rho2",true);
    _Manning_n=     Create1DInputParameter(List,"Manning n",true);
    _Kostiakov_a=   Create1DInputParameter(List,"Kostiakov a",true);
    _Kostiakov_k=   Create1DInputParameter(List,"Kostiakov k",true);
    _Kostiakov_fo=  Create1DInputParameter(List,"Kostiakov fo",true);

    _D=             Create2DGridParameter(List,"Drag force");
    _P=             Create2DGridParameter(List,"Hydrostatic pressure");
    _WP=            Create2DGridParameter(List,"Wetted-perimeter");
    _VP=            Create2DGridParameter(List,"Veloctity pressure factor");
    _dZdT=          Create2DGridParameter(List,"Infiltration rate");
}

SmartPointer<T1DGridParameter>        __fastcall        TFIDOSimulation::Create1DGridParameter(SmartPointer<TList>
List,AnsiString name,bool XOrientated)
{
    SmartPointer<T1DGridParameter> TempParameter(new T1DGridParameter(this,name));
    TempParameter->XOrientated=XOrientated;
    #ifdef ASSIGN_SMART_PTR_NAMES
    TempParameter.PtrName="TFIDOSimulation::Create1DGridParameter.TempParameter - "+name;
    #endif
    List->Add(TempParameter.Get());
    return TempParameter;
}

SmartPointer<T1DInputParameter>       __fastcall       TFIDOSimulation::Create1DInputParameter(SmartPointer<TList>
List,AnsiString name,bool XOrientated)
{
    SmartPointer<T1DInputParameter> TempParameter(new T1DInputParameter(this,name));
    TempParameter->XOrientated=XOrientated;
    #ifdef ASSIGN_SMART_PTR_NAMES
    TempParameter.PtrName="TFIDOSimulation::Create1DInputParameter.TempParameter - "+name;
    #endif
    List->Add(TempParameter.Get());
    return TempParameter;
}
```

```
SmartPointer<T2DGridParameter>        __fastcall        TFIDOSimulation::Create2DGridParameter(SmartPointer<TList>
List,AnsiString name)
{
    SmartPointer<T2DGridParameter> TempParameter(new T2DGridParameter(this,name));
    #ifdef ASSIGN_SMART_PTR_NAMES
    TempParameter.PtrName="TFIDOSimulation::Create2DGridParameter.TempParameter - "+name;
    #endif
    List->Add(TempParameter.Get());
    return TempParameter;
}

void __fastcall TFIDOSimulation::DealWithTheProblem(ECustomFIDOException & Problem)
{
    Problem.HandleException();
   // CurrentSimData->SaveErrorReport(OutputObject->ErrorMessage);
}

TCustomGridParameter*TFIDOSimulation::GetGridParameter(int index)
{
    if(index>=0&&index<GridParametersList->Count)
        return (TCustomGridParameter*)GridParametersList->Items[index];
    return 0;
}

void __fastcall TFIDOSimulation::RemoveEmptyCells(void)
{
    if(t>0&&CutoffTimeExceeded&&Convergence)
    {
          RemoveUnwantedUpstreamCells();
          RemoveUnwantedDownstreamCells();
          if(CellCount<=1&&t>1)
             stop=true;
    }

}

bool __fastcall TFIDOSimulation::GetStillSimulating()
{
    return !stop;
}

void __fastcall TFIDOSimulation::SetCurrentCell(int cell)
{
    FCurrentCell=cell;
    for(int i=0;i<GridParametersList->Count;++i)
        GridParameter[i]->CellIndex=cell;

    OutputObject->_dt->CellIndex=cell;
    OutputObject->_A->CellIndex=cell;
    OutputObject->_Q->CellIndex=cell;
    OutputObject->_Z->CellIndex=cell;
    OutputObject->_X->CellIndex=cell;
    OutputObject->_DownstreamCell->CellIndex=cell;
          OutputObject->_UpstreamCell->CellIndex=cell;
    OutputObject->_TotalTime->CellIndex=cell;
}

int __fastcall TFIDOSimulation::GetCurrentCell(void)
{
    return FCurrentCell;
}


void __fastcall TFIDOSimulation::DetermineIrrigationComponents(void)
{
    if(Components.Empty())
        Components=Components<<peAdvance<<peInflow;
    else if(!Components.Contains(peLateralFlow))
    {
        if(Components.Contains(peAdvance))
        {
            if(FieldLengthReached)
            {
                if(AllowRunoff)
                    Components=Components<<peRunoff>>peAdvance;
                else
                    Components=Components<<pePonding>>peAdvance;
            }
        }
        if(CutoffTimeReached&&!Components.Contains(peRecession))
        {
            Components=Components>>peInflow<<peRecession;
            //stop=true;
        }
        if(Components.Contains(peRecession))
        {
            if(CellFlowsAreNegligible)
                Components=Components<<peLateralFlow>>peRecession>>peAdvance>>peRunoff>>pePonding;
        }
    }
    if(StopWhenRunoffOccurrs&&(Components.Contains(peRunoff)||Components.Contains(peLateralFlow)))
        stop=true;
}

bool __fastcall TFIDOSimulation::GetStillConverging()
{
          return !Convergence;
}

void __fastcall TFIDOSimulation::DetermineSolutionCellRange(void)
{
```

```
    UpstreamCell(t)=(t!=1?UpstreamCell(t-1):UpstreamCell(t-1)+1); //hopefully, empty cells should be chopped at
the end of the last itteration.
    DownstreamCell(t)=(Components.Contains(peAdvance)?DownstreamCell(t-1)+1:DownstreamCell(t-1));
    firstcell=UpstreamCell(t);
    lastcell=DownstreamCell(t);
    if(lastcell>furtherestdownstreamcellindex)
        furtherestdownstreamcellindex=lastcell;
}

void __fastcall TFIDOSimulation::UpdateIterationCount(void)
{
         ++Iterations[t];
    ++TotalIterations;
    _X->IncreaseDeltaValueSize();
    _dt->IncreaseDeltaValueSize();
    _A->IncreaseDeltaValueSize();
    _Q->IncreaseDeltaValueSize();
    _TotalTime->IncreaseDeltaValueSize();


    DampeningFactor=GetDampeningFactor();
    ResetChecksAndTolerences();
}

void __fastcall TFIDOSimulation::CheckForAbnormalities(void)
{
   if(t>0)
   {
        FieldLengthExceeded=(Dist(t,lastcell)>FieldLength);
        if(!FieldLengthReached)
        {
            if(Dist(t,lastcell)==FieldLength)
            {
                FieldLengthReached=true;
                FieldLengthExceeded=false;
                //double dx1=Dist(t,lastcell)-Dist(t,lastcell-1);
            }
        }
        if(!CutoffTimeExceeded&&TotalTime(t)>=TimeToCutoff)
        {
            CutoffTimeExceeded=true;
           // stop=true;
        }
      if(t>1000)
        stop=true;
    }
}


void __fastcall TFIDOSimulation::ResetChecksAndTolerences(void)
{
    _Q->ResetConvergenceParameters();
          _A->ResetConvergenceParameters();
}

void __fastcall TFIDOSimulation::CheckConvergence(void)
{

          if(!Components.Contains(peLateralFlow)&&(!_Q->Convergence||!_A->Convergence))
    {
                    Convergence=false;
        if(AreThereOscillationsAtRecessionFront())
        {

            if(firstcell<lastcell)
            {
                Q(t,firstcell-1)=0;
                            Z(t,firstcell-1)+= A(t,firstcell-1)/2.0;
                            A(t,firstcell-1)=0;
                ++firstcell;
                UpstreamCell(t)=firstcell;
              //  Q(t,firstcell-1)=0;
              //  A(t,firstcell-1)=0;

            }
        }
        if(Components.Contains(peRecession)&&Iterations[t]>20)
        {
            Components=Components.Clear();
            Components=Components<<peLateralFlow;
            if(!(StopWhenRunoffOccurrs&&(Components.Contains(peRunoff)||Components.Contains(peLateralFlow))))
            {
                SetSolutionParameters();
                DetermineSolutionCellRange();
                SetSolutionFunctionPointers();
                SetInitialParameterEstimates();
                ResetGridParameterDeltaValues();
                Iterations[t]=0;
            }
            else
            {
                stop=true;
            }


        }
        if(Iterations[t]>100)
        {
            Convergence=false;
            throw     EConvergenceFailure(this,OutputObject,"Exeeded     "+AnsiString(100)+"     Iterations     on
"+AnsiString(SimulationRepeats)+" occasions!");
        }
```

```
    }
          else
          {
                    Convergence=true;
        IterationsExceeded=false;
                    NewTimeStep=true;
          }
    if(LogConvergence)
        CurrentSimData->LogConvergence();
}


bool __fastcall TFIDOSimulation::AreThereOscillationsAtRecessionFront(void)
{
    if(Components.Contains(peRecession)&&Iterations[t]>10
           &&(   (!_Q->Convergence&&_Q->CheckForOscillations(firstcell))
               ||(!_A->Convergence&&_A->CheckForOscillations(firstcell)) ))
    {
        return true;
    }

    return false;
}


bool __fastcall TFIDOSimulation::AreThereOscillationsAtAdvanceFront(void)
{
    if(!Components.Contains(peInflow)&&Iterations[t]>10&&!_Q->Convergence&&_Q->CheckForOscillations(lastcell))
    {
        if(_A->CheckForOscillations(lastcell))
            return true;
    }
    return false;
}

void __fastcall TFIDOSimulation::CheckForBreakInSimulation(void)
{

}

void __fastcall TFIDOSimulation::IncrementTimeStep(void)
{
    if(!(FieldLengthExceeded))
        ++t;

    NewTimeStep=false;
}

void __fastcall TFIDOSimulation::ResetIterationCount(void)
{
          Convergence=false;
          Iterations[t]=0;
}

void __fastcall TFIDOSimulation::UndoTimeStep(void)
{

    t=t-1;

    SetupMemoryForParameters();
    ResetIterationCount();
    DetermineIrrigationComponents();
    SetSolutionParameters();
    DetermineSolutionCellRange();
    SetSolutionFunctionPointers();
    SetInitialParameterEstimates();

    ResetIterationCount();
    SetInitialParameterEstimates();

    //for(int i=0;i<GridParametersList->Count;++i)
   //     GridParameter[i]->UndoTimeStep();
}

void __fastcall TFIDOSimulation::SetupMemoryForParameters()
{
        int xcount,tcount;
        tcount=t+1;
        if( !(FieldLengthExceeded||FieldLengthReached))
            xcount=tcount;
        else
            xcount=furtherestdownstreamcellindex+1;
}

void __fastcall TFIDOSimulation::ResizeMemory(const int&xcount,const int&tcount)
{
    E.resize(xcount);
    H.resize(xcount);
    F.resize(xcount);
    U.resize(xcount);
    V.resize(xcount);
    Y.resize(xcount);
    W.resize(xcount);

    for(int i=0;i<GridParametersList->Count;++i)
        GridParameter[i]->AdjustMemory(tcount,xcount);

    OutputObject->_A->AdjustMemory(tcount,xcount);
    OutputObject->_Q->AdjustMemory(tcount,xcount);
    OutputObject->_Z->AdjustMemory(tcount,xcount);   //want to account for infilt even after advance receeds.
    OutputObject->_X->AdjustMemory(tcount,xcount);
    OutputObject->_DownstreamCell->AdjustMemory(tcount,0);
```

```
    OutputObject->_UpstreamCell->AdjustMemory(tcount,0);
    OutputObject->_TotalTime->AdjustMemory(tcount,0);
    OutputObject->_dt->AdjustMemory(tcount,0);
    FIterations.resize(tcount);

}

void __fastcall TFIDOSimulation::SaveParameterEstimates(void)
{
    for(int i=0;i<GridParametersList->Count;++i)
        GridParameter[i]->SaveCurrentValues();
    OutputObject->_dt->SaveCurrentValues();
    OutputObject->_A->SaveCurrentValues();
    OutputObject->_Q->SaveCurrentValues();
    OutputObject->_Z->SaveCurrentValues();
    OutputObject->_X->SaveCurrentValues();
    OutputObject->_DownstreamCell->SaveCurrentValues();
          OutputObject->_UpstreamCell->SaveCurrentValues();
    OutputObject->_TotalTime->SaveCurrentValues();
}

void __fastcall TFIDOSimulation::UndoIteration(void)
{
    for(int i=0;i<GridParametersList->Count;++i)
        GridParameter[i]->UndoLastChanges();
    OutputObject->_dt->UndoLastChanges();
    OutputObject->_A->UndoLastChanges();
    OutputObject->_Q->UndoLastChanges();
    OutputObject->_Z->UndoLastChanges();
    OutputObject->_X->UndoLastChanges();
    OutputObject->_DownstreamCell->UndoLastChanges();
          OutputObject->_UpstreamCell->UndoLastChanges();
    OutputObject->_TotalTime->UndoLastChanges();
}

void __fastcall TFIDOSimulation::ResetSimulation(void)
{
    t=0;
    x=0;
    maxxcount=0;
    furtherestdownstreamcellindex=0;
    stop=false;
    CutoffTimeExceeded=false;
    FieldLengthExceeded=false;
    IterationsExceeded=false;
    FieldLengthReached=false;
    Components.Clear();
    ReducedSteps=1;
    TotalIterations=0;
    StopAtPoint=-1;
    StopWhenRunoffOccurrs=false;
    for(int i=0;i<GridParametersList->Count;++i)
    {
        GridParameter[i]->Reset();
        GridParameter[i]->InitialiseNewElementsFromInputData();
    }
    OutputObject->_dt->Reset();
    OutputObject->_A->Reset();
    OutputObject->_Q->Reset();
    OutputObject->_Z->Reset();
    OutputObject->_X->Reset();
    OutputObject->_DownstreamCell->Reset();
          OutputObject->_UpstreamCell->Reset();
    OutputObject->_TotalTime->Reset();

    OutputObject->_A->InitialiseNewElementsFromInputData();
    OutputObject->_Q->InitialiseNewElementsFromInputData();
    OutputObject->_Z->InitialiseNewElementsFromInputData();
    OutputObject->_X->InitialiseNewElementsFromInputData();
    OutputObject->_DownstreamCell->InitialiseNewElementsFromInputData();
          OutputObject->_UpstreamCell->InitialiseNewElementsFromInputData();
    OutputObject->_TotalTime->InitialiseNewElementsFromInputData();



    _A->LowerLimit=0.0001;//0.02*pow(pow(  ManN[0]*(*_Qin)[0] , 2 )/(rho1[0]*So[0])  ,  1.0/rho2[0]);
    NewTimeStep=true;
          if(FOnResetSimulation)
                    FOnResetSimulation();
}

void __fastcall TFIDOSimulation::RemoveUnwantedUpstreamCells(void)
{
    int tempfirstcell=firstcell;
    double tolvalue=A(1,0)*0.05;
    while(tempfirstcell<lastcell&&A(t,tempfirstcell-1)<=tolvalue)   //need to validate this.
    {
        Q(t,tempfirstcell-1)=0;
        Z(t,tempfirstcell-1)+= A(t,tempfirstcell-1)/2.0; //this is an average.
        A(t,tempfirstcell-1)=0;
        ++tempfirstcell;
          }
    UpstreamCell(t)=tempfirstcell;
}

void __fastcall TFIDOSimulation::RemoveUnwantedDownstreamCells(void)
{
          int middlepos=GetPosAtGreatestDepth();
    int endcell=lastcell;
   // double ALast=A(t,middlepos);
```

```
          for(int i=middlepos+1;i<endcell;++i)
    {
        if( A(t,i)<=_A->LowerLimit)
        {
                            DownstreamCell(t)=i;
            Q(t,i)=0;
                    Z(t,i-1)+= A(t,i)/2.0; //this is an average.
                    A(t,i-1)=0;
                            i=endcell;
            lastcell=i;
             Components=Components.Clear();
            Components=Components<<peLateralFlow;
        }
 //       ALast=A(t,i);
        }

    //for(int i=DownstreamCell(t);i<endcell;++i)
//    {
//        A(t,i)=0;
//        Q(t,i)=0;
//    }
}


void __fastcall TFIDOSimulation::SolveEquationsForFirstCellForDistance(void)
{
    CurrentCell=1;
    if(Iterations[t]==10)
    {   A(1,0)      =   0.005;
        Dist(t,1)   =   Q(t,0)*dt(t)/A(t,0);
    }
    if(Iterations[t]==20)
    {
        A(1,0)      =   0.0005;
        Dist(t,1)   =   Q(t,0)*dt(t)/A(t,0);
    }
    CalculateCellParameters();
    //SetDerivativeFunctionPointers();
    a=dRC_dAl();
    d=dRC_dX();
    g=ResidualOfContinuity();
    p=dRM_dAl();
    _s=dRM_dX();
    w=ResidualOfMomentum();
          double denominator=d*p-_s*a;
    DA[0][Iterations[t]-1]=  (g*_s-w*d)/denominator;
    DX[1][Iterations[t]-1]=  (w*a-g*p)/denominator;
}

int __fastcall TFIDOSimulation::GetPosAtGreatestDepth(void)
{
    double max=0;
          int pos=UpstreamCell(t)-1;
          for(int i=pos;i<=DownstreamCell(t);++i)
    {
        if(A(t,i)>max)
        {
            max=A(t,i);
            pos=i;
        }
    }
    return pos;
}

void __fastcall TFIDOSimulation::SetSolutionFunctionPointers(void)
{
 //   if(firstcell==1)
        InitialSolutionDirection=sdTopToBottom;
 //   else
   //     InitialSolutionDirection=sdBottomToTop;
    if(!Components.Contains(peLateralFlow))
    {
          if(t>1&&InitialSolutionDirection==sdTopToBottom)
          {
              FCalculateAuxCoefficients=CalculateAuxCoefficients1;
              FUpdateParameterEstimates=UpdateParameterEstimates1;
          }
          else if(t>1)
          {
              FCalculateAuxCoefficients=CalculateAuxCoefficients2;
              FUpdateParameterEstimates=UpdateParameterEstimates2;
          }
         else
         {
              FCalculateAuxCoefficients=SolveEquationsForFirstCellForDistance;
              FUpdateParameterEstimates=UpdateParameterEstimatesForFirstCell;
         }
    }
    else
    {
        FCalculateAuxCoefficients=0;
        FUpdateParameterEstimates=CalculateInfiltrationForLateralSurfaceFlow;
    }
}


void __fastcall TFIDOSimulation::CalculateAuxCoefficients1(void)
{
    int lastindex=firstcell-1;
    E[lastindex]=0;
    H[lastindex]=0;
          F[lastindex]=0;
```

```
    for (int index=firstcell;index<=lastcell;++index)
    {
        lastindex=index-1;
        CalculateDerivativeValues(index);
        T1=a+b*E[lastindex];
        T2=p+q*E[lastindex];
        T3=e+b*H[lastindex];
        T4=u+q*H[lastindex];
        T5=g+b*F[lastindex];
        T6=w+q*F[lastindex];
        denom=d*T2-_s*T1;
        if(denom!=0)
        {
            E[index]=(  r*T1 -  c*T2 )/denom;
            H[index]=( T4*T1 - T3*T2 )/denom;
            F[index]=( T6*T1 - T5*T2 )/denom;
        }
        else
        {
            E[index]=0;
            H[index]=0;
            F[index]=0;
        }
        U[lastindex]= - c/T1;
        V[lastindex]= - d/T1;
        Y[lastindex]= -T3/T1;
        W[lastindex]= -T5/T1;
    }
}

void __fastcall TFIDOSimulation::CalculateAuxCoefficients2(void)
{
    int nextindex;
    E[lastcell]=0.000001;
    if(Iterations[t]==1)
        H[lastcell]=(dt(t)>0?(Dist(t,lastcell)-Dist(t,lastcell-1))/dt(t):0.01);
    else
        H[lastcell]=(dt(t)>0?(_X->FDeltaValues[t][Iterations[t]-1])/_dt->FDeltaValue[Iterations[t]-1]:0.01);
          F[lastcell]=0;
    for (int index=lastcell;index>=firstcell;--index)
    {
        nextindex=index-1;
        CalculateDerivativeValues(index);
        T1=c+d*E[index];
        T2=r+_s*E[index];
        T3=e+d*H[index];
        T4=u+_s*H[index];
        T5=g+d*F[index];
        T6=w+_s*F[index];
        denom=b*T2-q*T1;

        E[nextindex]=(  p*T1 -  q*T2 )/denom;
        H[nextindex]=( T4*T1 - T3*T2 )/denom;
        F[nextindex]=( T6*T1 - T5*T2 )/denom;
        U[index]= - a/T1;
        V[index]= - b/T1;
        Y[index]= -T3/T1;
        W[index]= -T5/T1;
    }
   // F(lastcell)=F[lastcell-1];
}


void __fastcall TFIDOSimulation::UpdateParameterEstimates1(void)
{
    if(_X->IsSolutionParameter)      UpdateX(lastcell);
    else if(_dt->IsSolutionParameter)UpdateT(lastcell);
    else if(FieldLengthReached)
    {

        double man_mul=pow(rho1(lastcell)*So(lastcell),0.5)/ManN(lastcell);
        double man_exp=rho2(lastcell)/2.0;
        double ar=(A(t,lastcell));//>0?A(t,lastcell):0.00001);
        double dqdar=man_mul*man_exp*pow(ar,man_exp-1);
       // double da=F[lastcell]/(dqdar-E[lastcell]);
//        double dq=E[lastcell]*da+F[lastcell];
        double CC=c+d*dqdar;
        double RR=r+_s*dqdar;

        double da=(T2*T5-T1*T6)/(T1*RR-T2*CC);
 //       _A->Update(lastcell,da);
        double dq=da*man_exp*man_mul*pow(A(t,lastcell),man_exp-1);
        double newa=da+A(t,lastcell);

      //  if(A(t-1,lastcell)==0&&newa>A(t,lastcell-1))
//        {
//            da=A(t,lastcell-1)-A(t,lastcell);
//            _A->Update(lastcell,da);
//             CalculateAuxCoefficients();
//
//        }
//        else
          _A->Update(lastcell,da);


        dq=man_mul*pow(A(t,lastcell),man_exp)-Q(t,lastcell);

        _Q->Update(lastcell,dq);
//
        // if(A(t-1,lastcell)==0&&A(t,lastcell)>A(t,lastcell-1))
//        {
//           A(t,lastcell)=A(t,lastcell-1);
```

```
            Q(t,lastcell)=man_mul*pow(A(t,lastcell),man_exp);
//         }



//        UpdateA_Runoff(lastcell);

     //   double last=Q(t,lastcell);
     //   UpdateQ(lastcell);      2

        //double norm=pow(rho1(lastcell)*So(lastcell),0.5)/ManN(lastcell)*pow(A(t,lastcell),rho2(lastcell)/2.0);
     //   if( Q(t,lastcell)>norm)
     //   {
      //      double temp=Q(t,lastcell);
       //     Q(t,lastcell)=norm;
      //      int iteration=_Q->DeltaValues[lastcell].size()-1;
      //      _Q->DeltaValues[lastcell][iteration]=norm-last;//+=norm-temp;


      //  int iteration=_Q->DeltaValues[lastcell].size()-1;
//        _Q->DeltaValues[lastcell][iteration]=Q(t,lastcell)-last;
//
    }
    if(_X->IsSolutionParameter)
        UpdateA_LastCell(lastcell-1,lastcell);
    else
        UpdateA(lastcell-1,lastcell);
    UpdateQ(lastcell-1);
    for(int index=lastcell-2;index>=firstcell;--index)
    {
        UpdateA(index,index+1);
        UpdateQ(index);
    }
    UpdateA(firstcell-1,firstcell);
}

void __fastcall TFIDOSimulation::UpdateParameterEstimates2(void)
{
    _A->FDeltaValues[firstcell-1][Iterations[t]-1]=0;
    _Q->FDeltaValues[firstcell-1][Iterations[t]-1]=0;
    _A->FDeltaValues[lastcell][Iterations[t]-1]=0;
    _Q->FDeltaValues[lastcell][Iterations[t]-1]=0;
    if(_dt->IsSolutionParameter)    UpdateT(firstcell-1);
    for(int index=firstcell;index<=lastcell-1;++index)
    {
        UpdateA(index,index-1);
        UpdateQ(index);
    }
  //  UpdateA_LastCell(lastcell-1,lastcell);
  //  UpdateQ_LastCell(lastcell-1); //dont think I need to do anthing here.
  //  However, we should check that DX doesn't play a role in any of the other calcs!
  //  should also check that other dqvalue are equal to zero....  lines 2-5 should probably fix that..

    if(_X->IsSolutionParameter)     UpdateX(lastcell);

 //   int i=lastcell;
//    double ddx1=(H[i]*DT + F[i]);
//    double ddx2=H[i]/Y[i]*(-W[i]-U[i]*DA[i-1]-V[i]*DQ[i-1]);
//
//    double ddx=(ddx1==0?ddx1*DampeningFactor:ddx2*DampeningFactor);
//     _X->Update(i,ddx);

}

void __fastcall TFIDOSimulation::UpdateA(const int& i, const int& j)
{
    _A->Update(i,(U[i]*DA[j][Iterations[t]-1]   +   V[i]*DQ[j][Iterations[t]-1]   +   Y[i]*DT[Iterations[t]-1]   +
W[i])*DampeningFactor);
}

void __fastcall TFIDOSimulation::UpdateA_LastCell(const int& i, const int& j)
{                       //
    _A->Update(i,(U[i]*DA[j][Iterations[t]-1]   +   V[i]*DX[j][Iterations[t]-1]   +   Y[i]*DT[Iterations[t]-1]   +
W[i])*DampeningFactor);
}

void __fastcall TFIDOSimulation::UpdateA_Runoff(const int& i)
{
   //                  _A->Update(i,F[i]/(rho2(i)/2.0*pow(rho1(i)*So(i),0.5)/ManN(i)*pow(A(t,i),rho2(i)/2.0-1.0)-
E[i])*DampeningFactor);
    _A->Update(i,(rho2(i)/2.0*pow(rho1(i)*So(i),0.5)/ManN(i)*pow(A(t,i),rho2(i)/2.0-1.0)-
F[i])/E[i]*DampeningFactor);
}

void __fastcall TFIDOSimulation::UpdateQ(const int& i)
{
    _Q->Update(i,(E[i]*DA[i][Iterations[t]-1] + H[i]*DT[Iterations[t]-1] + F[i])*DampeningFactor);
}

void __fastcall TFIDOSimulation::UpdateX(const int& i)
{
    _X->Update(i,(H[i]*DT[Iterations[t]-1] + F[i])*DampeningFactor);
}

void __fastcall TFIDOSimulation::UpdateT(const int& i)
{
    _dt->Update(t,(-F[i]/H[i])*DampeningFactor);
    TotalTime(t)=TotalTime(t-1)+dt(t);
}
```

```
double __fastcall TFIDOSimulation::GetDampeningFactor(void)
{
    return 1.0;
}

void __fastcall TFIDOSimulation::UpdateParameterEstimatesForFirstCell(void)
{
    _X->Update(t,DX[t][Iterations[t]-1]);
    _A->Update(0,DA[0][Iterations[t]-1]);
}

void __fastcall TFIDOSimulation::CalculateDerivativeValues(int cell)
{
    double a2,b2,c2,d2,e2,p2,q2,r2,s2,u2;
    CurrentCell=cell;
    CalculateCellParameters();
    SetDerivativeFunctionPointers();
    g=ResidualOfContinuity();
    a=dRC_dAl();
    b=dRC_dQl();
    if(!(cell==lastcell&&Components.Contains(peRunoff)))
    {
        d=dRC_dParam(); //calc this one first- could get used on next line
        c=dRC_dAr();
    }
    else
    {
        c=dRC_dAr_Runoff();
        d=0;
    }
    e=dRC_dT();
    w=ResidualOfMomentum();
    p=dRM_dAl();
    q=dRM_dQl();
    if(!(cell==lastcell&&Components.Contains(peRunoff)))
    {
        _s=dRM_dParam(); //again...
        r=dRM_dAr();
    }
    else
    {
         r=dRM_dAr_Runoff();
        _s=0;
    }
    u=dRM_dT();
}

void __fastcall TFIDOSimulation::CalculateCellParameters(void)
{
    if(FCurrentCell==firstcell||InitialSolutionDirection==sdBottomToTop)
        CalculateHydraulicParameters(FCurrentCell-1);
    if(FCurrentCell==lastcell||InitialSolutionDirection==sdTopToBottom)
        CalculateHydraulicParameters(FCurrentCell);
    CalculateCellPositions(FCurrentCell);
}

void __fastcall TFIDOSimulation::CalculateHydraulicParameters(int CellSide)
{
    WP(t,CellSide) =CalculateWettedPerimeter(CellSide);
    Z(t,CellSide)  =CalculateWettedPerimeterDependantInfiltration(CellSide);
    P(t,CellSide)  =CalculateHydrostaticPressure(CellSide);
    D(t,CellSide)  =CalculateDragForce(CellSide);
    VP(t,CellSide)  =CalculateVelocityPressureFactor(CellSide);
    dZdT(t,CellSide) =dZ_dT(CellSide);

}

void __fastcall TFIDOSimulation::CalculateEulerianCellPositions(int xpos)
{
    Xrl=Dist(t,xpos)-Dist(t,xpos-1);
    Xmj=Dist(t,xpos)-Dist(t,xpos-1);
    Xrm=0;
    Xlj=0;
}

void __fastcall TFIDOSimulation::CalculateLangrangianCellPositions(int xpos)
{
    if(xpos!=1)
    {
        Xrl=Dist(t,xpos)-Dist(t,xpos-1);
        Xmj=Dist(t-1,xpos-1)-Dist(t-1,xpos-2);
        Xrm=Dist(t,xpos)-Dist(t-1,xpos-1);
        Xlj=Dist(t,xpos-1)-Dist(t-1,xpos-2);
    }
    else
    {
        Xrl=Dist(t,xpos)-Dist(t,xpos-1);
        Xmj=0;
        Xrm=Dist(t,xpos)-Dist(t-1,xpos-1);
        Xlj=0;
    }
}

double __fastcall TFIDOSimulation::ZeroFunction(void)
{
    return 0;
}

double __fastcall TFIDOSimulation::ResidualOfContinuity(void)
{
    return    +(theta*(Ql - Qr) + inv_theta*(Qj - Qm))  * dt(t)
              -(theta*(Al + Zl) + inv_theta*(Aj + Zj))  * Xlj
```

```
              +(theta*(Ar + Zr) + inv_theta*(Am + Zm))  * Xrm
              +(phi*(Aj + Zj)   + inv_phi*(Am + Zm))    * Xmj
              -(phi*(Al + Zl)   + inv_phi*(Ar + Zr))    * Xrl   ;
}

double __fastcall TFIDOSimulation::ResidualOfContinuity_Runoff(void)
{
    double man_mul=pow(rho1(lastcell)*So(lastcell),0.5)/ManN(lastcell);
    double man_exp=rho2(lastcell)/2.0;
    double qr=man_mul*pow(Ar,man_exp);

    return    +(theta*(Ql - qr) + inv_theta*(Qj - Qm))  * dt(t)
              -(theta*(Al + Zl) + inv_theta*(Aj + Zj))  * Xlj
              +(theta*(Ar + Zr) + inv_theta*(Am + Zm))  * Xrm
              +(phi*(Aj + Zj)   + inv_phi*(Am + Zm))    * Xmj
              -(phi*(Al + Zl)   + inv_phi*(Ar + Zr))    * Xrl   ;
}


double __fastcall TFIDOSimulation::ResidualOfMomentum_Runoff(void)
{
    double man_mul=pow(rho1(lastcell)*So(lastcell),0.5)/ManN(lastcell);
    double man_exp=rho2(lastcell)/2.0;
    double qr=man_mul*pow(Ar,man_exp);
    double vpr;
    if(Ar>=_A->LowerLimit)
        vpr= pow(qr,2.0) / (9.81 * Ar)+Pr;
    else
        vpr=0;

    drmdt=       theta*(VPl-vpr)+inv_theta*(VPj-VPm)
                -theta*         (phi * (Dl - SOl * Al) + inv_phi * (Dr - SOr * Ar)) * Xrl
                -inv_theta*     (phi * (Dj - SOj * Aj) + inv_phi * (Dm - SOm * Am)) * Xmj;
    return  (   +(phi   * Qj + inv_phi   * Qm)* Xmj
                -(phi   * Ql + inv_phi   * qr)* Xrl
                +(theta * qr + inv_theta * Qm)* Xrm
                -(theta * Ql + inv_theta * Qj)* Xlj )/9.81 + drmdt*dt(t) ;
}

double __fastcall TFIDOSimulation::ResidualOfMomentum(void)
{
    drmdt=       theta*(VPl-VPr)+inv_theta*(VPj-VPm)
                -theta*         (phi * (Dl - SOl * Al) + inv_phi * (Dr - SOr * Ar)) * Xrl
                -inv_theta*     (phi * (Dj - SOj * Aj) + inv_phi * (Dm - SOm * Am)) * Xmj;
    return  (   +(phi   * Qj + inv_phi   * Qm)* Xmj
                -(phi   * Ql + inv_phi   * Qr)* Xrl
                +(theta * Qr + inv_theta * Qm)* Xrm
                -(theta * Ql + inv_theta * Qj)* Xlj )/9.81 + drmdt*dt(t) ;
}

double __fastcall TFIDOSimulation::RCTip(void)
{
    return theta*Ql*dt(t) -(phi*(Al + Zl))*Xrl;
}

double __fastcall TFIDOSimulation::RMTip(void)
{
         return -(phi*Ql+inv_phi*Qr)* Xrl/9.81  +  theta*(VPl-phi*(Dl-SOl*Al)*Xrl)*dt(t);
}

double __fastcall TFIDOSimulation::dRC_dt(void)
{   //should check this
    return          +(theta*(Ql-Qr)+inv_theta*(Qj-Qm))
                    -(theta*dZdTl  +  inv_theta*dZdTj)  *Xlj
                    +(theta*dZdTr  +  inv_theta*dZdTm)  *Xrm
                    +(phi*dZdTj    +  inv_phi*dZdTm)    *Xmj
                    -(phi*dZdTl    +  inv_phi*dZdTr)    *Xrl
                    ;
}


double __fastcall TFIDOSimulation::dRM_dt(void)
{
    return   drmdt;//this term was calculated when calculating the Residual of Momentum.
}

double __fastcall TFIDOSimulation::dRC_dX(void)
{
         return -phi*(Al + Zl);      //this is like this since it is the only bit to survive the massacre of the
triangular cell tip.
}


double __fastcall TFIDOSimulation::dRM_dX(void)
{
    return -phi*Ql/9.81 -theta*phi*(Dl-SOl*Al)*dt(t);

}

double __fastcall TFIDOSimulation::dZ_dA(const unsigned& xcoord,const unsigned& tcoord)
{
   //     if (A[xcoord,tcoord]==0) return 0;
         //return                     (5.0/2.0-3.0/4.0*rho2)*pow(1.0/rho1,3.0/4.0)*pow(A[xcoord,tcoord],3.0/2.0-
3.0/4.0*rho2)*CalculateKostiakovLewisInfiltration(TotalTime[tcoord]-TotalTime[xcoord]);
    return 0;
}

double __fastcall TFIDOSimulation::dZ_dT(const unsigned& xi)
{
    if(Z(t,xi)==0)return 0;       //is this ok here???
    return (KosA(xi)*KosK(xi)*pow(TotalTime(t)-TotalTime(xi),KosA(xi)-1.0)+KosFo(xi));
}
```

```
double __fastcall TFIDOSimulation::dRC_dAl(void)
{
         return -phi * Xrl -theta*Xlj; // - phi*Xrl*dZ_dA(x-1,t);
}


double __fastcall TFIDOSimulation::dRC_dQl(void)
{
         return theta*dt(t);
}

double __fastcall TFIDOSimulation::dRC_dAr_Normal(void)
{
    if(FCurrentCell==lastcell&&Components.Contains(peRunoff)==false)
        return 0;
    return -inv_phi*Xrl +theta*Xrm;  //-(1-phi)*(Dist[x,t]-Dist[x-1,t])*dZ_dA(x,t);
}

double __fastcall TFIDOSimulation::dRC_dAr_Runoff(void)
{
//       return  rho2(lastcell)/2.0*pow(rho1(lastcell)*So(lastcell),0.5)/ManN(lastcell)*pow(Ar,rho2(lastcell)/2.0-
1.0)*d;
    double drcdar=dRC_dAr_Normal();
    double drcdqr=dRC_dQr();
    return                                                                                          drcdar+
rho2(lastcell)/2.0*pow(rho1(lastcell)*So(lastcell),0.5)/ManN(lastcell)*pow(Ar,rho2(lastcell)/2.0-1.0)*drcdqr;
}

double __fastcall TFIDOSimulation::dRM_dAl(void)
{
    double dVl_dAl   =(Al>0          ? - pow(Ql , 2.0) / (9.81 *pow(Al,2.0))                        :0);
    double dPl_dAl     =(Al>0          ? pow(sigma1l,-1.0/sigma2l)/sigma2l * pow(Al,1.0/sigma2l)       :0);
    double dDl_dAl     =(Al>0&&Ql>0    ? (1.0-rho2l)*pow(ManNl,2.0)/rho1l * pow(Ql,2.0)/pow(Al,rho2l)  :0);
    return (    theta*(dVl_dAl+dPl_dAl)  -  theta*phi*(dDl_dAl-SOl)*Xrl  )*dt(t);
}

double __fastcall TFIDOSimulation::dRM_dAr_Normal(void)
{
    if(FCurrentCell==lastcell&&Components.Contains(peRunoff)==false)
        return 0;
    double dVr_dAr   =(Ar>0      ? -pow(Qr , 2.0) / (9.81 *pow(Ar,2.0))                         :0);
    double dPr_dAr     =(Ar>0       ? pow(sigma1r,-1.0/sigma2r)/sigma2r * pow(Ar,1.0/sigma2r)      :0);
    double dDr_dAr     =(Ar>0&&Qr>0 ? (1.0-rho2r)*pow(ManNr,2.0)/rho1r * pow(Qr,2.0)/pow(Ar,rho2r) :0);
    return (    -theta*(dVr_dAr+dPr_dAr)  -  theta*inv_phi*(dDr_dAr-SOr)*Xrl )*dt(t);
}

double __fastcall TFIDOSimulation::dRM_dAr_Runoff(void)
{
////      return  rho2(lastcell)/2.0*pow(rho1(lastcell)*So(lastcell),0.5)/ManN(lastcell)*pow(Ar,rho2(lastcell)/2.0-
1.0)*_s;
//                                                                                                  return
dRM_dAr_Normal()+rho2(lastcell)/2.0*pow(rho1(lastcell)*So(lastcell),0.5)/ManN(lastcell)*pow(Ar,rho2(lastcell)/2.0-
1.0)*dRM_dQr();

        return
dRM_dAr_Normal()+rho2(lastcell)/2.0*pow(rho1(lastcell)*So(lastcell),0.5)/ManN(lastcell)*pow(Ar,rho2(lastcell)/2.0-
1.0)*dRM_dQr_Runoff();
}

double __fastcall TFIDOSimulation::dRC_dQr(void)
{

         return -theta*dt(t);
}


double __fastcall TFIDOSimulation::dRM_dQl(void)
{
    double dVl_dQl   =(Al>0 ? 2.0*Ql / (9.81 *Al)                                    :0);
    double dDl_dQl     =(Al>0 ?  2.0*pow(ManNl,2.0)/rho1l * Ql * pow(Al,1.0-rho2l) :0);
         return -(phi*Xrl + theta*Xlj)/9.81 + theta*(dVl_dQl - phi*dDl_dQl*Xrl)*dt(t);
}

double __fastcall TFIDOSimulation::dRM_dQr(void)
{

    double dVr_dQr   =(Ar>0  ? 2.0*Qr/(9.81*Ar)                                      :0);
    double dDr_dQr     =(Ar>0  ? 2.0*pow(ManNr,2.0)/rho1r * Qr * pow(Ar,1.0-rho2r) :0);
    return (theta*Xrm-inv_phi*Xrl)/9.81 - theta*(dVr_dQr + inv_phi*dDr_dQr*Xrl)*dt(t);
}

double __fastcall TFIDOSimulation::dRM_dQr_Runoff(void)
{
    double man_mul=pow(rho1(lastcell)*So(lastcell),0.5)/ManN(lastcell);
    double man_exp=rho2(lastcell)/2.0;
    double qr=man_mul*pow(Ar,man_exp);


    double dVr_dQr   =(Ar>0  ? 2.0*qr/(9.81*Ar)                                      :0);
    double dDr_dQr     =(Ar>0  ? 2.0*pow(ManNr,2.0)/rho1r * qr * pow(Ar,1.0-rho2r) :0);
    return (theta*Xrm-inv_phi*Xrl)/9.81 - theta*(dVr_dQr + inv_phi*dDr_dQr*Xrl)*dt(t);
}

double __fastcall TFIDOSimulation::CalculateHydrostaticPressure(int xpos)
{
         if(A(t,xpos)<=_A->LowerLimit)return 0;
         return(pow(sigma1(xpos),-1.0/sigma2(xpos))/(1.0+sigma2(xpos))*pow(A(t,xpos),1.0+(1.0/sigma2(xpos))));
}

double __fastcall TFIDOSimulation::CalculateDragForce(int xpos)
{
```

```
          if(A(t,xpos)<=_A->LowerLimit)return 0;
          return(pow(ManN(xpos),2.0)/rho1(xpos)*pow(Q(t,xpos),2.0)*pow(A(t,xpos),1.0-rho2(xpos)));
}

double __fastcall TFIDOSimulation::CalculateWettedPerimeter(int xpos)
{
    if(A(t,xpos)<=_A->LowerLimit)return 0;
          return pow( 1.0/rho1(xpos) , 3.0/4.0 )*pow(A(t,xpos),5.0/2.0-3.0/4.0*rho2(xpos));
}

double __fastcall TFIDOSimulation::CalculateVelocityPressureFactor(int xpos)
{
    if(A(t,xpos)<=_A->LowerLimit)return 0;
    return pow(Q(t,xpos),2.0) / (9.81 * A(t,xpos))+P(t,xpos);

}

double __fastcall TFIDOSimulation::CalculateWettedPerimeterDependantInfiltration(int xpos)
{
          return CalculateKostiakovLewisInfiltration(xpos);
}

double __fastcall TFIDOSimulation::CalculateKostiakovLewisInfiltration(int xpos)
{
    double OppTime=TotalTime(t)-TotalTime(xpos);
          if(OppTime<=0)return 0;
          return(KosK(xpos)*pow(OppTime,KosA(xpos))+KosFo(xpos)*OppTime);
}


void __fastcall TFIDOSimulation::CalculateInfiltrationForLateralSurfaceFlow(void)
{
    double dz;
    int finalcell=(Components.Contains(peAdvance) ?lastcell-1:lastcell);
    dz=CalculateKostiakovLewisInfiltration(firstcell-1)-Z(t-1,firstcell-1);
    if(dz>0.0001)
    {
        for(int i=firstcell-1;i<=finalcell;++i)
        {

            dz=CalculateKostiakovLewisInfiltration(i)-Z(t-1,i); //this will need changing.
            if(dz<A(t-1,i))
            {
                Z(t,i)=Z(t-1,i)+dz;
                A(t,i)=A(t-1,i)-dz;

            }
            else
            {
                Z(t,i)=Z(t-1,i)+A(t-1,i);
                A(t,i)=0;
                Q(t,i)=0;
                ++firstcell;
                UpstreamCell(t)=firstcell;
            }
        }
        for(int i=lastcell;i>=firstcell-1;--i)
        {
            if(A(t,i)==0)
            {
                --lastcell;
                DownstreamCell(t)=lastcell;
            }
        }
    }
    else
    {
        for(int i=firstcell-1;i<=finalcell;++i)
        {

            {
                Z(t,i)=Z(t-1,i)+A(t-1,i);
                A(t,i)=0;
                Q(t,i)=0;
                firstcell=lastcell;
                UpstreamCell(t)=DownstreamCell(t);
            }
        }

    }

}

void __fastcall TFIDOSimulation::SetInitialParameterEstimatesForFirstCell(void)
{
    double Qi;
    SetTimeStep();
    if(KosA(0)!=1)
        Qi=Qin(1)-(KosA(0)*KosK(0)*pow(dt(1),KosA(0)-1)+KosFo(0));
    else
        Qi=Qin(1)-(KosA(0)*KosK(0)+KosFo(0));
    A(1,0)=pow(pow(Qi*ManN(0),2)/(rho1(0)*So(0)),1.0/rho2(0));
          Dist(t,1)=Qin(1)*dt(1)/A(1,0);
}

void __fastcall TFIDOSimulation::SetInitialParameterEstimates(void)
{
    if(InitialSolutionDirection==sdBottomToTop)
    {
        UpstreamCell(t)=UpstreamCell(t)+1;
        firstcell=UpstreamCell(t);
    }
```

```
     for(int i=0;i<GridParametersList->Count;++i)
        GridParameter[i]->InitialiseNewElementsFromInputData();

    OutputObject->_A->InitialiseNewElementsFromInputData();
    OutputObject->_Q->InitialiseNewElementsFromInputData();
    OutputObject->_Z->InitialiseNewElementsFromInputData();
    OutputObject->_X->InitialiseNewElementsFromInputData();
    OutputObject->_DownstreamCell->InitialiseNewElementsFromInputData();
          OutputObject->_UpstreamCell->InitialiseNewElementsFromInputData();
    OutputObject->_TotalTime->InitialiseNewElementsFromInputData();
    OutputObject->_dt->InitialiseNewElementsFromInputData();
    if(!Components.Contains(peLateralFlow))
    {
    if(t!=1)
    {
        _X->InitialiseAsPrevious();
        if(FieldLengthReached)
        {

            Dist(t,lastcell)=Dist(t-1,lastcell);
        }
        else if(FieldLengthExceeded)
        {
            Dist(t,lastcell)=FieldLength;
            double dx1=Dist(t,lastcell)-Dist(t,lastcell-1);
            double dx2=Dist(t,lastcell-1)-Dist(t,lastcell-2);
            if(dx1<dx2/4.0)
                CombineLastTwoCells();
            SetTimeStep();
            return;
        }
        else
        {
            _X->InitialiseAsPrevious();
            Dist(t,lastcell)=Dist(t-1,lastcell-1)+(Dist(t-1,lastcell-1) - Dist(t-1,lastcell-2));
        }

        _Z->InitialiseAsPrevious();

        if(Components.Contains(peAdvance)||Components.Contains(peRecession))
        {
            _Q->InitialiseAsPreviousAndLast();
            _A->InitialiseAsPreviousAndLast();
        }
        else
        {
            _Q->InitialiseAsPrevious();//AndLast();//
            _A->InitialiseAsPrevious();//AndLast();//
        }


        if(Components.Contains(peRunoff))
        {
            if(A(t-1,lastcell)==0)
            {

                if(lastcell>4)
                    A(t,lastcell)=A(t-1,lastcell-5);
                else if(lastcell>3)
                    A(t,lastcell)=A(t-1,lastcell-4);
                else if(lastcell>2)
                    A(t,lastcell)=A(t-1,lastcell-3);
                else if(lastcell>1)
                    A(t,lastcell)=A(t-1,lastcell-2);
                else if(lastcell>0)
                    A(t,lastcell)=A(t-1,lastcell-1);
            }

                 else
            {
                A(t,lastcell)=A(t-1,lastcell);

            }

Q(t,lastcell)=pow(rho1(lastcell)*So(lastcell),0.5)/ManN(lastcell)*pow(A(t,lastcell),rho2(lastcell)/2.0);
        }
    }
    else
        SetInitialParameterEstimatesForFirstCell();
    if(Components.Contains(peInflow))
        Q(t,0)=Qin(t);
    else
        Q(t,0)=0;
    if(Components.Contains(pePonding))
        Q(t,lastcell)=0;//sqrt(rho1(lastcell)*So(lastcell))/ManN(lastcell)* pow(A(t,lastcell),rho2(lastcell)/2.0);




     SetTimeStep();
    if(Components.Contains(peRecession))
    {

        double dz;
        int finalcell= (Components.Contains(peAdvance)?lastcell-1:lastcell);
        for(int i=firstcell-1;i<=finalcell;++i)
        {
            dz=CalculateKostiakovLewisInfiltration(i)-Z(t-1,i); //this will need changing.
            if(dz>A(t-1,i))
            {
```

```
                Z(t,i)=Z(t-1,i)+A(t-1,i);
                A(t,i)=0;
                Q(t,i)=0;

                for(int cell=i+i;cell<=finalcell;++cell)
                {
                    if(cell<lastcell)
                    {
                        A(t,cell)=(A(t,cell)+A(t,cell+1))/2.0;
                        Q(t,cell)=(Q(t,cell)+Q(t,cell+1))/2.0;
                    }
                }
                ++firstcell;
                UpstreamCell(t)=firstcell;
            }
            else
                i=finalcell+1;
        }

    }
    if(InitialSolutionDirection==sdBottomToTop)
    {
        for(int i=firstcell-1;i<=lastcell;++i)
        {
            A(t,i)=A(t,i)*0.1;
            Q(t,i)=Q(t,i)*0.1;

        }
        A(t,firstcell-1)=0;//.000001;
        Q(t,firstcell-1)=0;
        SolveForT=true;
    }
    }
    else
    {
        _Q->InitialiseAsZero();
        _A->InitialiseAsZero();
        _Z->InitialiseAsPrevious();
        _X->InitialiseAsPrevious();
         SetTimeStep();
    }

}


void __fastcall TFIDOSimulation::SetTimeStep(void)
{
    if(!FieldLengthExceeded)
    {
        if(t<=5)
            dt(t)=double(t)/5.0*SimulationTimeStep;
        else //(t>5&&t<200)
        {
             dt(t)=SimulationTimeStep;
        }

    }
    else
        dt(t)=SimulationTimeStep*(Dist(t,lastcell)-Dist(t,lastcell-1))/(Dist(t-1,lastcell-1)-Dist(t-1,lastcell-
2));
    TotalTime(t)=TotalTime(t-1)+dt(t);
}

void __fastcall TFIDOSimulation::SetSolutionParameters(void)
{
    SolveForX=(Components.Contains(peAdvance)&&!FieldLengthExceeded);
    SolveForT=FieldLengthExceeded;//(Components.Contains(peRecession));
    SolveForRunoff=Components.Contains(peRunoff);
}


TGridType __fastcall TFIDOSimulation::GetGridType()
{
    return FGridType;
}

void __fastcall TFIDOSimulation::SetGridType(TGridType type)
{
    FGridType=type;
    if(type==gtEulerian)
        FCalculateCellPositions=CalculateEulerianCellPositions;
    else
        FCalculateCellPositions=CalculateLangrangianCellPositions;
}

int __fastcall TFIDOSimulation::GetCellCount(void)
{
    return DownstreamCell(t)-UpstreamCell(t)+1;
}

void __fastcall TFIDOSimulation::SetDerivativeFunctionPointers(void)
{
    if(FCurrentCell!=lastcell)
    {
        dRC_dAr=dRC_dAr_Normal;
        dRM_dAr=dRM_dAr_Normal;
        dRC_dParam=dRC_dQr;
        dRM_dParam=dRM_dQr;
    }
    else
    {
        dRC_dParam=dRC_dParam_LastCell;
```

```
        dRM_dParam=dRM_dParam_LastCell;
        dRC_dAr=dRC_dAr_LastCell;
        dRM_dAr=dRM_dAr_LastCell;
    }
}

void __fastcall TFIDOSimulation::SetSolveForT(bool value)
{
    _TotalTime->IsSolutionParameter=value;
    _dt->IsSolutionParameter=value;
    if(value)
    {
        dRC_dT=dRC_dt;
        dRM_dT=dRM_dt;
    }
    else
    {
        dRC_dT=ZeroFunction;
        dRM_dT=ZeroFunction;
    }
}

void __fastcall TFIDOSimulation::SetSolveForX(bool value)
{
    _X->IsSolutionParameter=value;
    if(value)
    {
        dRC_dParam_LastCell=dRC_dX;
        dRM_dParam_LastCell=dRM_dX;
    }
    else
    {
        dRC_dParam_LastCell=dRC_dQr;
        dRM_dParam_LastCell=dRM_dQr;
    }
}

void __fastcall TFIDOSimulation::SetSolveForRunoff(bool value)
{
    if(value)
    {
        dRC_dAr_LastCell=dRC_dAr_Runoff;
        dRM_dAr_LastCell=dRM_dAr_Runoff;
    }
    else
    {
        dRC_dAr_LastCell=dRC_dAr_Normal;
        dRM_dAr_LastCell=dRM_dAr_Normal;
    }
}

bool __fastcall TFIDOSimulation::GetCutoffTimeReached(void)
{
    return CutoffTimeExceeded;
}
bool __fastcall TFIDOSimulation::GetCellFlowsAreNegligible(void)
{
    for(int i=firstcell;i<=lastcell;++i)
    {
        if( Q(t-1,i)>0.001*Q(1,0) )
            return false;

        if(Q(t-1,i)>0.001*Q(1,0))
            return false;
    }
    return true;
}

bool __fastcall TFIDOSimulation::GetAllowRunoff(void)
{
    return true;
}

void __fastcall TFIDOSimulation::SetCurrentSimData(TSimulationParametersObject*newrecord)
{
    FCurrentSimData=newrecord;
    OutputObject=FCurrentSimData->OutputObject;
    FCurrentSimData->UpdateModelParameters();
}

void __fastcall TFIDOSimulation::SetOutputObject(TFIDOOutputTreeObject*object)
{
    FOutputObject=object;
    FOutputObject->ErrorMessage="";
    FOutputObject->IsHappy=true;
    FOutputObject->LinkSolutionParameters();
}

TSimulationParametersObject* __fastcall TFIDOSimulation::GetCurrentSimData()
{
    return FCurrentSimData;
}

TFIDOOutputTreeObject* __fastcall TFIDOSimulation::GetOutputObject()
{
    return FOutputObject;
}


void __fastcall TFIDOSimulation::UpdateOutputObjectProperties(void)
{
    if(t>0)
```

```
    {
        FOutputObject->EndCell=lastcell;
        FOutputObject->FurtherestDownstreamCellIndex=furtherestdownstreamcellindex;
        FOutputObject->TotalIterations=TotalIterations;
        FOutputObject->MaxFlowDepth=pow(pow(  ManN(0)*(*_Qin)(1) , 2 )/(rho1(0)*So(0))  ,  1.0/rho2(0));;
        FOutputObject->MaxFlowDepth=pow(FOutputObject->MaxFlowDepth/sigma1(1), 1.0/sigma2(1));
        if(t>0&&Z(t,0)>Z(t-1,0))
            FOutputObject->MaxZ=Z(t,0);
        else
            FOutputObject->MaxZ=Z(t-1,0);

        FOutputObject->NumberTimeSteps=t;
        FOutputObject->_sigma1->Resize(_sigma1->Size);
        FOutputObject->_sigma2->Resize(_sigma2->Size);
        for(int i=0;i<_sigma1->Size;++i)
        {
            (*FOutputObject->_sigma1)(i)=sigma1(i);
            (*FOutputObject->_sigma2)(i)=sigma2(i);
        }  // tidy this up later with by creating and assign fn.
        FOutputObject->HasBeenSimulated=true;
        FOutputObject->CalculatePerformanceValues(t);
    }
}

// finite difference approximation... probably wont use it...
double __fastcall TFIDOSimulation::dRMd(double&param)
{
          double r1,r2,tem;
    double tol=param*0.01;
    CalculateCellParameters();
          r1=ResidualOfMomentum();
    tem=param;
    param+=tol;
    CalculateCellParameters();
    r2=ResidualOfMomentum();
          param=tem;
    CalculateCellParameters();
    return (r2-r1)/tol;
}

// finite difference approximation... probably wont use it...
double __fastcall TFIDOSimulation::dRCd(double&param)
{
          double r1,r2,tem;
    double tol=param*0.01;
    CalculateCellParameters();
          r1=ResidualOfContinuity();
    tem=param;
    param+=tol;
    CalculateCellParameters();
    r2=ResidualOfContinuity();
          param=tem;
    CalculateCellParameters();
    return (r2-r1)/tol;
}


void __fastcall TFIDOSimulation::CombineLastTwoCells(void)
{
    Dist(t,lastcell-1)  =   Dist(t,lastcell);
    A(t,lastcell-1)     =   A(t,lastcell);
    Q(t,lastcell-1)     =   A(t,lastcell);
    Z(t,lastcell-1)     =   A(t,lastcell);
    //not sure if these are requrired... better to play safe
    D(t,lastcell-1)     =   D(t,lastcell);
    P(t,lastcell-1)     =   P(t,lastcell);
    WP(t,lastcell-1)    =   WP(t,lastcell);
    VP(t,lastcell-1)    =   VP(t,lastcell);
    dZdT(t,lastcell-1)  =   dZdT(t,lastcell);

 //   ++RemovedCellCount;
    --lastcell;
    if(furtherestdownstreamcellindex==lastcell+1)
        furtherestdownstreamcellindex=lastcell;
    --DownstreamCell(t);
    SetupMemoryForParameters();


}



void __fastcall TFIDOSimulation::RefineGrid(const int& time)
{
    SmartPointer<TCurveFit>AFit=CreateCurveFit();
    SmartPointer<TCurveFit>ZFit=CreateCurveFit();
    SmartPointer<TCurveFit>QFit=CreateCurveFit();
    SmartPointer<TCurveFit>DFit=CreateCurveFit();
    SmartPointer<TCurveFit>PFit=CreateCurveFit();
    SmartPointer<TCurveFit>WPFit=CreateCurveFit();
    int firstcell =UpstreamCell(time);
    int endcell= DownstreamCell(time);
          for (int i=firstcell-1;i<=endcell;++i)
    {
        AFit->EnterStatValue (  Dist(time,i) , A(time,i)    );
        ZFit->EnterStatValue (  Dist(time,i) , Z(time,i)    );
        QFit->EnterStatValue (  Dist(time,i) , Q(time,i)    );
        DFit->EnterStatValue (  Dist(time,i) , D(time,i)    );
        PFit->EnterStatValue (  Dist(time,i) , P(time,i)    );
        WPFit->EnterStatValue(  Dist(time,i) , WP(time,i)   );
    }
    double dx=double(Dist(time,endcell)-Dist(time,firstcell-1))/double(endcell-(firstcell-1));
```

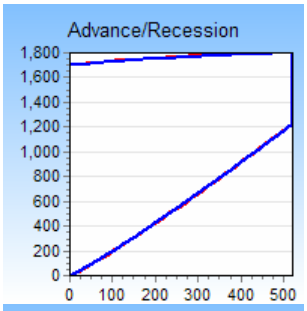
```
    double first=Dist(time,firstcell-1);
    int count=0;
    for (int i=firstcell-1;i<endcell;++i)
    {
        Dist(time,i)=first+double(count)*dx;
        ++count;
    }

    for (int i=firstcell-1;i<=endcell;++i)
    {
        if(Dist(time,i)>0)
        {
            A(time,i) = AFit->CubicSpline (Dist(time,i));
            Z(time,i) = ZFit->CubicSpline (Dist(time,i));
            Q(time,i) = QFit->CubicSpline (Dist(time,i));
            D(time,i) = DFit->CubicSpline (Dist(time,i));
            P(time,i) = PFit->CubicSpline (Dist(time,i));
            WP(time,i)= WPFit->CubicSpline(Dist(time,i));

            if(A(time,i)<0) A(time,i)=0;
            if(Z(time,i)<0) Z(time,i)=0;
            if(Q(time,i)<0) Q(time,i)=0;
            if(D(time,i)<0) D(time,i)=0;
            if(P(time,i)<0) P(time,i)=0;
            if(WP(time,i)<0)WP(time,i)=0;
        }
    }
    ResetIterationCount();
    SetInitialParameterEstimates();
}

SmartPointer<TCurveFit>__fastcall TFIDOSimulation::CreateCurveFit(void)
{
    SmartPointer<TCurveFit>Fit(new TCurveFit);
    Fit->Init();
    return Fit;
}



void __fastcall TFIDOSimulation::EnableConvergenceLogging(void)
{
    LogConvergence=true;
}


void __fastcall TFIDOSimulation::DisableConvergenceLogging(void)
{
    LogConvergence=false;
}


void __fastcall TFIDOSimulation::ResetGridParameterDeltaValues(void)
{
     _X->ResetDeltaValues();
    _A->ResetDeltaValues();
    _Q->ResetDeltaValues();
    _TotalTime->ResetDeltaValues();
    _dt->ResetDeltaValues();
}




void __fastcall TFIDOSimulation::SetStopAtPoint(int value)
{
    FStopAtPoint = value;
}

int __fastcall TFIDOSimulation::GetStopAtPoint()
{
    return FStopAtPoint;
}
```

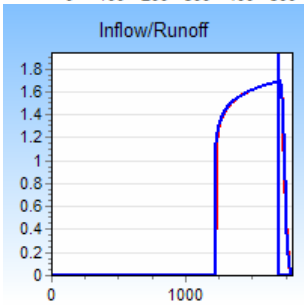
Appendix 3.2 Validation of FIDO Simulation Engine against SIRMOD Output.

Validation Output: C_Turner's Property

Field 19 7/10/2000 Furrow 1 Irrigation no:1

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\Turner_00-01\TURNER_F19\TUR00_fld19irr1fur1.cfg

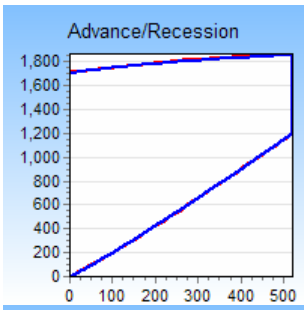


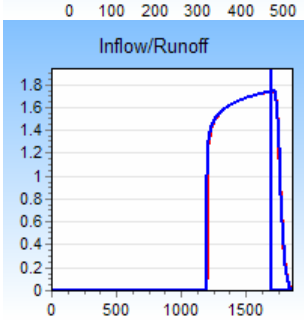


Flowrate (m^3/sec)	0.00194 *NOTE: INFILT value =1.941667 l/sec*
Time-to-cutoff (mins)	1690 *NOTE: SIRMOD value =1690 minsOld Value 1689*
Field-length (m)	520
Field-slope	0.00151
Manning n	0.03
Kostiakov a	0.10155
Kostiakov k (m^3/min^a/m)	0.13916
Kostiakov fo (m^3/min/m)	0
Z-required (m)	280 *NOTE: SIRMOD value =0.28 mOld Value 0.111*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 19 7/10/2000 Furrow ave Irrigation no:1

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\Turner_00-1\TURNER_F19\TUR00_fld19irr1furAVG.cfg

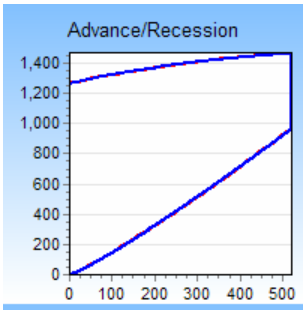


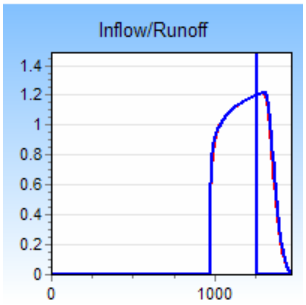


Flowrate (m^3/sec)	0.00194 *NOTE: INFILT value =1.941667 l/sec*
Time-to-cutoff (mins)	1690 *NOTE: SIRMOD value =1690 minsOld Value 1689*
Field-length (m)	520
Field-slope	0.00151
Manning n	0.05
Kostiakov a	0.08592 *NOTE: SIRMOD value =0.08592*
Kostiakov k (m^3/min^a/m)	0.1469 *NOTE: SIRMOD value =0.1469*
Kostiakov fo (m^3/min/m)	0
Z-required (m)	222 *NOTE: SIRMOD value =0.222 mOld Value 0.111*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 19 31/12/2000 Furrow ave Irrigation no:3

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\Turner_00-1\TURNER_F19\TUR00_fld19irr3furAVG.cfg

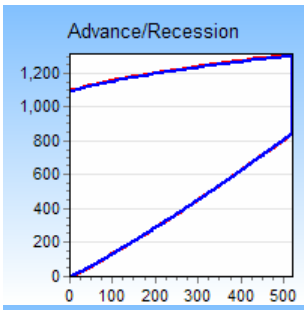


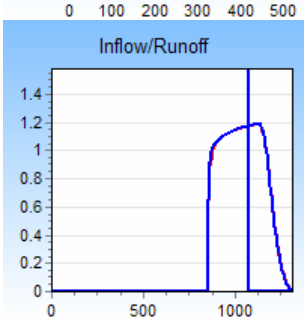


Flowrate (m^3/sec)	0.001486 *NOTE: SIRMOD value =1.486 lps NOTE: INFILT value =1.486667 l/sec*
Time-to-cutoff (mins)	1248
Field-length (m)	520
Field-slope	0.00151
Manning n	0.08
Kostiakov a	0.13823 *NOTE: SIRMOD value =0.13823*
Kostiakov k (m^3/min^a/m)	0.06337 *NOTE: SIRMOD value =0.06337*
Kostiakov fo (m^3/min/m)	0
Z-required (m)	144 *NOTE: SIRMOD value =0.144 mOld Value 0.067*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 19 12/01/2001 Furrow ave Irrigation no:4

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\Turner_00-1\TURNER_F19\TUR01_fld19irr4furAVG.cfg

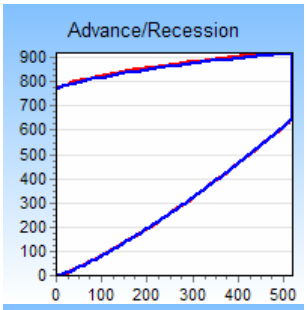


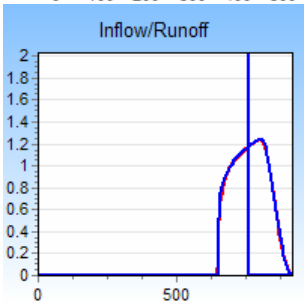


Flowrate (m^3/sec)	0.00158
Time-to-cutoff (mins)	1070
Field-length (m)	520
Field-slope	0.00151
Manning n	0.1
Kostiakov a	0.07352
Kostiakov k (m^3/min^a/m)	0.07585
Kostiakov fo (m^3/min/m)	0.00003 *NOTE: SIRMOD value =0*
Z-required (m)	130 *NOTE: SIRMOD value =0.13 mOld Value 0.068*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 19 24/01/2001 Furrow ave Irrigation no:5

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\Turner_00-1\TURNER_F19\TUR01_fld19irr5furAVG.cfg

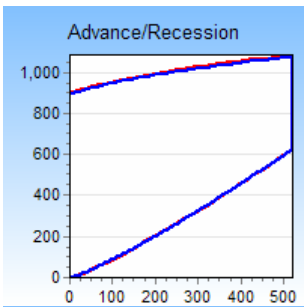


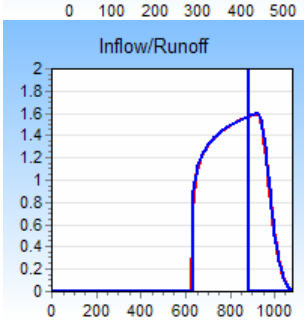


Flowrate (m^3/sec)	0.00203
Time-to-cutoff (mins)	755
Field-length (m)	520
Field-slope	0.00151
Manning n	0.08
Kostiakov a	0.2198 *NOTE: SIRMOD value =0.2198*
Kostiakov k (m^3/min^a/m)	0.03456 *NOTE: SIRMOD value =0.03456*
Kostiakov fo (m^3/min/m)	0.00002 *NOTE: SIRMOD value =0*
Z-required (m)	112 *NOTE: SIRMOD value =0.112 mOld Value 0.056*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 19 12/02/2001 Furrow ave Irrigation no:6

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\Turner_00-1\TURNER_F19\TUR01_fld19irr6furAVG.cfg

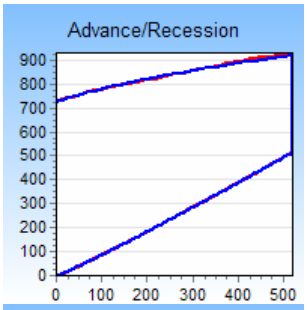


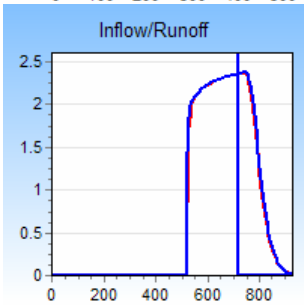


Flowrate (m^3/sec)	0.002
Time-to-cutoff (mins)	875 *NOTE: SIRMOD value =875 minsOld Value 445*
Field-length (m)	520
Field-slope	0.00151
Manning n	0.08
Kostiakov a	0.18432 *NOTE: SIRMOD value =0.18432*
Kostiakov k (m^3/min^a/m)	0.04261 *NOTE: SIRMOD value =0.04261*
Kostiakov fo (m^3/min/m)	0
Z-required (m)	122 *NOTE: SIRMOD value =0.122 mOld Value 0.061*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 19 26/02/2001 Furrow ave Irrigation no:7

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\Turner_00-1\TURNER_F19\TUR01_fld19irr7furAVG.cfg

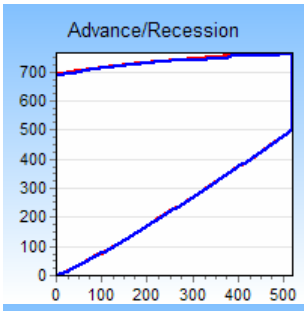


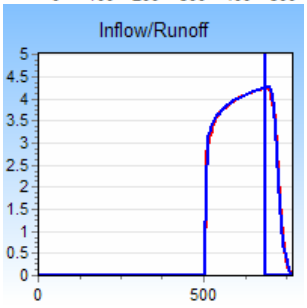


Flowrate (m^3/sec)	0.0026
Time-to-cutoff (mins)	705
Field-length (m)	520
Field-slope	0.00151
Manning n	0.07
Kostiakov a	0.082 *NOTE: SIRMOD value =0.082*
Kostiakov k (m^3/min^a/m)	0.08137 *NOTE: SIRMOD value =0.08137*
Kostiakov fo (m^3/min/m)	0
Z-required (m)	118 *NOTE: SIRMOD value =0.118 mOld Value 0.059*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 20 9/10/2000 Furrow ave Irrigation no:1

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\Turner_00-1\TURNER_F20\TUR00_fld20irr1furAVG.cfg

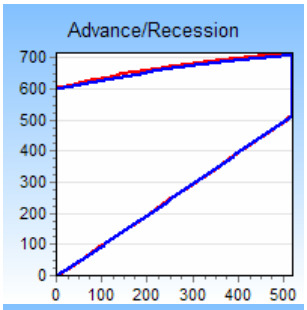


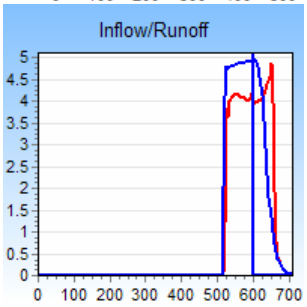


Flowrate (m^3/sec)	0.00505
Time-to-cutoff (mins)	680
Field-length (m)	520
Field-slope	0.00151
Manning n	0.03
Kostiakov a	0.12189 *NOTE: SIRMOD value =0.12189*
Kostiakov k (m^3/min^a/m)	0.13952 *NOTE: SIRMOD value =0.13952*
Kostiakov fo (m^3/min/m)	0
Z-required (m)	266 *NOTE: SIRMOD value =0.266 mOld Value 0.134*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 20 10/12/2000 Furrow ave Irrigation no:2

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\Turner_00-1\TURNER_F20\TUR00_fld20irr2furAVG.cfg

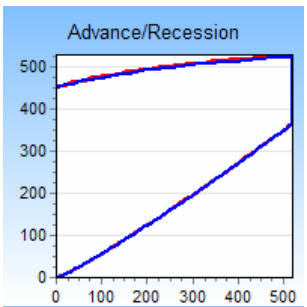


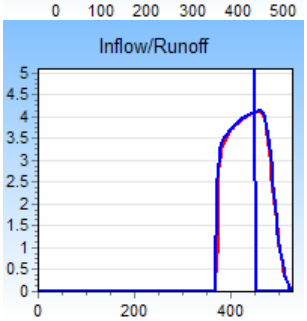


Flowrate (m^3/sec)	0.00511
Time-to-cutoff (mins)	594
Field-length (m)	520
Field-slope	0.00151
Manning n	0.03
Kostiakov a	0.01991 *NOTE: SIRMOD value =0.01991*
Kostiakov k (m^3/min^a/m)	0.24978 *NOTE: SIRMOD value =0.24978*
Kostiakov fo (m^3/min/m)	0 *NOTE: SIRMOD value =0.000000*
Z-required (m)	286.608 *NOTE: SIRMOD value =0.286608 mOld Value 0.098*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 20 1/01/2001 Furrow ave Irrigation no:3

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\Turner_00-1\TURNER_F20\TUR01_fld20irr3furAVG.cfg

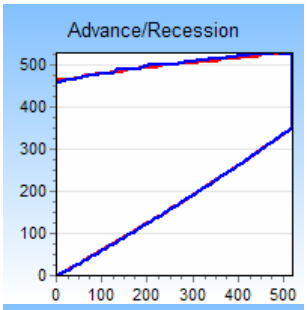


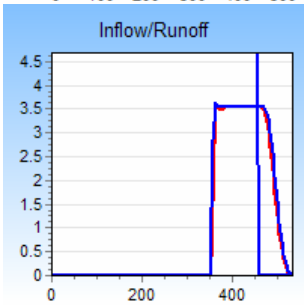


Flowrate (m^3/sec)	0.0051
Time-to-cutoff (mins)	445 *NOTE: SIRMOD value =445 minsOld Value 875*
Field-length (m)	520
Field-slope	0.00151
Manning n	0.03
Kostiakov a	0.12373
Kostiakov k (m^3/min^a/m)	0.10257
Kostiakov fo (m^3/min/m)	0
Z-required (m)	162 *NOTE: SIRMOD value =0.162 mOld Value 0.081*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 20 11/01/2001 Furrow ave Irrigation no:4

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\Turner_00-1\TURNER_F20\TUR01_fld20irr4furAVG.cfg

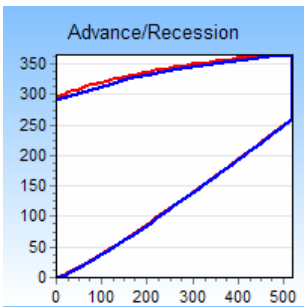


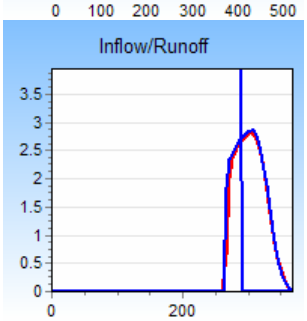


Flowrate (m^3/sec)	0.00468
Time-to-cutoff (mins)	450
Field-length (m)	520
Field-slope	0.00151
Manning n	0.03
Kostiakov a	0 *NOTE: SIRMOD value =0.000000*
Kostiakov k (m^3/min^a/m)	0.14413 *NOTE: SIRMOD value =0.14413*
Kostiakov fo (m^3/min/m)	0.00013 *NOTE: SIRMOD value =0.000002*
Z-required (m)	302.13 *NOTE: SIRMOD value =0.30213 mOld Value 0.079*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 20 29/01/2001 Furrow ave Irrigation no:5

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\Turner_00-1\TURNER_F20\TUR01_fld20irr5furAVG.cfg

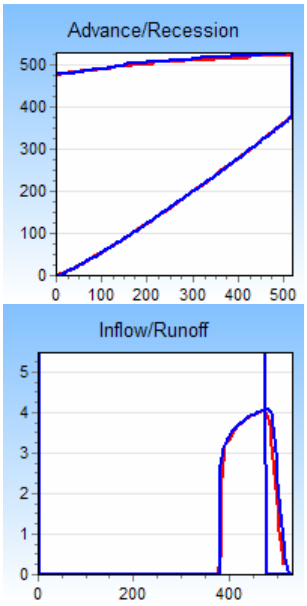




Flowrate (m^3/sec)	0.00395 *NOTE: SIRMOD value =3.95 lps NOTE: INFILT value =3.95 l/sec*
Time-to-cutoff (mins)	285
Field-length (m)	520
Field-slope	0.00151
Manning n	0.03
Kostiakov a	0.1606
Kostiakov k (m^3/min^a/m)	0.0464 *NOTE: SIRMOD value =0.0464*
Kostiakov fo (m^3/min/m)	0
Z-required (m)	140 *NOTE: SIRMOD value =0.14 mOld Value 0.072*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 20 13/02/2001 Furrow ave Irrigation no:6

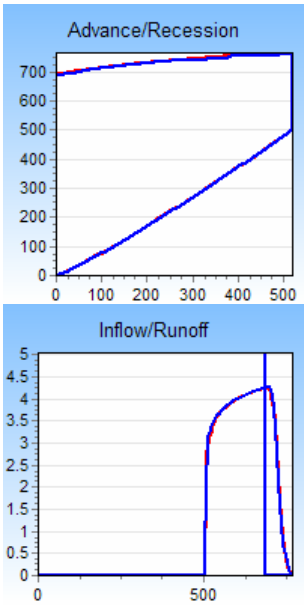
SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\Turner_00-1\TURNER_F20\TUR01_fld20irr6furAVG.cfg



Flowrate (m^3/sec)	0.00548
Time-to-cutoff (mins)	468
Field-length (m)	520
Field-slope	0.00151
Manning n	0.02
Kostiakov a	0.13235 *NOTE: SIRMOD value =0.13235*
Kostiakov k (m^3/min^a/m)	0.10811 *NOTE: SIRMOD value =0.10811*
Kostiakov fo (m^3/min/m)	0.00004 *NOTE: SIRMOD value =0.000001*
Z-required (m)	200 *NOTE: SIRMOD value =0.2 mOld Value 0.1*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 20 26/02/2001 Furrow ave Irrigation no:7

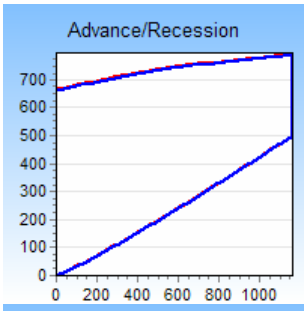
SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\Turner_00-1\TURNER_F20\TUR01_fld20irr7furAVG.cfg

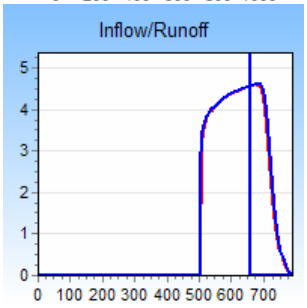


Flowrate (m^3/sec)	0.00505 *NOTE: SIRMOD value =5.05 lps*
Time-to-cutoff (mins)	680 *NOTE: SIRMOD value =680 minsOld Value 425*
Field-length (m)	520
Field-slope	0.00151
Manning n	0.03
Kostiakov a	0.12189 *NOTE: SIRMOD value =0.12189*
Kostiakov k (m^3/min^a/m)	0.13952 *NOTE: SIRMOD value =0.13952*
Kostiakov fo (m^3/min/m)	0
Z-required (m)	266 *NOTE: SIRMOD value =0.266 mOld Value 0.095*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 17 11/01/2000 Furrow 8 Irrigation no:4

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\Turner_99-00\TURNER_F17\TUR00_fld17irr4fur8.cfg

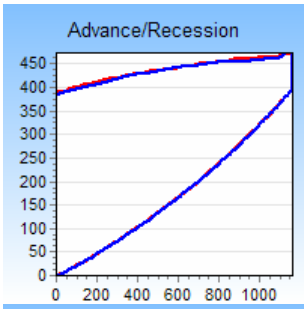


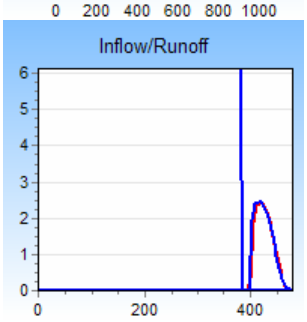


Flowrate (m^3/sec)	0.00536
Time-to-cutoff (mins)	650
Field-length (m)	1160
Field-slope	0.00141
Manning n	0.02
Kostiakov a	0.11259 *NOTE: SIRMOD value =0.11259*
Kostiakov k (m^3/min^a/m)	0.06531 *NOTE: SIRMOD value =0.06531*
Kostiakov fo (m^3/min/m)	0
Z-required (m)	142 *NOTE: SIRMOD value =0.142 mOld Value 0.065*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 17 20/01/2000 Furrow 8 Irrigation no:5

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\Turner_99-00\TURNER_F17\TUR00_fld17irr5fur8.cfg

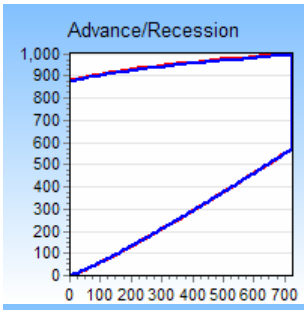


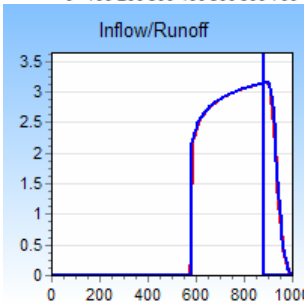


Flowrate (m^3/sec)	0.00613
Time-to-cutoff (mins)	380
Field-length (m)	1160
Field-slope	0.00141
Manning n	0.02
Kostiakov a	0.0744 *NOTE: SIRMOD value =0.0744*
Kostiakov k (m^3/min^a/m)	0.05012 *NOTE: SIRMOD value =0.05012*
Kostiakov fo (m^3/min/m)	0.00015 *NOTE: SIRMOD value =0.000003*
Z-required (m)	128 *NOTE: SIRMOD value =0.128 mOld Value 0.064*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 18 1/10/1999 Furrow 8 Irrigation no:1

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\Turner_99-00\TURNER_F18\TUR99_fld18irr1fur8.cfg

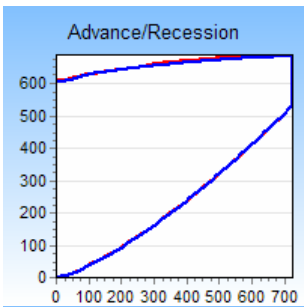


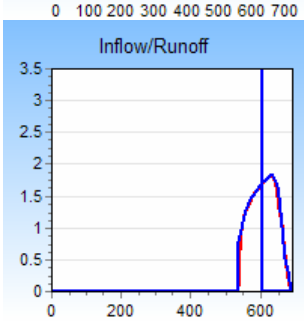


Flowrate (m^3/sec)	0.00364
Time-to-cutoff (mins)	873
Field-length (m)	725 *NOTE: SIRMOD value =725 mOld Value 750*
Field-slope	0.00151
Manning n	0.03
Kostiakov a	0.13103 *NOTE: SIRMOD value =0.13103*
Kostiakov k (m^3/min^a/m)	0.07486 *NOTE: SIRMOD value =0.07486*
Kostiakov fo (m^3/min/m)	0
Z-required (m)	140 *NOTE: SIRMOD value =0.14 mOld Value 0.08*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 18 11/01/2000 Furrow 2 Irrigation no:4

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\Turner_99-00\TURNER_F18\TUR00_fld18irr4fur8.cfg



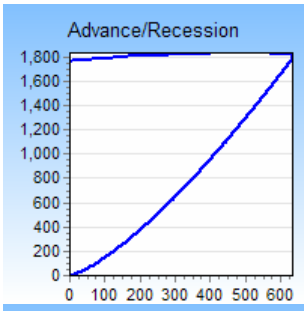


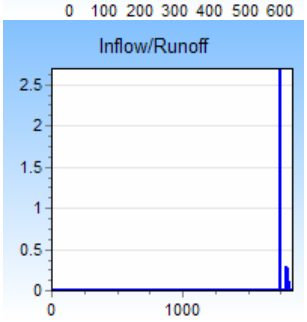
Flowrate (m^3/sec)	0.0035
Time-to-cutoff (mins)	600
Field-length (m)	725 *NOTE: SIRMOD value =725 mOld Value 750*
Field-slope	0.00151
Manning n	0.03
Kostiakov a	0.29564 *NOTE: SIRMOD value =0.29564*
Kostiakov k (m^3/min^a/m)	0.02655 *NOTE: SIRMOD value =0.02655*
Kostiakov fo (m^3/min/m)	0
Z-required (m)	114 *NOTE: SIRMOD value =0.114 mOld Value 0.059*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Validation Output: N_Walton's Property

Field 7a 25/09/2001 Furrow 2 Irrigation no:1

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\waltons\WAL_fldAirr1fur2.cfg

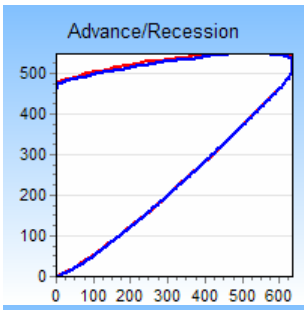


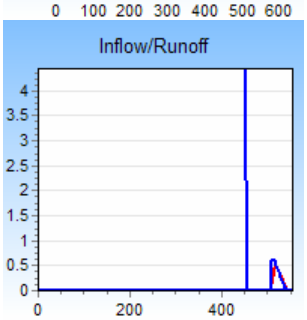


Flowrate (m^3/sec)	0.0027 *NOTE: SIRMOD value =2.7 lps*
Time-to-cutoff (mins)	1745
Field-length (m)	635
Field-slope	0.001
Manning n	0.035
Kostiakov a	0.27002
Kostiakov k (m^3/min^a/m)	0.07049
Kostiakov fo (m^3/min/m)	0
Z-required (m)	300 *NOTE: SIRMOD value =0.3 mOld Value 0.15*
Furrow top width (m)	0.4
Furrow mid width (m)	0.25
Furrow bot width (m)	0.1
Furrow max depth (m)	0.25

Field 7a 27/12/2001 Furrow 2 Irrigation no:2

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\waltons\WAL_fldAirr2fur2.cfg

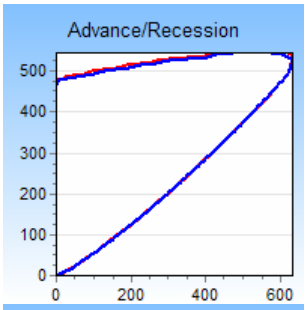


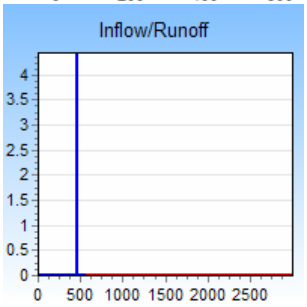


Flowrate (m^3/sec)	0.004442 *NOTE: SIRMOD value =4.442 lps*
Time-to-cutoff (mins)	450 *NOTE: SIRMOD value =450 minsOld Value 495*
Field-length (m)	635
Field-slope	0.001
Manning n	0.04
Kostiakov a	0.19263 *NOTE: SIRMOD value =0.19263*
Kostiakov k (m^3/min^a/m)	0.06375 *NOTE: SIRMOD value =0.06375*
Kostiakov fo (m^3/min/m)	0
Z-required (m)	199.998 *NOTE: SIRMOD value =0.199998 mOld Value 0.078*
Furrow top width (m)	0.4
Furrow mid width (m)	0.25
Furrow bot width (m)	0.1
Furrow max depth (m)	0.25

Field 7a 27/12/2001 Furrow 6 Irrigation no:2

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\waltons\WAL_fldAirr2fur6.cfg

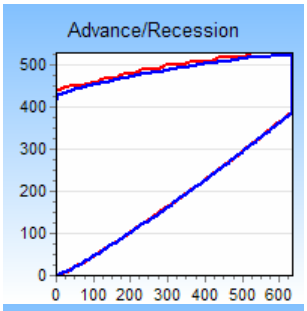


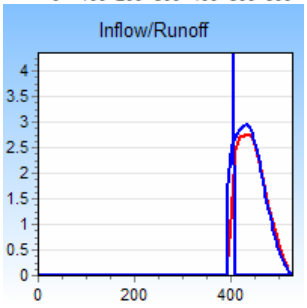


Flowrate (m^3/sec)	0.004442 *NOTE: SIRMOD value =4.442 lps*
Time-to-cutoff (mins)	450 *NOTE: SIRMOD value =450 minsOld Value 495*
Field-length (m)	635
Field-slope	0.001
Manning n	0.04
Kostiakov a	0.13452
Kostiakov k (m^3/min^a/m)	0.07844
Kostiakov fo (m^3/min/m)	0.00008 *NOTE: SIRMOD value =0.000001*
Z-required (m)	96 *NOTE: SIRMOD value =0.096 mOld Value 0.078*
Furrow top width (m)	0.4
Furrow mid width (m)	0.25
Furrow bot width (m)	0.1
Furrow max depth (m)	0.25

Field 7a 14/01/2002 Furrow 2 Irrigation no:3

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\waltons\WAL_fldAirr3fur2.cfg

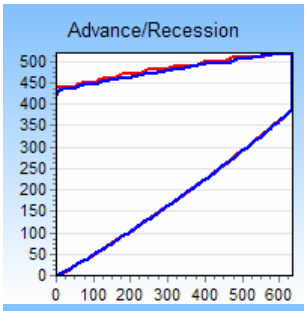


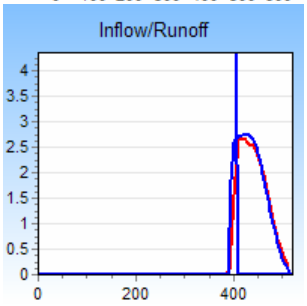


Flowrate (m^3/sec)	0.00436 *NOTE: SIRMOD value =4.36 lps*
Time-to-cutoff (mins)	400
Field-length (m)	635
Field-slope	0.001
Manning n	0.04
Kostiakov a	0.10515
Kostiakov k (m^3/min^a/m)	0.07071
Kostiakov fo (m^3/min/m)	0.00005 *NOTE: SIRMOD value =0.000001*
Z-required (m)	200 *NOTE: SIRMOD value =0.2 mOld Value 0.065*
Furrow top width (m)	0.5
Furrow mid width (m)	0.25
Furrow bot width (m)	0.1
Furrow max depth (m)	0.25

Field 7a 14/01/2002 Furrow 4 Irrigation no:3

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\waltons\WAL_fldAirr3fur4.cfg

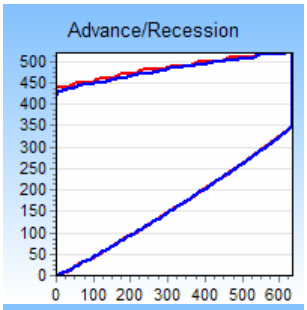


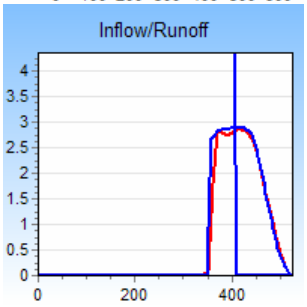


Flowrate (m^3/sec)	0.00436 *NOTE: SIRMOD value =4.36 lps*
Time-to-cutoff (mins)	400
Field-length (m)	635
Field-slope	0.001
Manning n	0.04
Kostiakov a	0
Kostiakov k (m^3/min^a/m)	0.10071
Kostiakov fo (m^3/min/m)	0.00015 *NOTE: SIRMOD value =0.000003*
Z-required (m)	200 *NOTE: SIRMOD value =0.2 mOld Value 0.065*
Furrow top width (m)	0.5
Furrow mid width (m)	0.25
Furrow bot width (m)	0.1
Furrow max depth (m)	0.25

Field 7a 14/01/2002 Furrow 6 Irrigation no:3

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\waltons\WAL_fldAirr3fur6.cfg

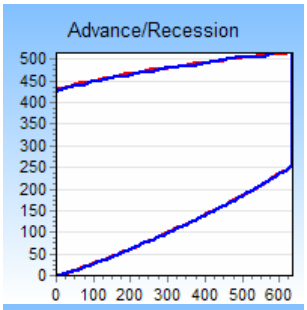


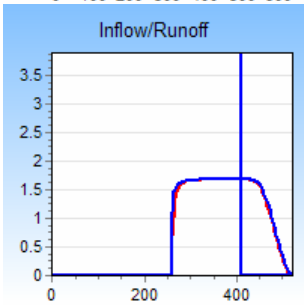


Flowrate (m^3/sec)	0.00436 *NOTE: SIRMOD value =4.36 lps*
Time-to-cutoff (mins)	400
Field-length (m)	635
Field-slope	0.001
Manning n	0.04
Kostiakov a	0
Kostiakov k (m^3/min^a/m)	0.08913
Kostiakov fo (m^3/min/m)	0.00014 *NOTE: SIRMOD value =0.000002*
Z-required (m)	191.13 *NOTE: SIRMOD value =0.19113 mOld Value 0.065*
Furrow top width (m)	0.5
Furrow mid width (m)	0.25
Furrow bot width (m)	0.1
Furrow max depth (m)	0.25

Field 7a 27/01/2002 Furrow 7 Irrigation no:4

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\waltons\WAL_fldAirr4fur7.cfg

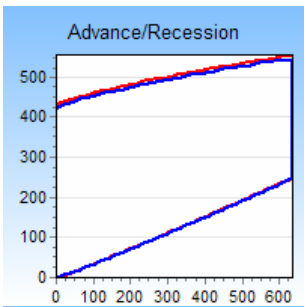


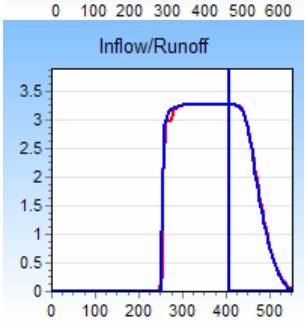


Flowrate (m^3/sec)	0.003897 *NOTE: SIRMOD value =3.897 lps*
Time-to-cutoff (mins)	400
Field-length (m)	635
Field-slope	0.001
Manning n	0.05
Kostiakov a	0
Kostiakov k (m^3/min^a/m)	0.0367
Kostiakov fo (m^3/min/m)	0.00021 *NOTE: SIRMOD value =0.000004*
Z-required (m)	200 *NOTE: SIRMOD value =0.2 mOld Value 0.085*
Furrow top width (m)	0.5
Furrow mid width (m)	0.25
Furrow bot width (m)	0.1
Furrow max depth (m)	0.25

Field 7a 27/01/2002 Furrow ave Irrigation no:4

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\waltons\WAL_fldAirr4furAVG.cfg

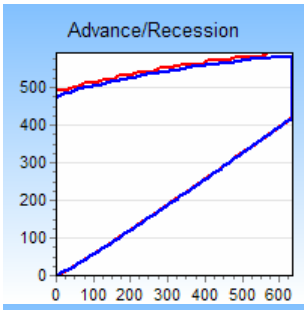


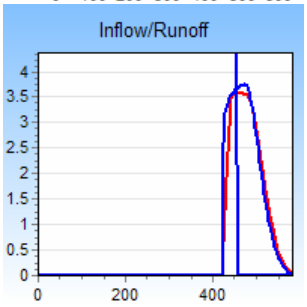


Flowrate (m^3/sec)	0.003897 *NOTE: SIRMOD value =3.897 lps*
Time-to-cutoff (mins)	400
Field-length (m)	635
Field-slope	0.001
Manning n	0.04
Kostiakov a	0.00001 *NOTE: SIRMOD value =0.00001*
Kostiakov k (m^3/min^a/m)	0.05575 *NOTE: SIRMOD value =0.05575*
Kostiakov fo (m^3/min/m)	0.00006 *NOTE: SIRMOD value =0.000001*
Z-required (m)	200 *NOTE: SIRMOD value =0.2 mOld Value 0.085*
Furrow top width (m)	0.5
Furrow mid width (m)	0.25
Furrow bot width (m)	0.1
Furrow max depth (m)	0.25

Field 7a 20/02/2002 Furrow 2 Irrigation no:6

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\waltons\WAL_fldAirr6fur2.cfg

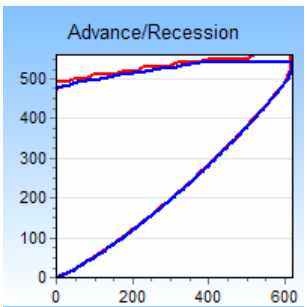


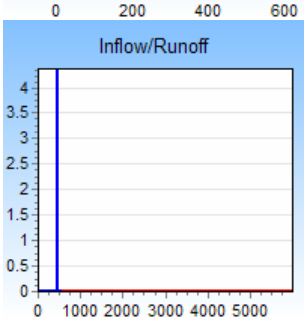


Flowrate (m^3/sec)	0.00437
Time-to-cutoff (mins)	450
Field-length (m)	635
Field-slope	0.001
Manning n	0.03
Kostiakov a	0.06954
Kostiakov k (m^3/min^a/m)	0.1037
Kostiakov fo (m^3/min/m)	0
Z-required (m)	200 *NOTE: SIRMOD value =0.2 mOld Value 0.105*
Furrow top width (m)	0.5
Furrow mid width (m)	0.25
Furrow bot width (m)	0.1
Furrow max depth (m)	0.25

Field 7a 20/02/2002 Furrow 4 Irrigation no:6

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\waltons\WAL_fldAirr6fur4.cfg

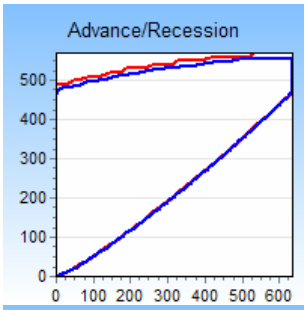


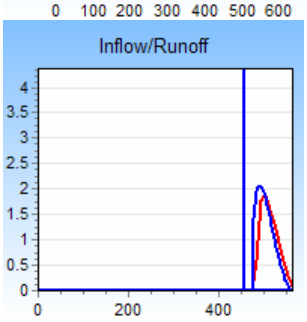


Flowrate (m^3/sec)	0.00437
Time-to-cutoff (mins)	450
Field-length (m)	635
Field-slope	0.001
Manning n	0.05
Kostiakov a	0.12549
Kostiakov k (m^3/min^a/m)	0.07019
Kostiakov fo (m^3/min/m)	0.00016 *NOTE: SIRMOD value =0.000003*
Z-required (m)	200 *NOTE: SIRMOD value =0.2 mOld Value 0.105*
Furrow top width (m)	0.5
Furrow mid width (m)	0.25
Furrow bot width (m)	0.1
Furrow max depth (m)	0.25

Field 7a 20/02/2002 Furrow 6 Irrigation no:6

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\waltons\WAL_fldAirr6fur6.cfg

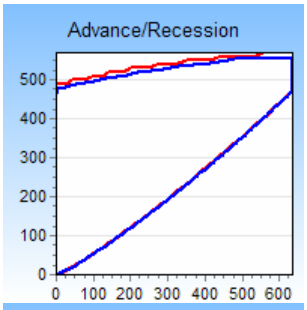


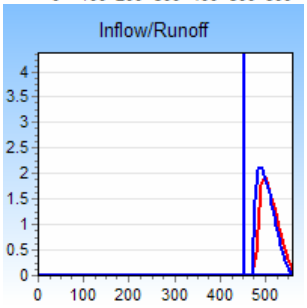


Flowrate (m^3/sec)	0.00437
Time-to-cutoff (mins)	450
Field-length (m)	635
Field-slope	0.001
Manning n	0.04
Kostiakov a	0.15281
Kostiakov k (m^3/min^a/m)	0.06884
Kostiakov fo (m^3/min/m)	0.00004 *NOTE: SIRMOD value =0.000001*
Z-required (m)	200 *NOTE: SIRMOD value =0.2 mOld Value 0.105*
Furrow top width (m)	0.5
Furrow mid width (m)	0.25
Furrow bot width (m)	0.1
Furrow max depth (m)	0.25

Field 7a 20/02/2002 Furrow 8 Irrigation no:6

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\waltons\WAL_fldAirr6fur8.cfg

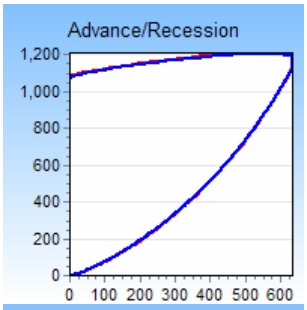


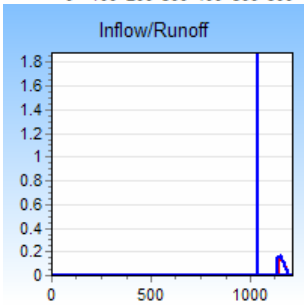


Flowrate (m^3/sec)	0.00437
Time-to-cutoff (mins)	450
Field-length (m)	635
Field-slope	0.001
Manning n	0.04
Kostiakov a	0.11616
Kostiakov k (m^3/min^a/m)	0.08008
Kostiakov fo (m^3/min/m)	0.00007 *NOTE: SIRMOD value =0.000001*
Z-required (m)	200 *NOTE: SIRMOD value =0.2 mOld Value 0.105*
Furrow top width (m)	0.5
Furrow mid width (m)	0.25
Furrow bot width (m)	0.1
Furrow max depth (m)	0.25

Field 7b 27/12/2001 Furrow 6 Irrigation no:2

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\waltons\WAL_fldBirr2fur6.cfg

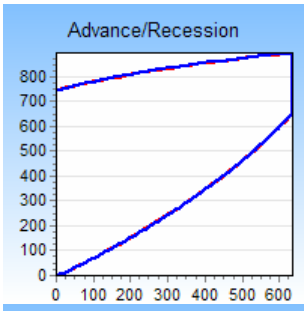


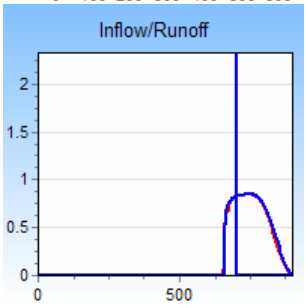


Flowrate (m^3/sec)	0.00188
Time-to-cutoff (mins)	1035 *NOTE: SIRMOD value =1035 minsOld Value 1300*
Field-length (m)	635
Field-slope	0.001
Manning n	0.1
Kostiakov a	0.233316 *NOTE: SIRMOD value =0.233316*
Kostiakov k (m^3/min^a/m)	0.02546
Kostiakov fo (m^3/min/m)	0.00009 *NOTE: SIRMOD value =0.000002*
Z-required (m)	122 *NOTE: SIRMOD value =0.122 mOld Value 0.065*
Furrow top width (m)	0.4
Furrow mid width (m)	0.25
Furrow bot width (m)	0.1
Furrow max depth (m)	0.25

Field 7b 14/01/2002 Furrow 2 Irrigation no:3

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\waltons\WAL_fldBirr3fur2.cfg

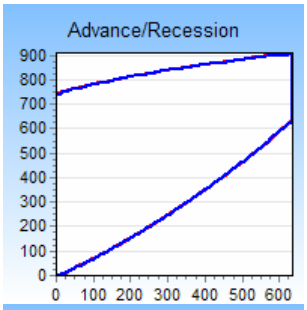


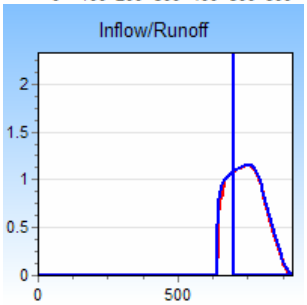


Flowrate (m^3/sec)	0.00233
Time-to-cutoff (mins)	694
Field-length (m)	635
Field-slope	0.001
Manning n	0.1
Kostiakov a	0
Kostiakov k (m^3/min^a/m)	0.06171
Kostiakov fo (m^3/min/m)	0.00014 *NOTE: SIRMOD value =0.000002*
Z-required (m)	200 *NOTE: SIRMOD value =0.2 mOld Value 0.064*
Furrow top width (m)	0.5
Furrow mid width (m)	0.25
Furrow bot width (m)	0.1
Furrow max depth (m)	0.25

Field 7b 14/01/2002 Furrow 4 Irrigation no:3

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\waltons\WAL_fldBirr3fur4.cfg

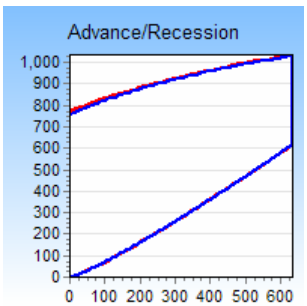


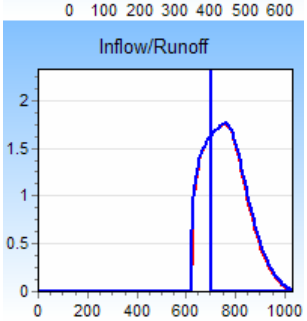


Flowrate (m^3/sec)	0.00233
Time-to-cutoff (mins)	694
Field-length (m)	635
Field-slope	0.001
Manning n	0.09
Kostiakov a	0.08275
Kostiakov k (m^3/min^a/m)	0.04904
Kostiakov fo (m^3/min/m)	0.00009 *NOTE: SIRMOD value =0.000002*
Z-required (m)	200 *NOTE: SIRMOD value =0.2 mOld Value 0.064*
Furrow top width (m)	0.5
Furrow mid width (m)	0.25
Furrow bot width (m)	0.1
Furrow max depth (m)	0.25

Field 7b 14/01/2002 Furrow 6 Irrigation no:3

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\waltons\WAL_fldBirr3fur6.cfg

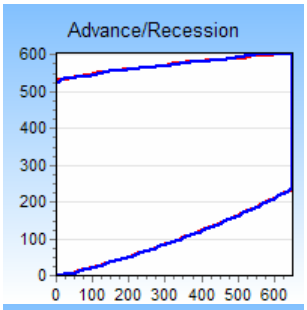


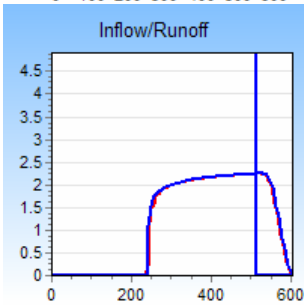


Flowrate (m^3/sec)	0.00233
Time-to-cutoff (mins)	694
Field-length (m)	635
Field-slope	0.001
Manning n	0.1
Kostiakov a	0.16665
Kostiakov k (m^3/min^a/m)	0.0391
Kostiakov fo (m^3/min/m)	0
Z-required (m)	128 *NOTE: SIRMOD value =0.128 mOld Value 0.064*
Furrow top width (m)	0.5
Furrow mid width (m)	0.25
Furrow bot width (m)	0.1
Furrow max depth (m)	0.25

Field 7b 14/01/2002Furrow 8 Irrigation no:3

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\waltons\WAL_fldBirr3fur8.cfg

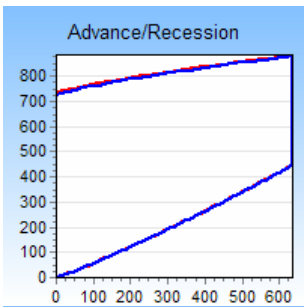


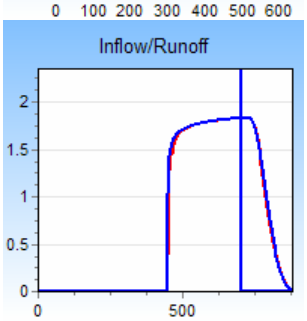


Flowrate (m^3/sec)	0.00491 *NOTE: SIRMOD value =4.91 lps*
Time-to-cutoff (mins)	505 *NOTE: SIRMOD value =505 minsOld Value 694*
Field-length (m)	650 *NOTE: SIRMOD value =650 mOld Value 635*
Field-slope	0.001
Manning n	0.04
Kostiakov a	0.10758 *NOTE: SIRMOD value =0.10758*
Kostiakov k (m^3/min^a/m)	0.02967 *NOTE: SIRMOD value =0.02967*
Kostiakov fo (m^3/min/m)	0.00023 *NOTE: SIRMOD value =0.000004*
Z-required (m)	200 *NOTE: SIRMOD value =0.2 mOld Value 0.064*
Furrow top width (m)	0.5
Furrow mid width (m)	0.25
Furrow bot width (m)	0.1
Furrow max depth (m)	0.25

Field 7b 27/01/2002 Furrow 2 Irrigation no:4

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\waltons\WAL_fldBirr4fur2.cfg

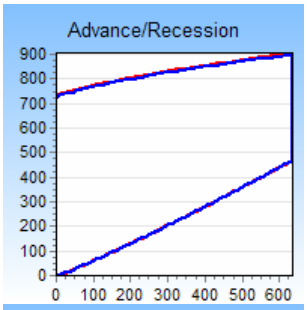


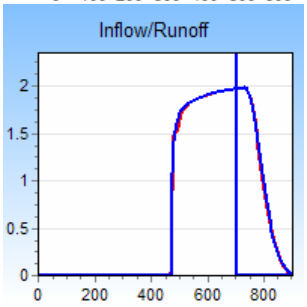


Flowrate (m^3/sec)	0.0023507 *NOTE: SIRMOD value =2.3507 lps*
Time-to-cutoff (mins)	695
Field-length (m)	635
Field-slope	0.001
Manning n	0.05
Kostiakov a	0.05407 *NOTE: SIRMOD value =0.05407*
Kostiakov k (m^3/min^a/m)	0.05122 *NOTE: SIRMOD value =0.05122*
Kostiakov fo (m^3/min/m)	0.00004 *NOTE: SIRMOD value =0.000001*
Z-required (m)	200 *NOTE: SIRMOD value =0.2 mOld Value 0.085*
Furrow top width (m)	0.5
Furrow mid width (m)	0.25
Furrow bot width (m)	0.1
Furrow max depth (m)	0.25

Field 7b 27/01/2002 Furrow 4 Irrigation no:4

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\waltons\WAL_fldBirr4fur4.cfg

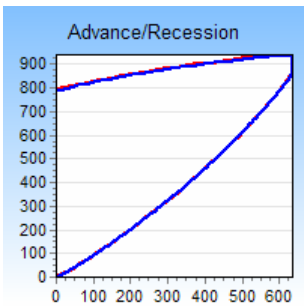


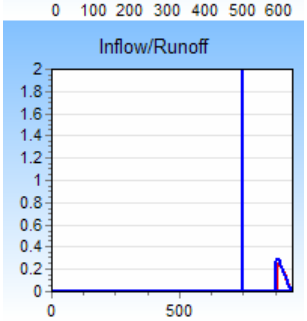


Flowrate (m^3/sec)	0.0023507 *NOTE: SIRMOD value =2.3507 lps*
Time-to-cutoff (mins)	695
Field-length (m)	635
Field-slope	0.001
Manning n	0.05
Kostiakov a	0.07832 *NOTE: SIRMOD value =0.07832*
Kostiakov k (m^3/min^a/m)	0.05099 *NOTE: SIRMOD value =0.05099*
Kostiakov fo (m^3/min/m)	0.00002 *NOTE: SIRMOD value =0*
Z-required (m)	178 *NOTE: SIRMOD value =0.178 mOld Value 0.085*
Furrow top width (m)	0.5
Furrow mid width (m)	0.25
Furrow bot width (m)	0.1
Furrow max depth (m)	0.25

Field 7b 20/02/2002 Furrow 6 Irrigation no:6

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\waltons\WAL_fldBirr6fur6.cfg

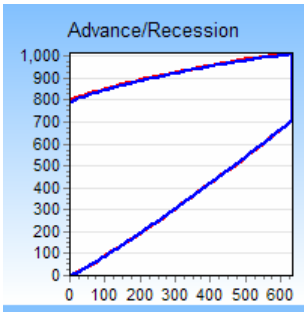


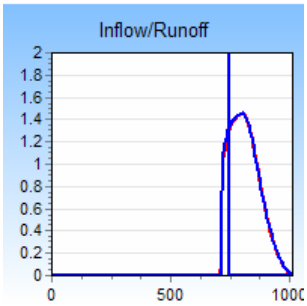


Flowrate (m^3/sec)	0.002
Time-to-cutoff (mins)	740 *NOTE: SIRMOD value =740 minsOld Value 750*
Field-length (m)	635
Field-slope	0.001
Manning n	0.09
Kostiakov a	0.03777
Kostiakov k (m^3/min^a/m)	0.06997
Kostiakov fo (m^3/min/m)	0.0001 *NOTE: SIRMOD value =0.000002*
Z-required (m)	200 *NOTE: SIRMOD value =0.2 mOld Value 0.097*
Furrow top width (m)	0.5
Furrow mid width (m)	0.25
Furrow bot width (m)	0.1
Furrow max depth (m)	0.25

Field 7b 20/02/2002 Furrow 8 Irrigation no:6

SIRMOD Datafile: D:\PROJECTS\FIDO\Bin2\Data\SIRMOD_files\waltons\WAL_fldBirr6fur8.cfg





Flowrate (m^3/sec)	0.002
Time-to-cutoff (mins)	740 *NOTE: SIRMOD value =740 minsOld Value 750*
Field-length (m)	635
Field-slope	0.001
Manning n	0.08
Kostiakov a	0.09108
Kostiakov k (m^3/min^a/m)	0.05878
Kostiakov fo (m^3/min/m)	0.00002 *NOTE: SIRMOD value =0*
Z-required (m)	200 *NOTE: SIRMOD value =0.2 mOld Value 0.097*
Furrow top width (m)	0.5
Furrow mid width (m)	0.25
Furrow bot width (m)	0.1
Furrow max depth (m)	0.25

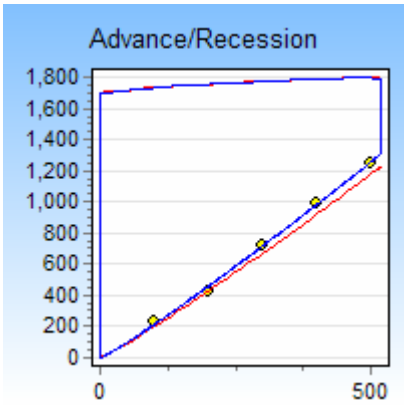
Created by David McClymont, NCEA on the 2006-08-24
mcclymon@usq.edu.au using FIDO v1.01 (beta)

Appendix 4.1 Calibrated advance curves

Calibration Summary: C_Turner's Property

Red lines respresent Simulations using *INFILT* calibrated infiltration parameters. Blue lines represent simulations using *"Hydrodynamic Method"* calibrated infiltrated parameters

Field 19 7/10/2000 Furrow 1 Irrigation no:1

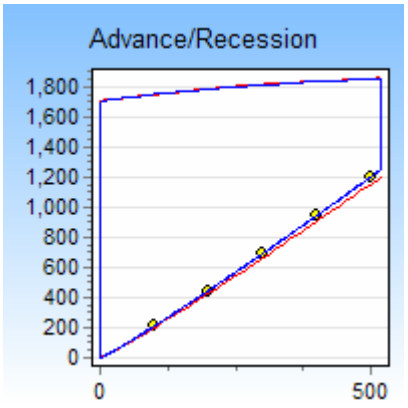


Measured Advance

x(m)	t(min)
100	234
200	426
300	724
400	988
500	1246

Flowrate (m^3/sec)	0.00194167 *NOTE: INFILT value =1.941667 l/sec*
Time-to-cutoff (mins)	1690 *NOTE: SIRMOD value =1690 minsOld Value 1689*
Field-length (m)	520
Field-slope	0.00151
Manning n	0.03
Kostiakov a	0.09162 **Previous: 0.10155**
Kostiakov k (m^3/min^a/m)	0.15781662 **Previous: 0.13916**
Kostiakov fo (m^3/min/m)	0 **Previous: 0**
Z-required (m)	280 *NOTE: SIRMOD value =0.28 mOld Value 0.111*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 19 7/10/2000 Furrow ave Irrigation no:1

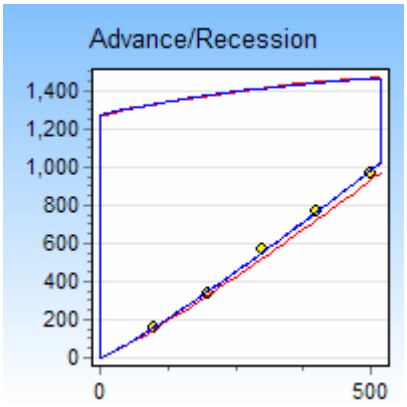


Measured Advance

x(m)	t(min)
100	214
200	438
300	689
400	948
500	1194

Flowrate (m^3/sec)	0.00194167 *NOTE: INFILT value =1.941667 l/sec*
Time-to-cutoff (mins)	1690 *NOTE: SIRMOD value =1690 minsOld Value 1689*
Field-length (m)	520
Field-slope	0.00151
Manning n	0.05
Kostiakov a	0.07999 **Previous: 0.08592**
Kostiakov k (m^3/min^a/m)	0.15949077 **Previous: 0.1469**
Kostiakov fo (m^3/min/m)	0 **Previous: 0**
Z-required (m)	222 *NOTE: SIRMOD value =0.222 mOld Value 0.111*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 19 31/12/2000 Furrow ave Irrigation no:3

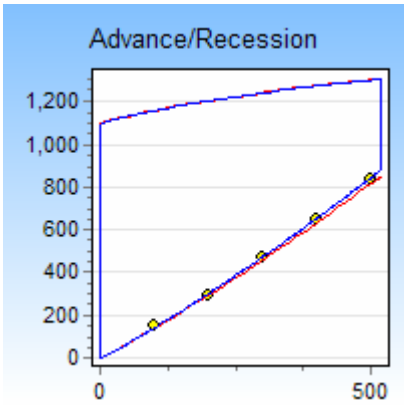


Measured Advance

x(m)	t(min)
100	156
200	338
300	562
400	770
500	969

Flowrate (m^3/sec)	0.00148667 *NOTE: SIRMOD value =1.486 lps NOTE: INFILT value =1.486667 l/sec*
Time-to-cutoff (mins)	1248
Field-length (m)	520
Field-slope	0.00151
Manning n	0.08
Kostiakov a	0.125755 **Previous: 0.13823**
Kostiakov k (m^3/min^a/m)	0.07233696 **Previous: 0.06337**
Kostiakov fo (m^3/min/m)	0 **Previous: 0**
Z-required (m)	144 *NOTE: SIRMOD value =0.144 mOld Value 0.067*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 19 12/01/2001 Furrow ave Irrigation no:4

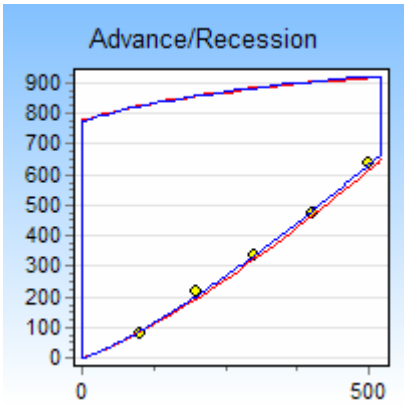


Measured Advance

x(m)	t(min)
100	146
200	293
300	472
400	651
500	839

Flowrate (m^3/sec)	0.00158
Time-to-cutoff (mins)	1070
Field-length (m)	520
Field-slope	0.00151
Manning n	0.1
Kostiakov a	0.07201 **Previous: 0.07352**
Kostiakov k (m^3/min^a/m)	0.07994569 **Previous: 0.07585**
Kostiakov fo (m^3/min/m)	0.0000294 **Previous: 0.00003**
Z-required (m)	130 *NOTE: SIRMOD value =0.13 mOld Value 0.068*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 19 24/01/2001 Furrow ave Irrigation no:5

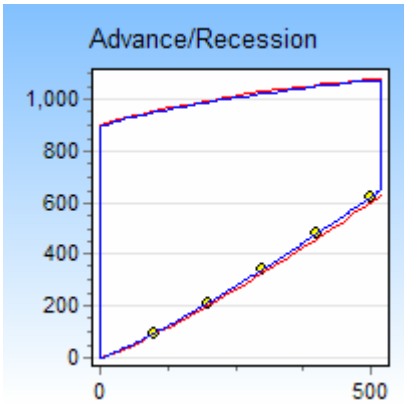


Measured Advance

x(m)	t(min)
100	82
200	214
300	336
400	472
500	640

Flowrate (m^3/sec)	0.00203
Time-to-cutoff (mins)	755
Field-length (m)	520
Field-slope	0.00151
Manning n	0.08
Kostiakov a	0.225925 Previous: 0.2198
Kostiakov k (m^3/min^a/m)	0.03588277 Previous: 0.03456
Kostiakov fo (m^3/min/m)	0.0000038 Previous: 0.00002
Z-required (m)	112 *NOTE: SIRMOD value =0.112 mOld Value 0.056*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 19 12/02/2001 Furrow ave Irrigation no:6

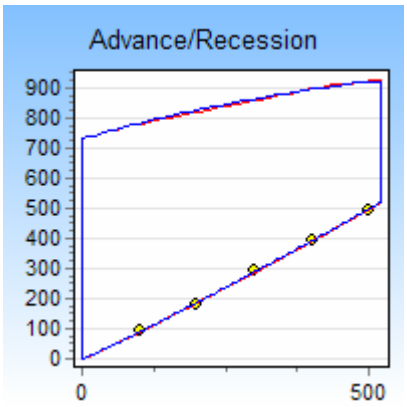


Measured Advance

x(m)	t(min)
100	92
200	212
300	343
400	480
500	617

Flowrate (m^3/sec)	0.002
Time-to-cutoff (mins)	875 *NOTE: SIRMOD value =875 minsOld Value 445*
Field-length (m)	520
Field-slope	0.00151
Manning n	0.08
Kostiakov a	0.17031 Previous: 0.18432
Kostiakov k (m^3/min^a/m)	0.04780402 Previous: 0.04261
Kostiakov fo (m^3/min/m)	0 Previous: 0
Z-required (m)	122 *NOTE: SIRMOD value =0.122 mOld Value 0.061*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 19 26/02/2001 Furrow ave Irrigation no:7

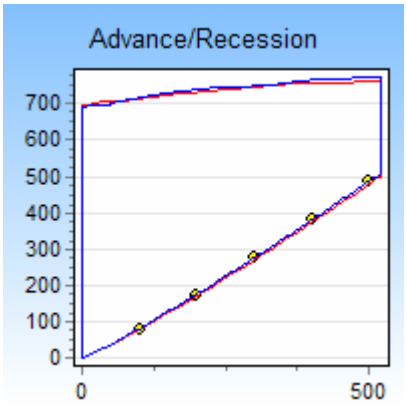


Measured Advance

x(m)	t(min)
100	90
200	180
300	291
400	393
500	493

Flowrate (m^3/sec)	0.0026
Time-to-cutoff (mins)	705
Field-length (m)	520
Field-slope	0.00151
Manning n	0.07
Kostiakov a	0.07068 Previous: 0.082
Kostiakov k (m^3/min^a/m)	0.0868203 Previous: 0.08137
Kostiakov fo (m^3/min/m)	0 Previous: 0
Z-required (m)	118 *NOTE: SIRMOD value =0.118 mOld Value 0.059*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 20 9/10/2000 Furrow ave Irrigation no:1

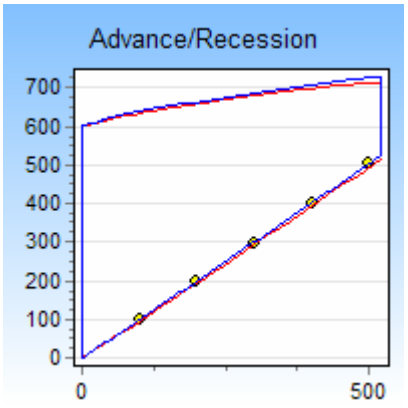


Measured Advance

x(m)	t(min)
100	78
200	173
300	278
400	381
500	485

Flowrate (m^3/sec)	0.00505
Time-to-cutoff (mins)	680
Field-length (m)	520
Field-slope	0.00151
Manning n	0.03
Kostiakov a	0.112265 Previous: 0.12189
Kostiakov k (m^3/min^a/m)	0.14946592 Previous: 0.13952
Kostiakov fo (m^3/min/m)	0 Previous: 0
Z-required (m)	266 *NOTE: SIRMOD value =0.266 mOld Value 0.134*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 20 10/12/2000 Furrow ave Irrigation no:2

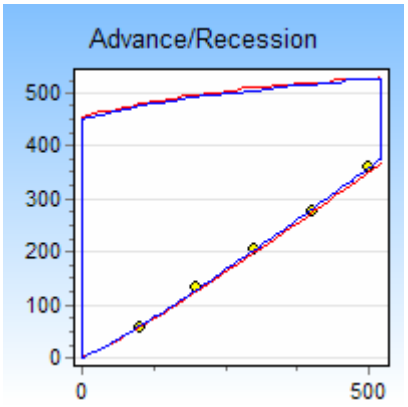


Measured Advance

x(m)	t(min)
100	98
200	197
300	297
400	400
500	502

Flowrate (m^3/sec)	0.00511
Time-to-cutoff (mins)	594
Field-length (m)	520
Field-slope	0.00151
Manning n	0.03
Kostiakov a	0.00848 Previous: 0.01991
Kostiakov k (m^3/min^a/m)	0.27023988 Previous: 0.24978
Kostiakov fo (m^3/min/m)	0 Previous: 0
Z-required (m)	286.608 *NOTE: SIRMOD value =0.286608 mOld Value 0.098*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 20 1/01/2001 Furrow ave Irrigation no:3

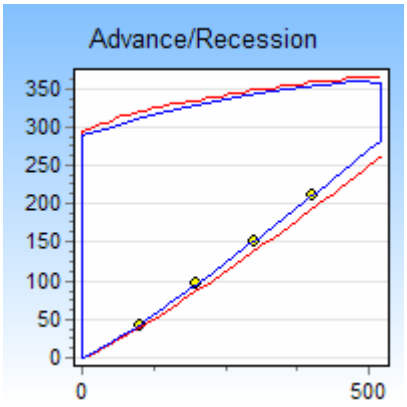


Measured Advance

x(m)	t(min)
100	56
200	133
300	203
400	277
500	360

Flowrate (m^3/sec)	0.0051
Time-to-cutoff (mins)	445 *NOTE: SIRMOD value =445 minsOld Value 875*
Field-length (m)	520
Field-slope	0.00151
Manning n	0.03
Kostiakov a	0.11379 Previous: 0.12373
Kostiakov k (m^3/min^a/m)	0.11054496 Previous: 0.10257
Kostiakov fo (m^3/min/m)	0 Previous: 0
Z-required (m)	162 *NOTE: SIRMOD value =0.162 mOld Value 0.081*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 20 29/01/2001 Furrow ave Irrigation no:5

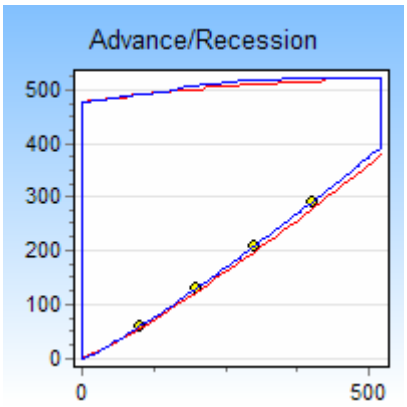


Measured Advance

x(m)	t(min)
100	43
200	96
300	152
400	210

Flowrate (m^3/sec)	0.00395 *NOTE: SIRMOD value =3.95 lps NOTE: INFILT value =3.95 l/sec*
Time-to-cutoff (mins)	285
Field-length (m)	520
Field-slope	0.00151
Manning n	0.03
Kostiakov a	0.132125 Previous: 0.1606
Kostiakov k (m^3/min^a/m)	0.05786585 Previous: 0.0464
Kostiakov fo (m^3/min/m)	0 Previous: 0
Z-required (m)	140 *NOTE: SIRMOD value =0.14 mOld Value 0.072*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 20 13/02/2001 Furrow ave Irrigation no:6

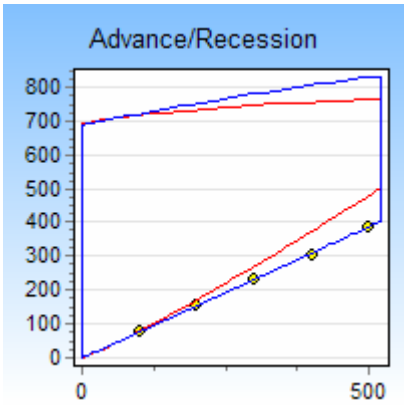


Measured Advance

x(m)	t(min)
100	60
200	130
300	209
400	290

Flowrate (m^3/sec)	0.00548
Time-to-cutoff (mins)	468
Field-length (m)	520
Field-slope	0.00151
Manning n	0.02
Kostiakov a	0.138605 Previous: 0.13235
Kostiakov k (m^3/min^a/m)	0.11260512 Previous: 0.10811
Kostiakov fo (m^3/min/m)	0 Previous: 0.00004
Z-required (m)	200 *NOTE: SIRMOD value =0.2 mOld Value 0.1*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Field 20 26/02/2001 Furrow ave Irrigation no:7



Measured Advance

x(m)	t(min)
100	75
200	153
300	229
400	302
500	388

Flowrate (m^3/sec)	0.00476 *NOTE: SIRMOD value =5.05 lps*
Time-to-cutoff (mins)	680 *NOTE: SIRMOD value =680 minsOld Value 425*
Field-length (m)	520
Field-slope	0.00151
Manning n	0.03
Kostiakov a	0.00602 Previous: 0.12189
Kostiakov k (m^3/min^a/m)	0.19065816 Previous: 0.13952
Kostiakov fo (m^3/min/m)	0 Previous: 0
Z-required (m)	266 *NOTE: SIRMOD value =0.266 mOld Value 0.095*
Furrow top width (m)	0.72
Furrow mid width (m)	0.48
Furrow bot width (m)	0.3
Furrow max depth (m)	0.2

Created by David McClymont, NCEA on the 2006-08-24 mcclymon@usq.edu.au using FIDO v1.01 (beta)

Appendix 5.1 Response-surface generation for different user-defined weightings of the objective-function

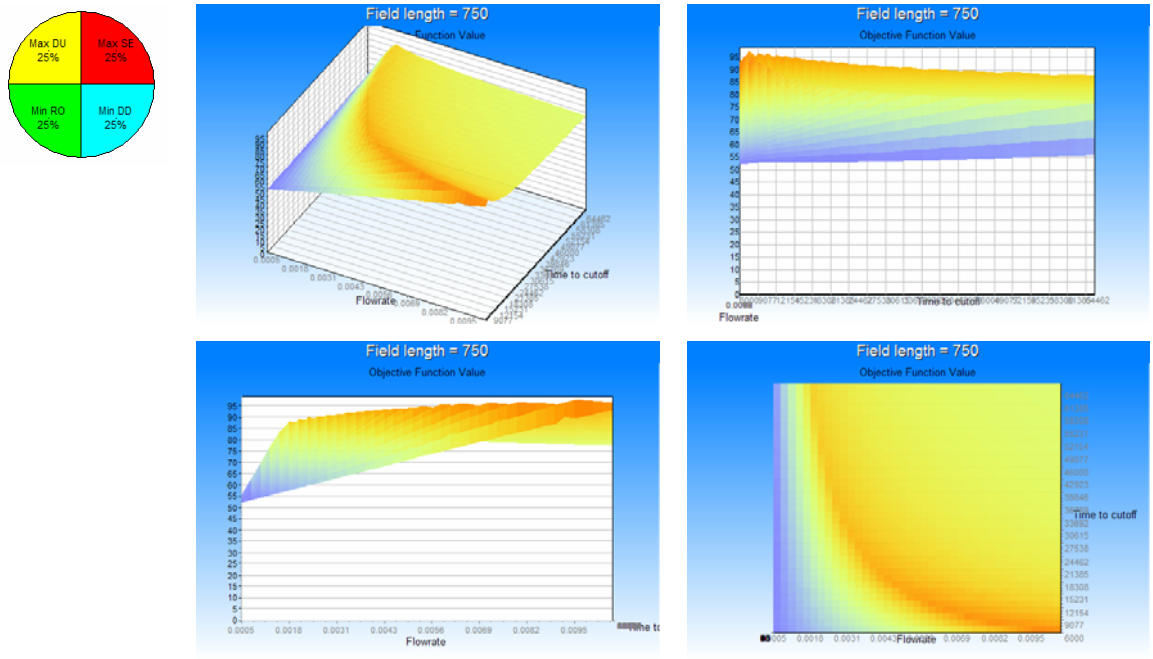


Figure A5.1.1: Response-surface for equal weightings of the objective-function components.

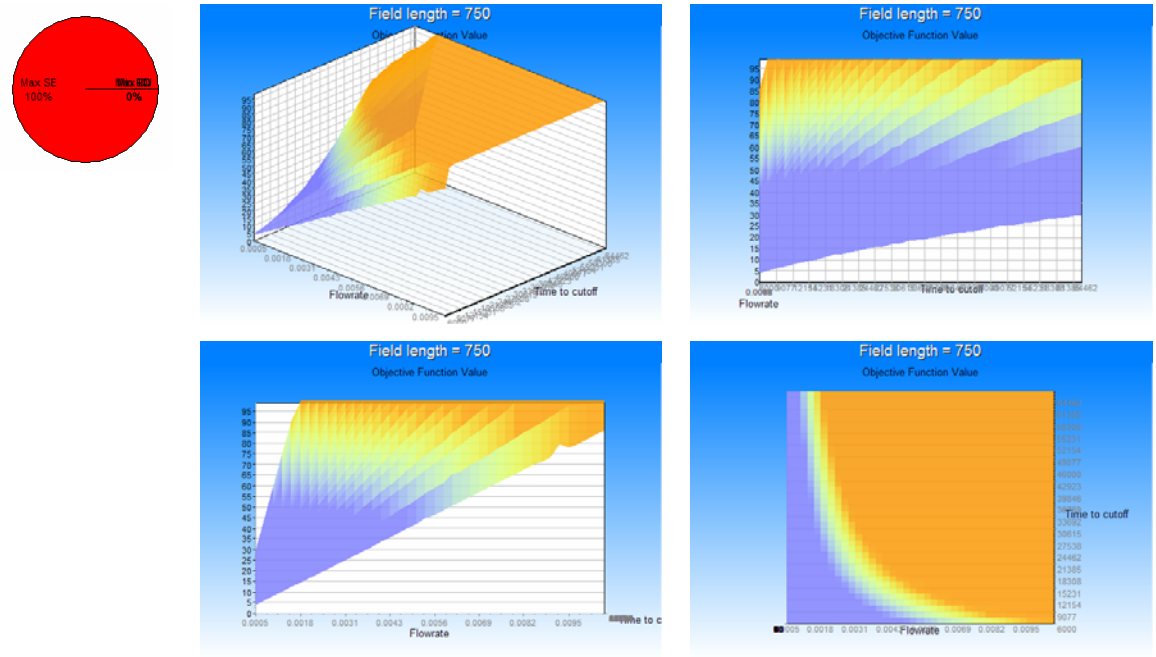


Figure A5.1.2: Response-surface for maximising storage efficiency.

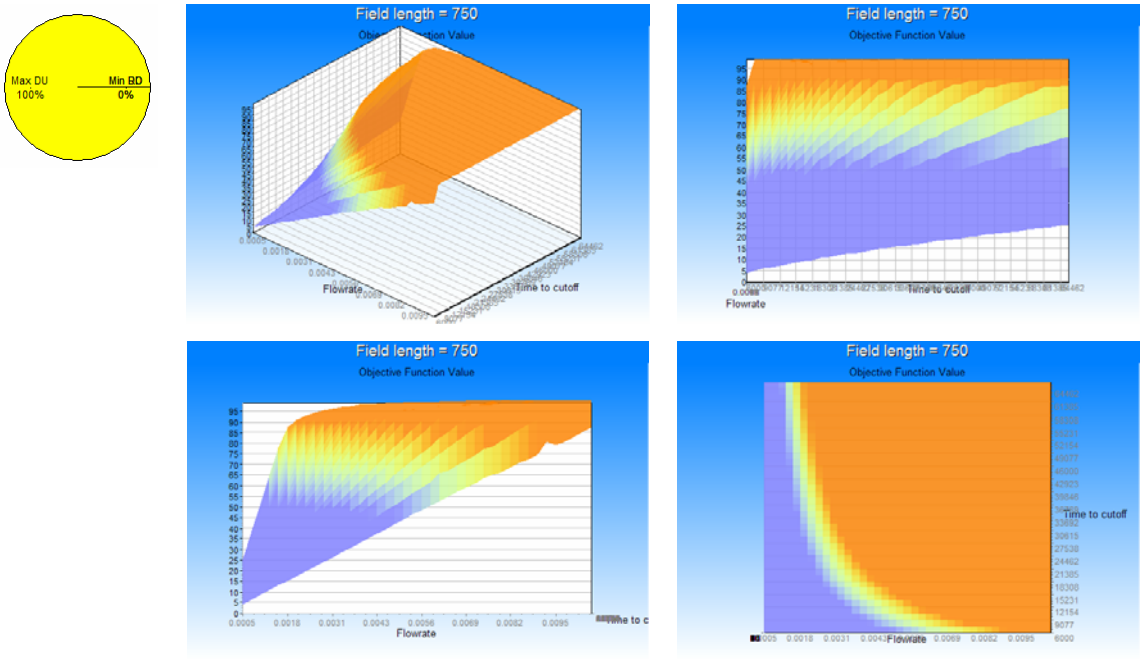


Figure A5.1.3: Response-surface for maximising application uniformity.

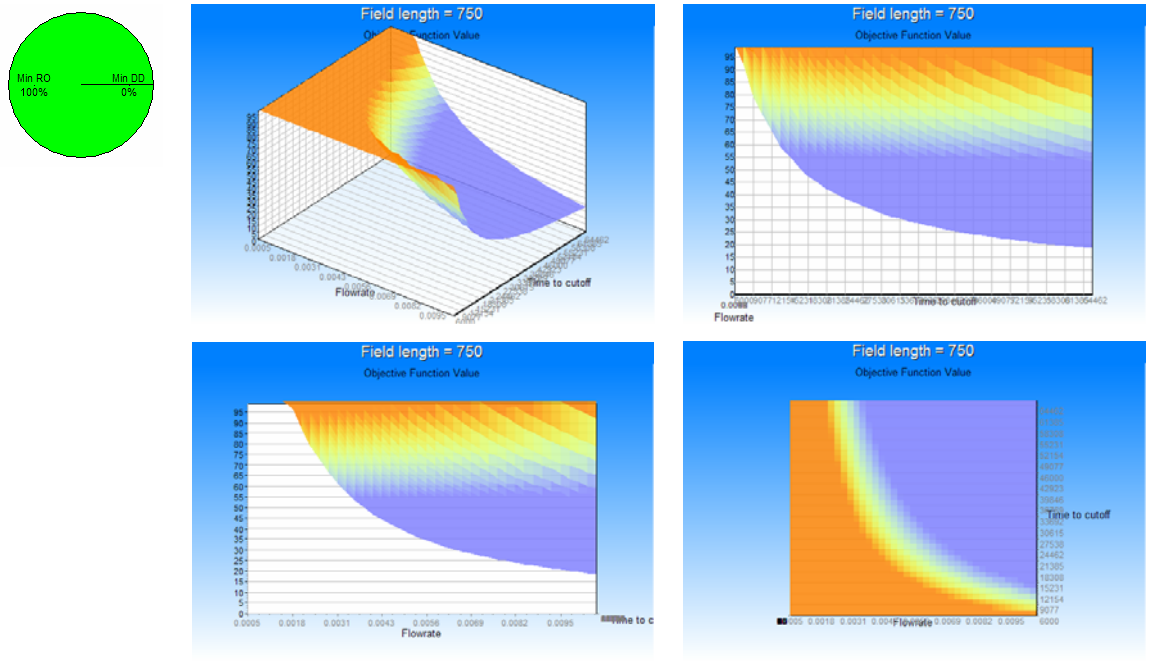


Figure A5.1.4: Response-surface for minimising runoff.

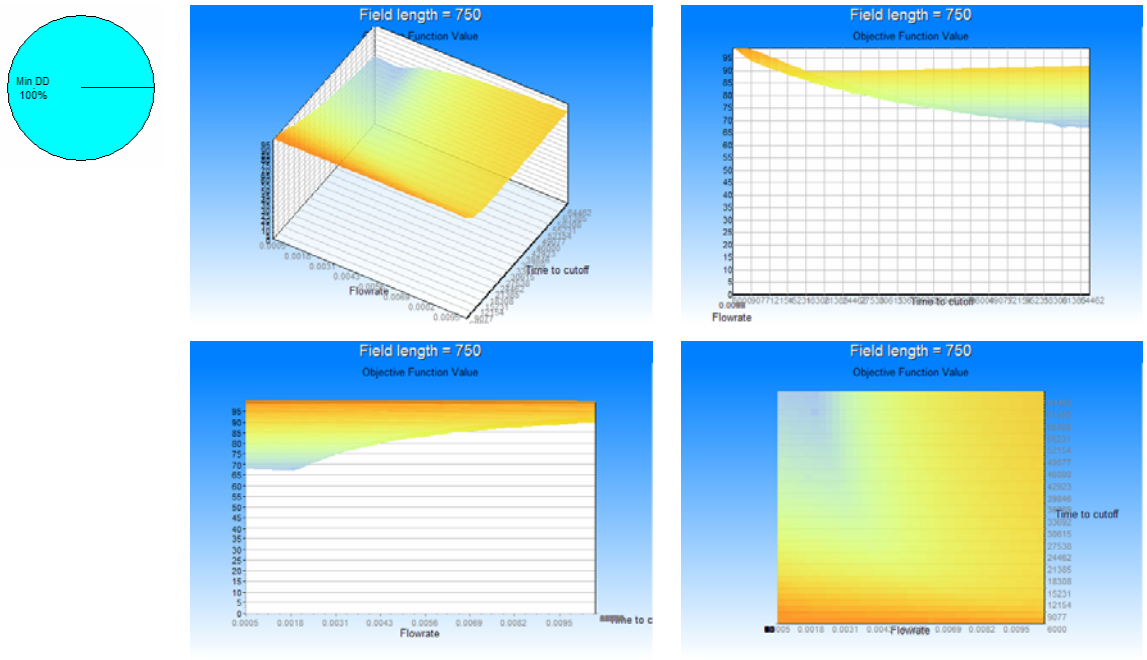


Figure A5.1.5: Response-surface for minimising deep drainage.

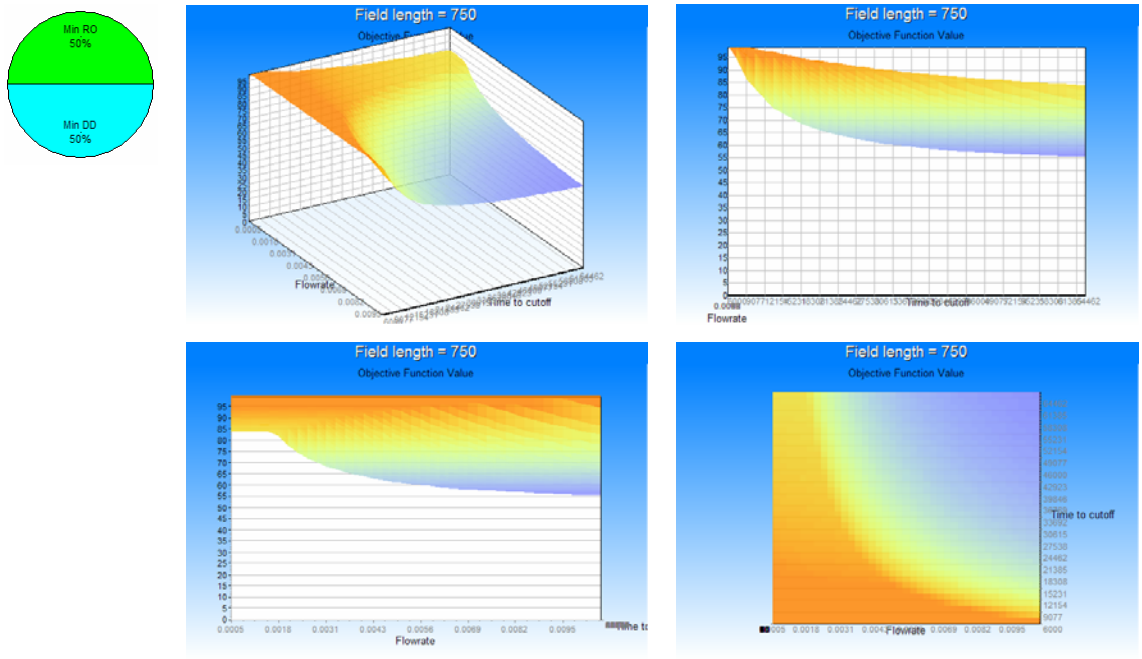


Figure A5.1.6: Response-surface for maximising storage efficiency.

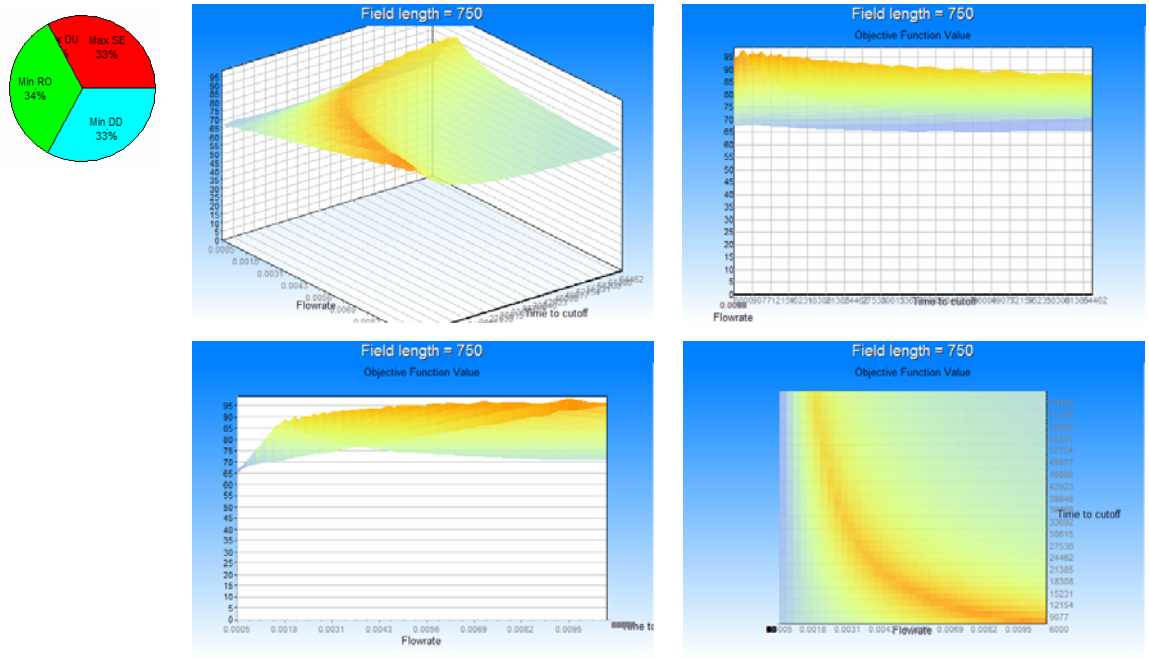


Figure A5.1.7: Response-surface for ignoring uniformity.

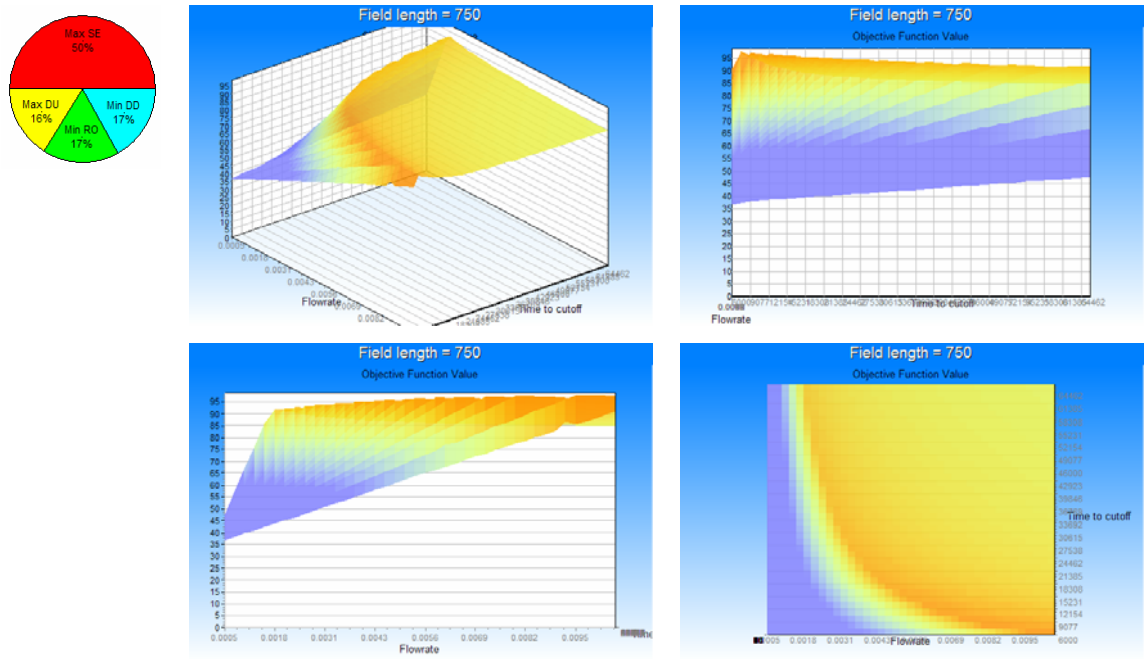


Figure A5.1.8: Response-surface for emphasising maximise storage efficiency.

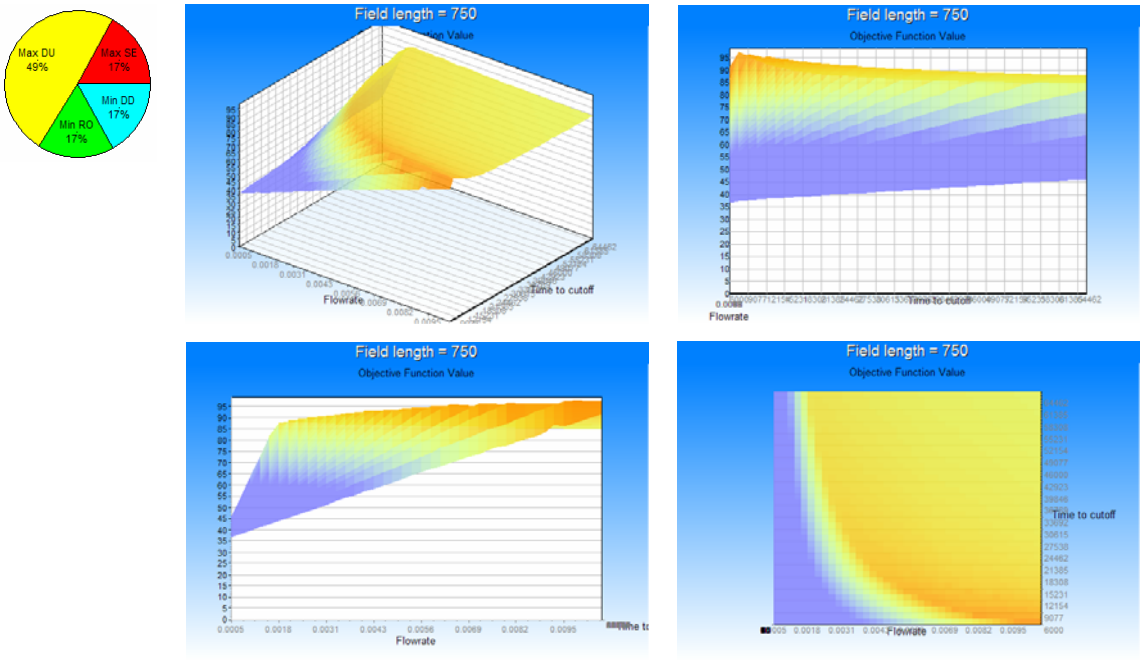


Figure A5.1.9: Response-surface for emphasising maximise application uniformity.

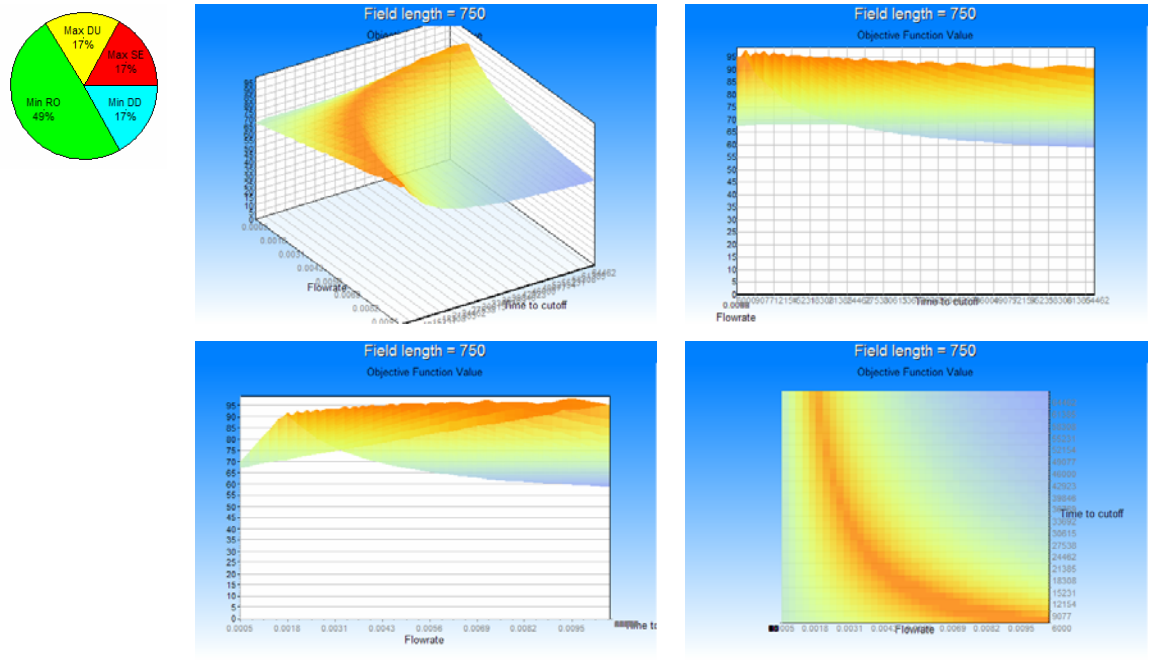


Figure A5.1.10: Response-surface for emphasising minimise runoff.

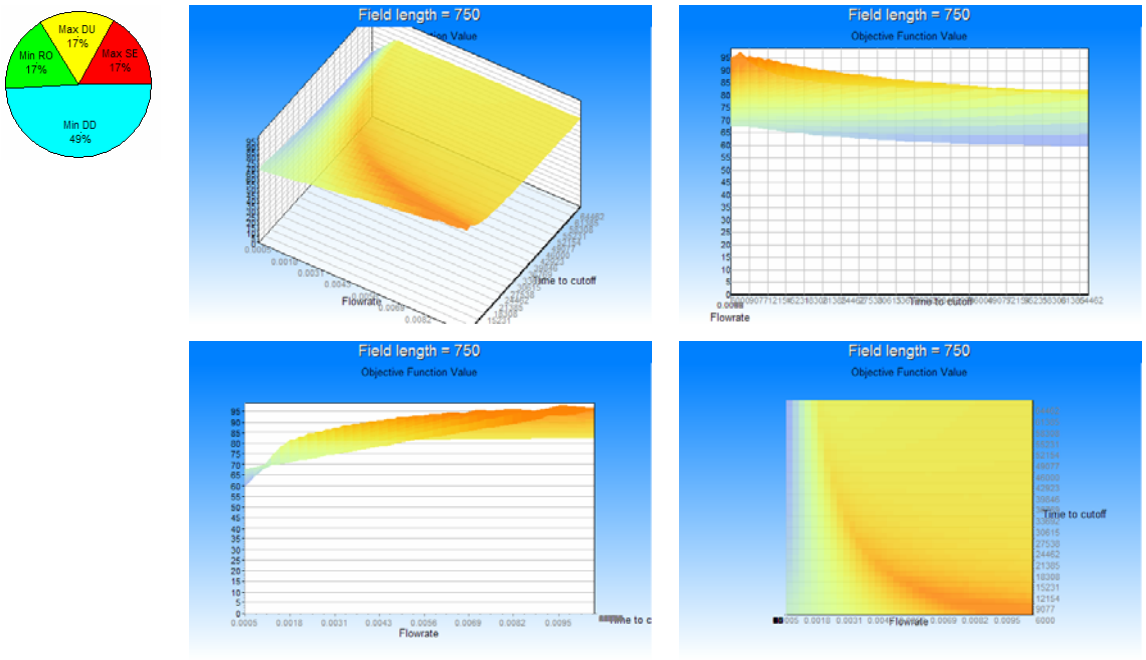


Figure A5.1.11: Response-surface for emphasising minimise drainage.

Appendix 5.2 FIDO Optimisation Trial Results

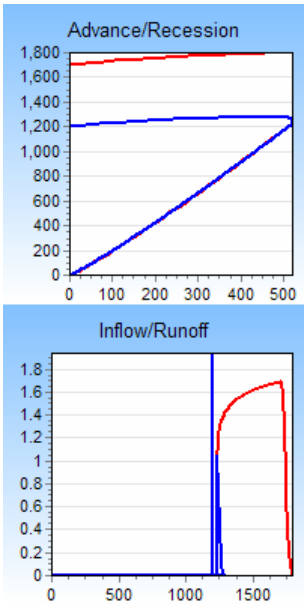
Note: Red lines in charts represent SIRMOD output, while blue lines are optimised outputs.

Optimisation Output: C_Turner's Property

Sirmod vs Optimised Comparison

Field 19, 7/10/2000: Furrow 1 Irrigation no:1

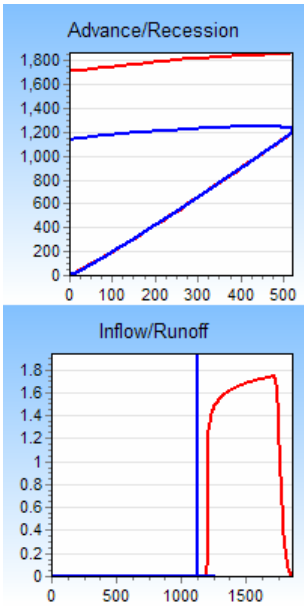
Optimisation 1. Run by at 09::26 17/11/2006



Optimisation Parameter	Measured	Optimised	Comments
Flowrate	0.00194	0.00194	
Time-to-cutoff	1690	1198	
Performance Measure	**Measured**	**Optimised**	**Comments**
Application Efficiency	73.3	98.5	
Storage Efficiency	99.3	94.5	
Application Uniformity	96	92.7	
Applied Volume	196982.6	139447.2	
Runoff Volume	49287.7	1623.9	
Stored Volume	144520.5	137534.3	
Drainage Volume	3354.6	442.2	

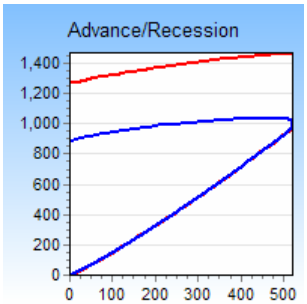
Field 19, 7/10/2000: Furrow ave Irrigation no:1

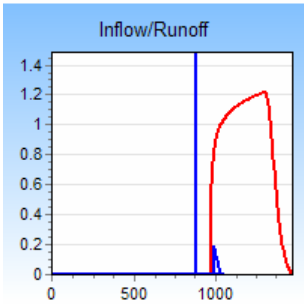
Optimisation 1. Run by at 09::26 17/11/2006



Optimisation Parameter	Measured	Optimised	Comments
Flowrate	0.00194	0.00194	
Time-to-cutoff	1690	1126	
Performance Measure	**Measured**	**Optimised**	**Comments**
Application Efficiency	58.5	87.7	
Storage Efficiency	100	99.7	
Application Uniformity	97	93.8	
Applied Volume	196941.9	131066.4	
Runoff Volume	56740.2	0	
Stored Volume	115440	115133.9	
Drainage Volume	24933.9	16066.6	

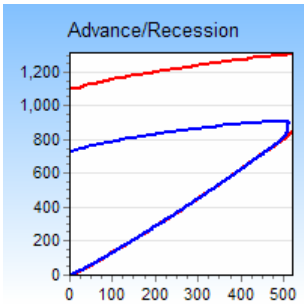
Field 19, **31/12/2000**: Furrow ave Irrigation no:3
Optimisation 1. Run by at 09::26 17/11/2006

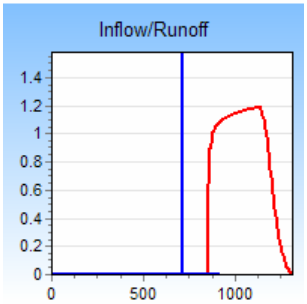




Optimisation Parameter	Measured	Optimised	Comments
Flowrate	0.001486	0.001486	
Time-to-cutoff	1248	870	
Performance Measure	**Measured**	**Optimised**	**Comments**
Application Efficiency	67.1	94.2	
Storage Efficiency	100	97.7	
Application Uniformity	95.6	91.8	
Applied Volume	111510.7	77569.2	
Runoff Volume	26965.2	262.7	
Stored Volume	74880	73189.4	
Drainage Volume	9715.8	4201.5	

Field 19, **12/01/2001**: Furrow ave Irrigation no:4
Optimisation 1. Run by at 09::26 17/11/2006

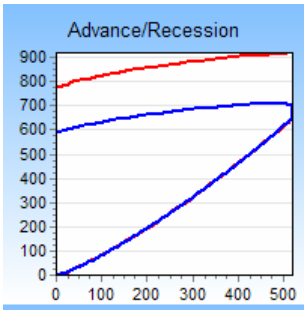


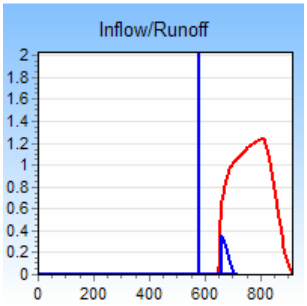


Optimisation Parameter	Measured	Optimised	Comments
Flowrate	0.00158	0.00158	
Time-to-cutoff	1070	710	
Performance Measure	**Measured**	**Optimised**	**Comments**
Application Efficiency	66.5	95.8	
Storage Efficiency	100	95.5	
Application Uniformity	93.2	89.3	
Applied Volume	101480.9	67308	
Runoff Volume	24265.4	0	
Stored Volume	67600	64581.4	
Drainage Volume	9734.9	2811.8	

Field 19, 24/01/2001: Furrow ave Irrigation no:5

Optimisation 1. Run by at 09::26 17/11/2006

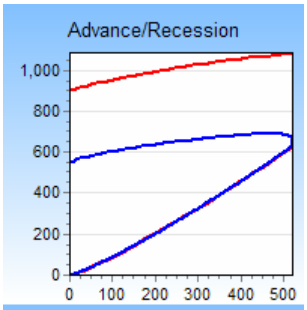


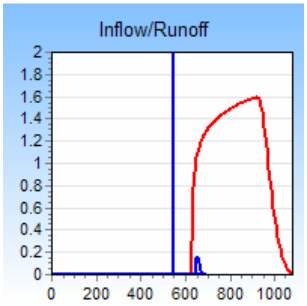


Optimisation Parameter	Measured	Optimised	Comments
Flowrate	0.00203	0.00203	
Time-to-cutoff	755	573.8	
Performance Measure	**Measured**	**Optimised**	**Comments**
Application Efficiency	63.1	82	
Storage Efficiency	100	98.5	
Application Uniformity	91.4	87.4	
Applied Volume	92151.8	69888.8	
Runoff Volume	13932.9	598	
Stored Volume	58240	57395.1	
Drainage Volume	20094.6	11960	

Field 19, 12/02/2001: Furrow ave Irrigation no:6

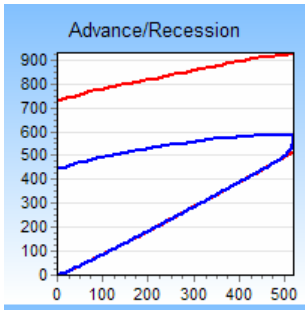
Optimisation 1. Run by at 09::26 17/11/2006

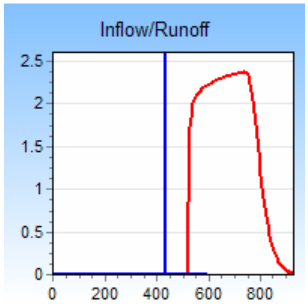




Optimisation Parameter	Measured	Optimised	Comments
Flowrate	0.002	0.002	
Time-to-cutoff	875	538.4	
Performance Measure	**Measured**	**Optimised**	**Comments**
Application Efficiency	60.1	94.7	
Storage Efficiency	100	96.6	
Application Uniformity	95.8	90.7	
Applied Volume	105258.3	64608	
Runoff Volume	31066.1	180.9	
Stored Volume	63440	61277.3	
Drainage Volume	10883.3	3216.1	

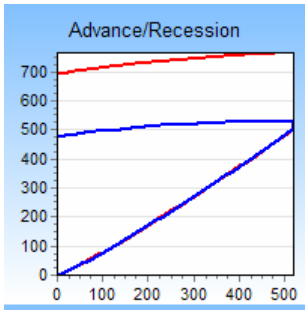
Field 19, **26/02/2001**: Furrow ave Irrigation no:7
Optimisation 1. Run by at 09::26 17/11/2006

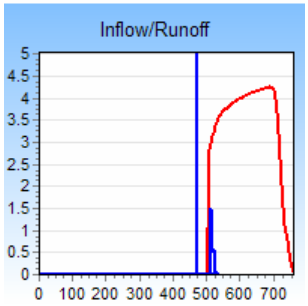




Optimisation Parameter	Measured	Optimised	Comments
Flowrate	0.0026	0.0026	
Time-to-cutoff	705	429	
Performance Measure	**Measured**	**Optimised**	**Comments**
Application Efficiency	55.3	91.5	
Storage Efficiency	100	89.8	
Application Uniformity	98.3	87.1	
Applied Volume	110708.3	66924	
Runoff Volume	39455.6	0	
Stored Volume	61360	55079.3	
Drainage Volume	10073.3	5724.6	

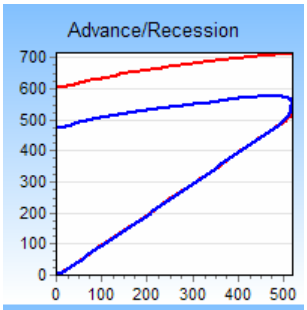
Field 20, **9/10/2000**: Furrow ave Irrigation no:1
Optimisation 1. Run by at 09::26 17/11/2006

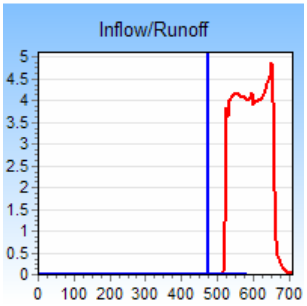




Optimisation Parameter	Measured	Optimised	Comments
Flowrate	0.00505	0.00505	
Time-to-cutoff	680	468.8	
Performance Measure	**Measured**	**Optimised**	**Comments**
Application Efficiency	66.7	94.8	
Storage Efficiency	100	97.7	
Application Uniformity	95.8	92.1	
Applied Volume	206557.7	142046.4	
Runoff Volume	52799.3	642.4	
Stored Volume	138320	135166.3	
Drainage Volume	15960.2	6672.1	

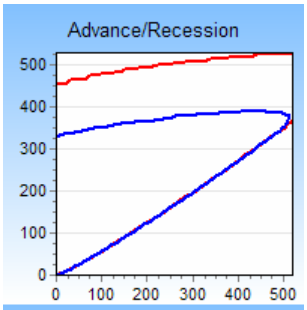
Field 20, **10/12/2000**: Furrow ave Irrigation no:2
Optimisation 1. Run by at 09::26 17/11/2006

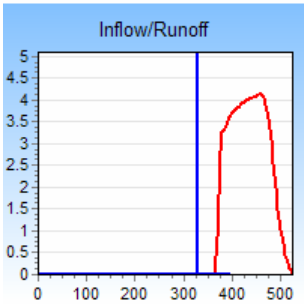




Optimisation Parameter	Measured	Optimised	Comments
Flowrate	0.00511	0.00511	
Time-to-cutoff	594	470.4	
Performance Measure	**Measured**	**Optimised**	**Comments**
Application Efficiency	80.1	100	
Storage Efficiency	98.2	97.1	
Application Uniformity	99.2	98.5	
Applied Volume	182312.1	144224.6	
Runoff Volume	36312.4	0	
Stored Volume	146348	144661.3	
Drainage Volume	0	0	

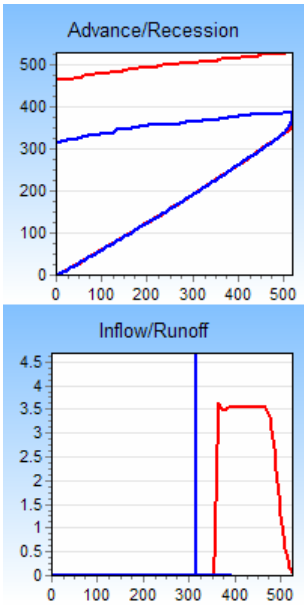
Field 20, **1/01/2001**: Furrow ave Irrigation no:3
Optimisation 1. Run by at 09::26 17/11/2006





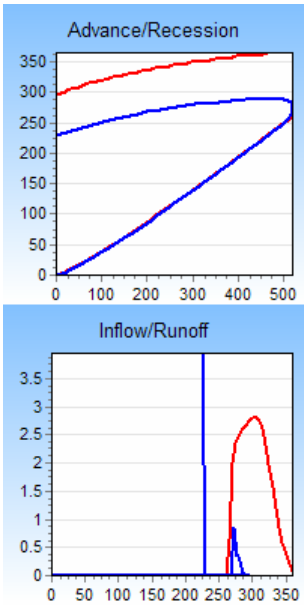
Optimisation Parameter	Measured	Optimised	Comments
Flowrate	0.0051	0.0051	
Time-to-cutoff	445	325	
Performance Measure	**Measured**	**Optimised**	**Comments**
Application Efficiency	61.2	82.8	
Storage Efficiency	100	98.3	
Application Uniformity	95.6	91.2	
Applied Volume	136910.2	99450	
Runoff Volume	28582	0	
Stored Volume	84240	82826.8	
Drainage Volume	24602.2	17072.9	

Field 20, **11/01/2001**: Furrow ave Irrigation no:4
Optimisation 1. Run by at 09::26 17/11/2006



Optimisation Parameter	Measured	Optimised	Comments
Flowrate	0.00468	0.00468	
Time-to-cutoff	450	311.4	
Performance Measure	**Measured**	**Optimised**	**Comments**
Application Efficiency	76.1	100	
Storage Efficiency	62.2	55.9	
Application Uniformity	91.8	90.9	
Applied Volume	127683.1	87441.1	
Runoff Volume	30561.2	0	
Stored Volume	97706.6	87897.2	
Drainage Volume	0	0	

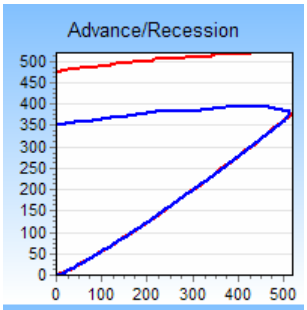
Field 20, **29/01/2001**: Furrow ave Irrigation no:5
Optimisation 1. Run by at 09::26 17/11/2006

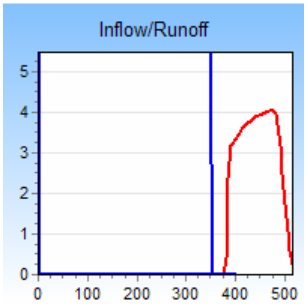


Optimisation Parameter	Measured	Optimised	Comments
Flowrate	0.00395	0.00395	
Time-to-cutoff	285	225	
Performance Measure	**Measured**	**Optimised**	**Comments**
Application Efficiency	83.4	99.3	
Storage Efficiency	78.4	73.3	
Application Uniformity	94.7	92	
Applied Volume	67952.5	53325	
Runoff Volume	11283.4	380.8	
Stored Volume	57061.9	53358.7	
Drainage Volume	0	0	

Field 20, 13/02/2001: Furrow ave Irrigation no:6

Optimisation 1. Run by at 09::26 17/11/2006

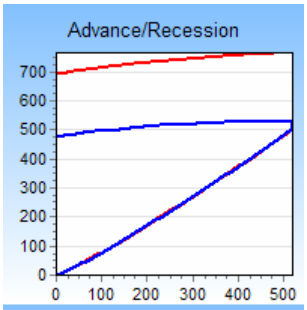


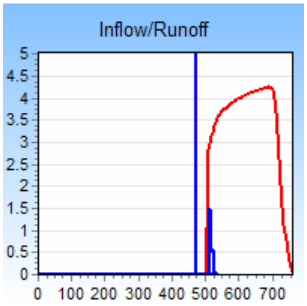


Optimisation Parameter	Measured	Optimised	Comments
Flowrate	0.00548	0.00548	
Time-to-cutoff	468	348	
Performance Measure	**Measured**	**Optimised**	**Comments**
Application Efficiency	66.6	88.2	
Storage Efficiency	100	97.3	
Application Uniformity	92.9	88.4	
Applied Volume	155402.4	114422.4	
Runoff Volume	28175.4	0	
Stored Volume	104000	101237.7	
Drainage Volume	23720.8	13553.5	

Field 20, 26/02/2001: Furrow ave Irrigation no:7

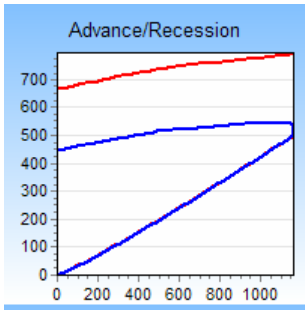
Optimisation 1. Run by at 09::26 17/11/2006

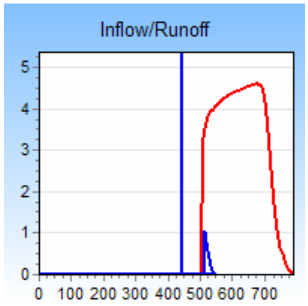




Optimisation Parameter	Measured	Optimised	Comments
Flowrate	0.00505	0.00505	
Time-to-cutoff	680	468.8	
Performance Measure	**Measured**	**Optimised**	**Comments**
Application Efficiency	66.7	94.8	
Storage Efficiency	100	97.7	
Application Uniformity	95.8	92.1	
Applied Volume	206557.7	142046.4	
Runoff Volume	52799.3	642.4	
Stored Volume	138320	135166.3	
Drainage Volume	15960.2	6672.1	

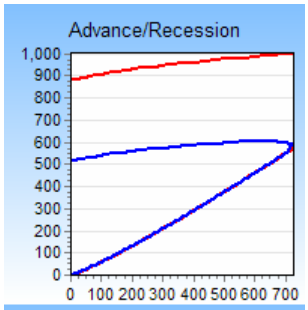
Field 17, **11/01/2000**: Furrow 8 Irrigation no:4
Optimisation 1. Run by at 09::26 17/11/2006

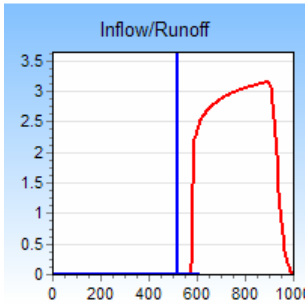




Optimisation Parameter	Measured	Optimised	Comments
Flowrate	0.00536	0.00536	
Time-to-cutoff	650	440	
Performance Measure	**Measured**	**Optimised**	**Comments**
Application Efficiency	72.1	99.4	
Storage Efficiency	92.5	85.6	
Application Uniformity	96.7	93.6	
Applied Volume	210553.5	141504	
Runoff Volume	58701.8	883.1	
Stored Volume	152287.8	141059.7	
Drainage Volume	0	0	

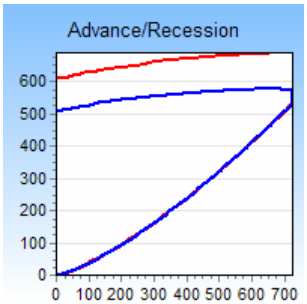
Field 18, **1/10/1999**: Furrow 8 Irrigation no:1
Optimisation 1. Run by at 09::26 17/11/2006

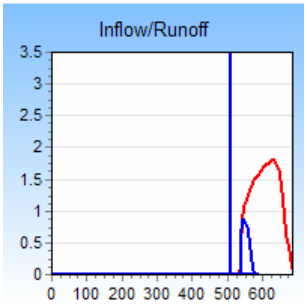




Optimisation Parameter	Measured	Optimised	Comments
Flowrate	0.00364	0.00364	
Time-to-cutoff	873	512.4	
Performance Measure	**Measured**	**Optimised**	**Comments**
Application Efficiency	53	88.8	
Storage Efficiency	100	98.1	
Application Uniformity	96.5	91	
Applied Volume	190925.9	111908.2	
Runoff Volume	63763	0	
Stored Volume	101500	99612.6	
Drainage Volume	25904.7	12512.4	

Field 18, **11/01/2000**: Furrow 2 Irrigation no:4
Optimisation 1. Run by at 09::26 17/11/2006





Optimisation Parameter	Measured	Optimised	Comments
Flowrate	0.0035	0.0035	
Time-to-cutoff	600	503.4	
Performance Measure	**Measured**	**Optimised**	**Comments**
Application Efficiency	65.1	76.8	
Storage Efficiency	100	98.4	
Application Uniformity	89.4	85.8	
Applied Volume	126452.9	105714	
Runoff Volume	12052.2	1391.8	
Stored Volume	82650	81364.6	
Drainage Volume	32044.4	23171	

Appendix 7.1 Software engineering tools

The choice of software engineering tools used in developing the FIDO decision support system has heavily influenced all aspects of its final design. The importance of this cannot be understated, and therefore necessitates documentation and discussion. Poor choice of tools can severely limit program functionality and may be irreversible once development is underway. Hence, careful consideration is required at the commencement of the project as to the:

- Choice of operating system;
- Choice of programming language(s);
- Choice of database environment;
- Choice of development environments; and
- Choice of third party tools and libraries.

A7.1.1 Target operating system

Microsoft Windows was chosen as the target operating system for *FIDO*. Cross-platform development was initially considered, but rejected on the grounds that:

- It is assumed that the target audience for the software would have access to *Microsoft Windows*. Many of these users would already be using *SIRMOD* or *SRFR* which can operate under the *Windows* environment (although there can be problems running *SRFR* as it is *DOS* based).
- Cross-platform development severely limits the choices of libraries, tools, and languages available. The true power of the *FIDO* software lies in its integration with existing third-party *Windows*-based libraries.

A7.1.2 Programming languages

C++ was used as the language of choice for developing *FIDO*, although others such as *Object Pascal*, *Fortran* and *BASIC* were also considered.

The benefits of using *C++* as a programming language for developing *FIDO* include:

- **It is an intermediate to high-level language**: Computer languages can be roughly categorized between high and low level languages. Higher level languages are distinguished as being closer to human languages and further from machine languages and are therefore easier to read, write and maintain but ultimately must be translated into machine language by a compiler or an interpreter (http://www.webopedia.com/TER/H_/high_level_language.html). This process can occur at development time before "running" the program such as with *C* and *Pascal*, or it can occur at run-time with languages like *BASIC*.
- **It is a precompiled language:** Precompiled languages such as *C++* and *Object Pascal* offer considerable performance advantages over interpreted languages such as *BASIC*, and until recently have been considered the standard for writing "serious" applications.

- **Traditionally it has been a mathematical language:** A large range of mathematical libraries exist in *C* and *C++* which are not available in languages such as *BASIC* and *Object Pascal* (although this is changing).

Another option that was investigated was the use *Microsoft's .NET* technology (http://www.microsoft.com). This would have permitted the use of the *Managed C++* language (now known as *C++/CLI*) or *C#* which would be compiled to a "*Common Language Runtime* (*CLR*)" which is then interpreted at runtime. This is quickly becoming the most popular technology for *Windows*' software developers and offers considerable advantages in software reuse and development. This language did not exist when this research started, and was still in its infancy when the most current version of FIDO was developed. At that time, there were a number of questions arising about the *CLR*'s mathematical performance. Reviewing user comments on the internet revealed a full range of opinions as to its suitability. It was thought that recursive mathematical operations "may" run as quickly as the fully compiled equivalent, since the technology compiles the code on the first iteration. Since FIDO has a modular structure separating interface and program code, it could quite easily be converted into a .NET program in the future, if a potential benefit is seen to exist (including third party support).

A7.1.3 Database environment

XML (eXtensible Mark-up Language) and *XML Schema* technology was chosen to develop the database components of *FIDO*. *XML* is a simple and flexible text format originally designed for use in large-scale electronic publishing. However, it is now widely used for data exchange and storage for all sorts of computing applications (www.w3.org/XML). Its main characteristics include (www.dclab.com):

- Only standard text (*ASCII* or *UniCode*) is used with a document;
- All data is clearly identified inside user-defined tags;
- The document remains unformatted although formatting information can be included in the *XML* tagging; and
- The document usually conforms to a user-defined rule-set or template such as a schema or *DTD* (Document Type Definition) although this is optional.

The two main benefits of using *XML* (www.dclab.com/xmlbenefits_p1.asp) as a database include:

1. **Content identification**: text elements are identified on the basis of what they are and not what they look like. Data is encapsulated within user-defined tags. For example, title information could be presented as `<Title>This is the title</Title>`.
2. **International standard**: *XML* is now an international standard that is maintained by an independent standards committee. Because of this it is compatible with a wide range of software products.

In the early stages of this project, the database was originally developed using *Microsoft Access* and connected to the *FIDO* interface through *Borland*'s database controls that ship with *Builder*.

A7.1.4 Development environments

Two main software engineering tools were used while developing *FIDO*. *Borland C++ Builder* (www.inprise.com) was used as the integrated development environment for programming tasks, while *XML Spy* (www.altova.com) was used to develop the *XML* database and reporting components.

Borland C++ Builder (Figure A7.1.1) was chosen over other programming development environments for two main reasons: *firstly* because it is an object-oriented programming environment for *Windows* using the *C++* language (and also *Object-Pascal*); and *secondly*, because it is compatible with the class-leading third-party components *TeeChart* (www.teemach.com) and *VirtualTreeView* (www.delphi-gems/VirtualTreeview) which feature prominently throughout the *FIDO* decision support system.

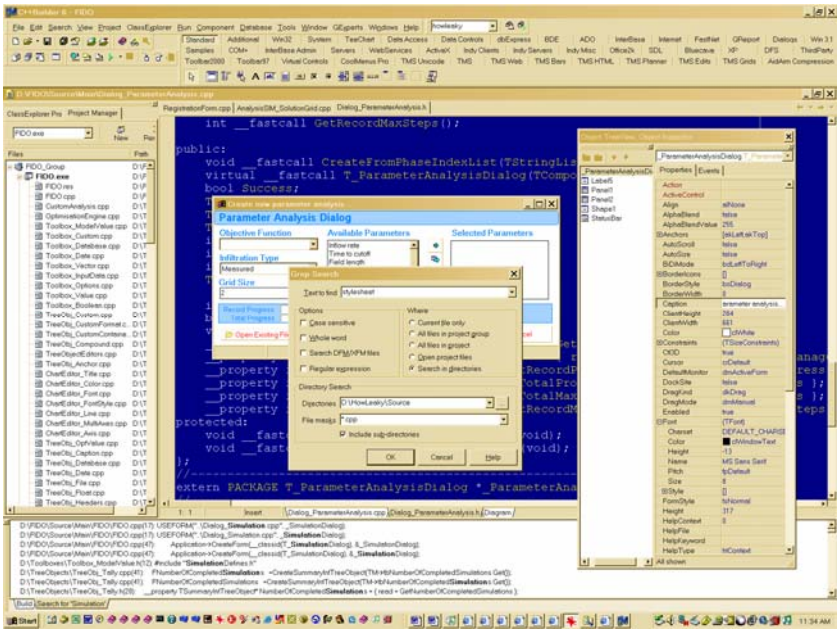


Figure A7.1.1: *Borland C++ Builder* has been used as the integrated development environment for developing *FIDO*.

Other products which were also considered include *Borland Delphi*, *Microsoft Visual C++*, and *Microsoft Visual Basic*. *Delphi* is *Builder*'s sister product and they both share the same *Windows* programming library (*Visual Class Library*) and compile to the same object-code. However, *Delphi* was overlooked as it is an *Object-Pascal* programming environment, and not this author's language of choice. *Visual C++* is arguably the industry leader in integrated development environments. However, it is let down by the way it handles third-party components. This area is the real strength of *Builder* (and *Delphi*) as it allows the developer to embed third-party components called "packages" directly into your own applications. In contrast, *Visual C++* requires dynamic link libraries that are external to the developed program, with very limited communication ability. *Visual Basic* was also briefly considered at the start of the project, but was dismissed based on its slower performance, especially when dealing with mathematical equations. However, it is often favoured by novice programmers for it simplicity.

XML Spy 2004 was used to develop the database and input/output components of *FIDO* and is now an industry standard for *XML* development. *XML Spy* allows you to graphically create a database or file structure using schema technology (Figure A7.1.2).

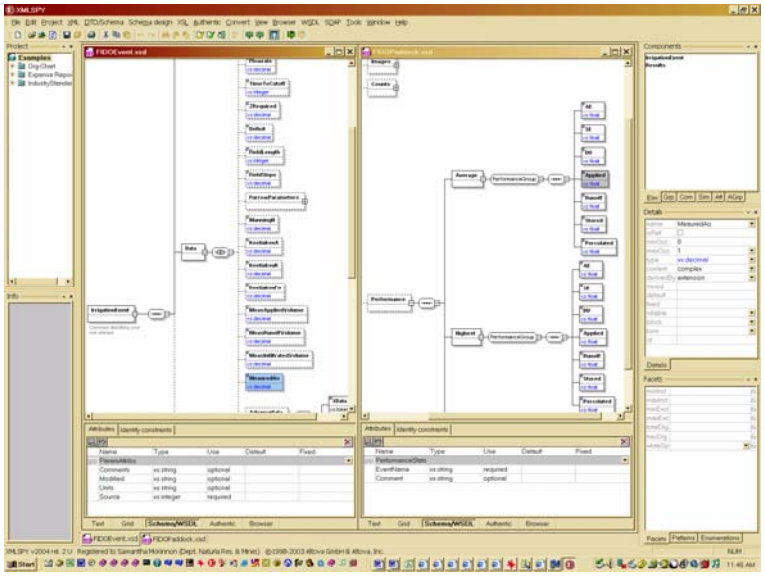


Figure A7.1.2: *XML Spy 2004* has been used to develop the *XML* database and reporting capabilities of *FIDO*

A7.1.5 Components and libraries

The use of "packages" in the *Borland C++ Builder* development environment is one of its greatest attributes. In the world of object-oriented programming where "reuse" is one of the primary goals, "packages" offer a simple means to distribute components so others can use them. These days, there are numerous companies dedicated to component writing. As a software developer, it makes good sense to use these third-party components rather than having to "reinvent the wheel". *FIDO* relies heavily on this technology making extensive use of two third-party components: *TeeChart* and *VirtualTreeView*. It also extensively uses the *XML* component library which ships with *Borland C++ Builder*. Without these components, *FIDO* would look and feel very different to its current form. It would also be much less powerful since these tools are widely regarded as "state of the art".

TeeChart is a class-leading and award-winning charting and plotting tool by *Steema Software*, written in *Object-Pascal* using the *Visual Class Library*. Because it uses this library, it works seamlessly in *Borland C++ Builder*, and is easily modified and customised. *FIDO* incorporates dozens of *TeeChart* objects throughout its interface, in both original and modified form. They are used for outputting results graphically, animating the simulation of water flowing down the furrow, and even as a slider-bar control for manipulating outputs (Figure A7.1.3).

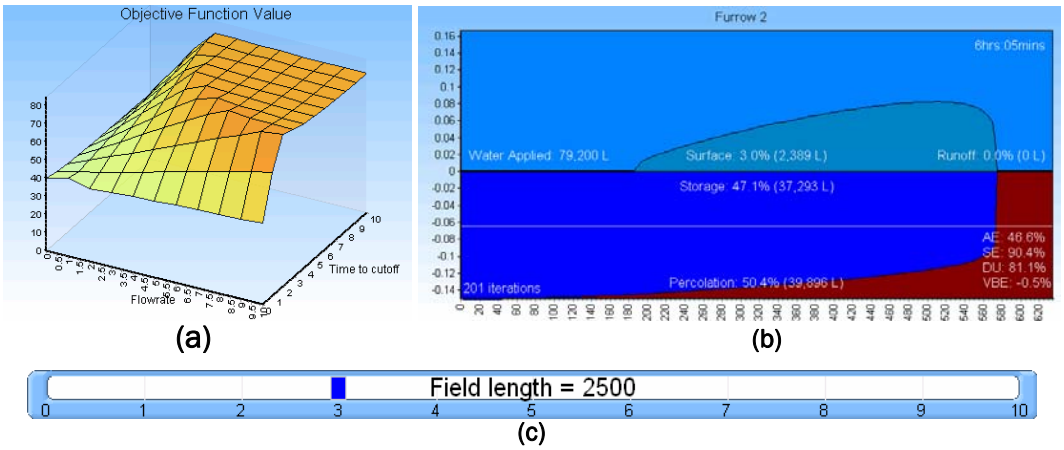


Figure A7.1.3: Examples of *TeeChart* as used in *FIDO*: (a) 3D surface for parameter analysis generation; (b) as a graphical animation of the simulation output; and (c) as a slider bar control.

VirtualTreeView is a powerful treeview control that is used throughout *FIDO* as the graphical control for data entry, selecting and viewing (Figure A7.1.4). This tool is quite unique in that it uses a different paradigm for tree management than other existing treeview controls. This component does not know anything about the data it manages (other than its size) and uses "events" to retrieve information and display it to the screen. It is very fast and has a very small memory footprint.

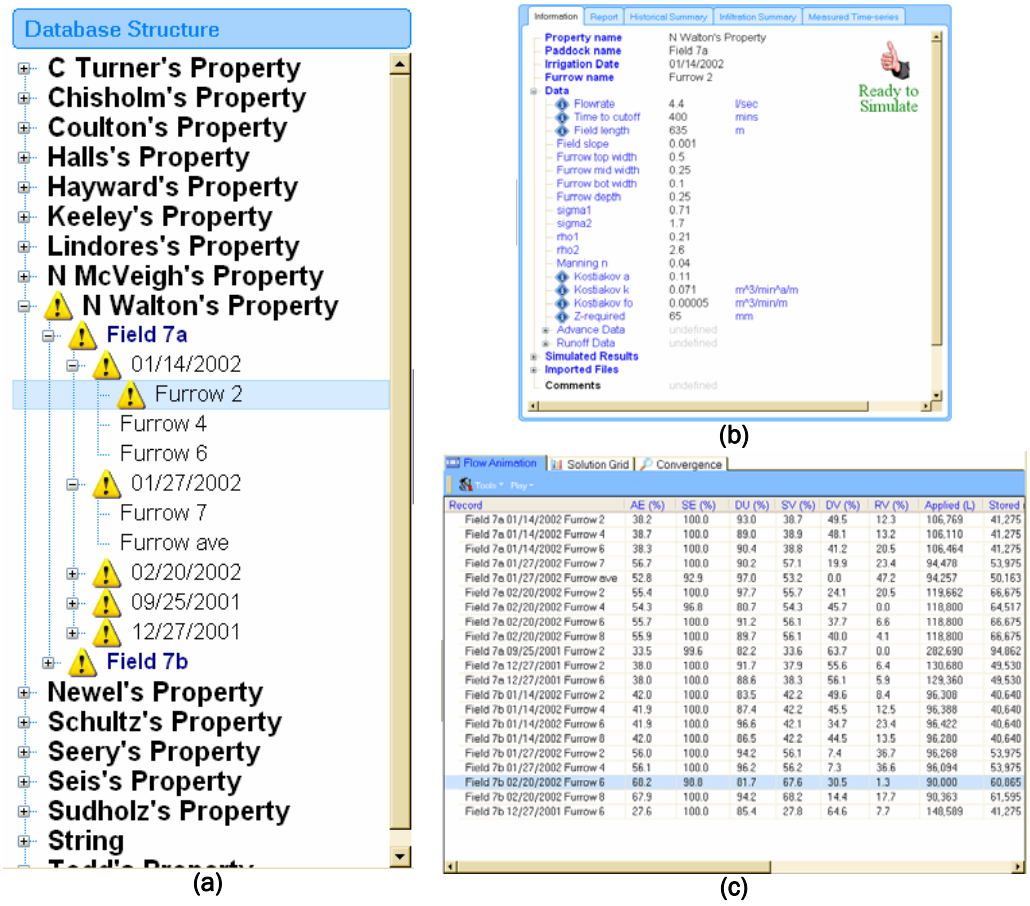


Figure A7.1.4: Examples of *VirtualTreeView* throughout *FIDO*: (a) As a data selector; (b) for data entry; and (c) as a grid control for data output.

The *XML* component library that ships with *Borland C++ Builder* is used extensively in *FIDO* for database management, report generation, and system file manipulation. This key component in this library is *TXMLDocument* which uses an external "Document Object Model" (DOM) parser to analyse any *XML* document. With it the user can load a document, read and modify it, and a save changes. In *FIDO,* it is used closely in conjunction with the *VirtualTreeView* component.

Appendix 7.2 FIDO XML Data Structures

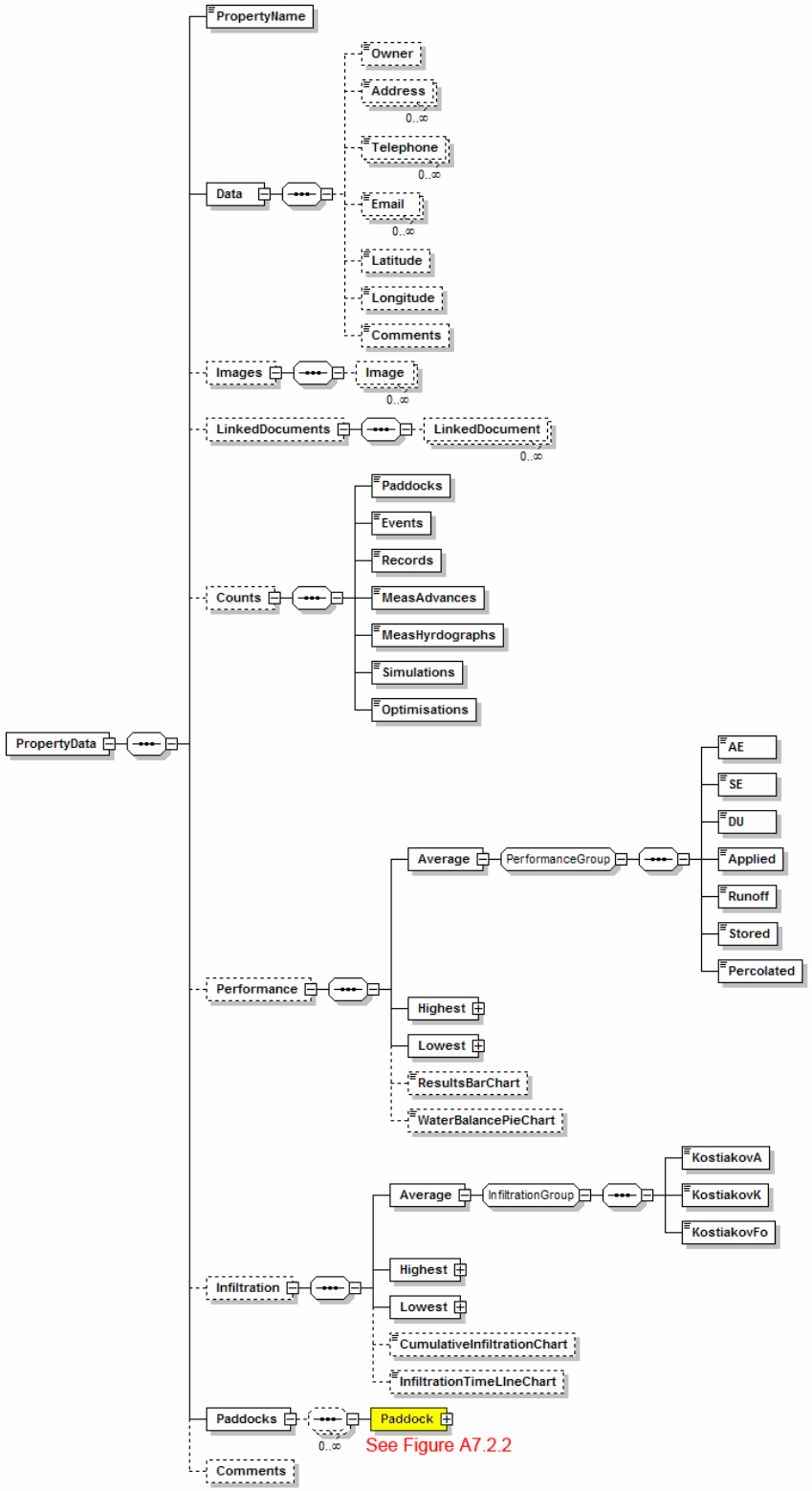


Figure A7.2.1: Property Data Structure

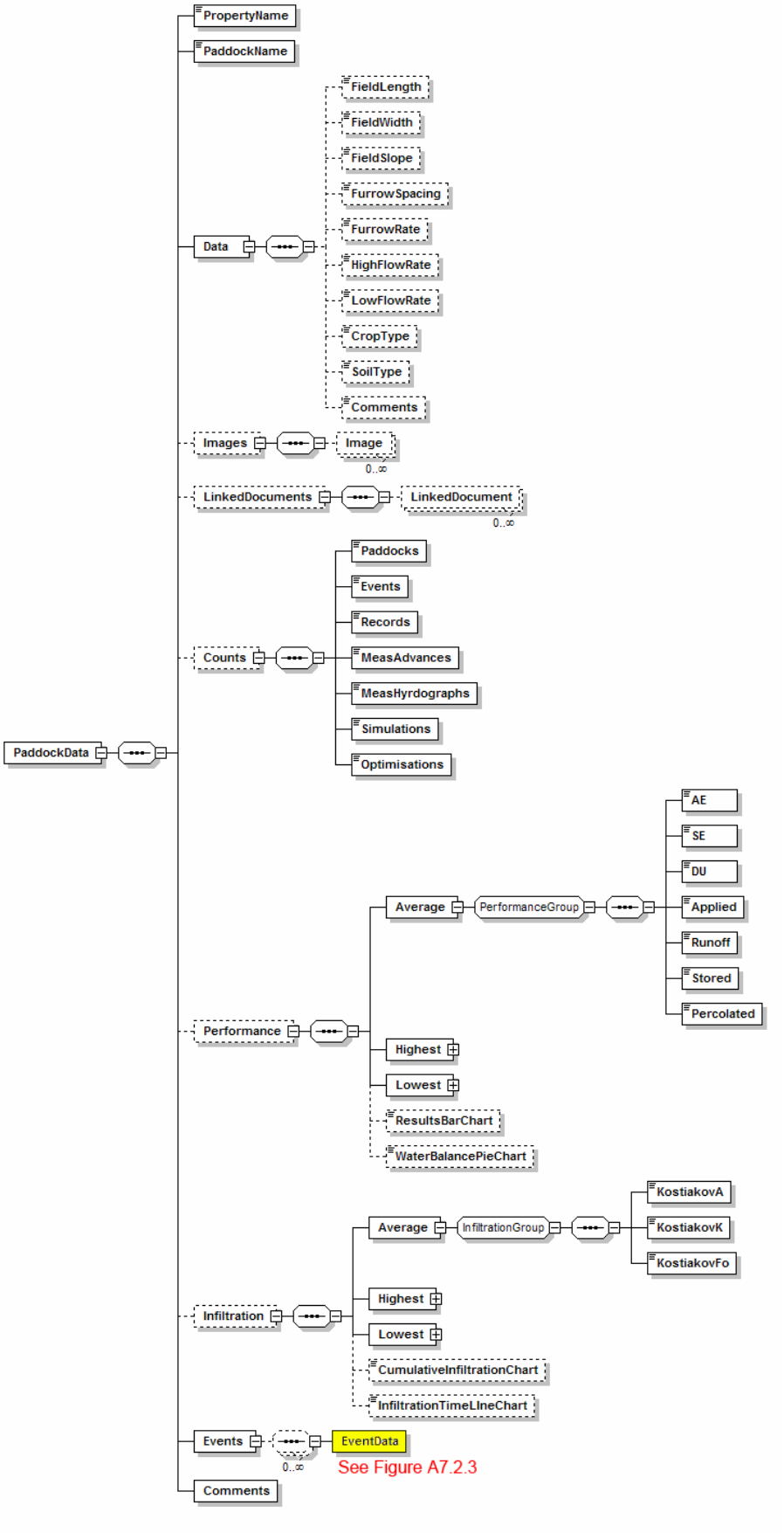


Figure A7.2.2: Paddock Data Structure

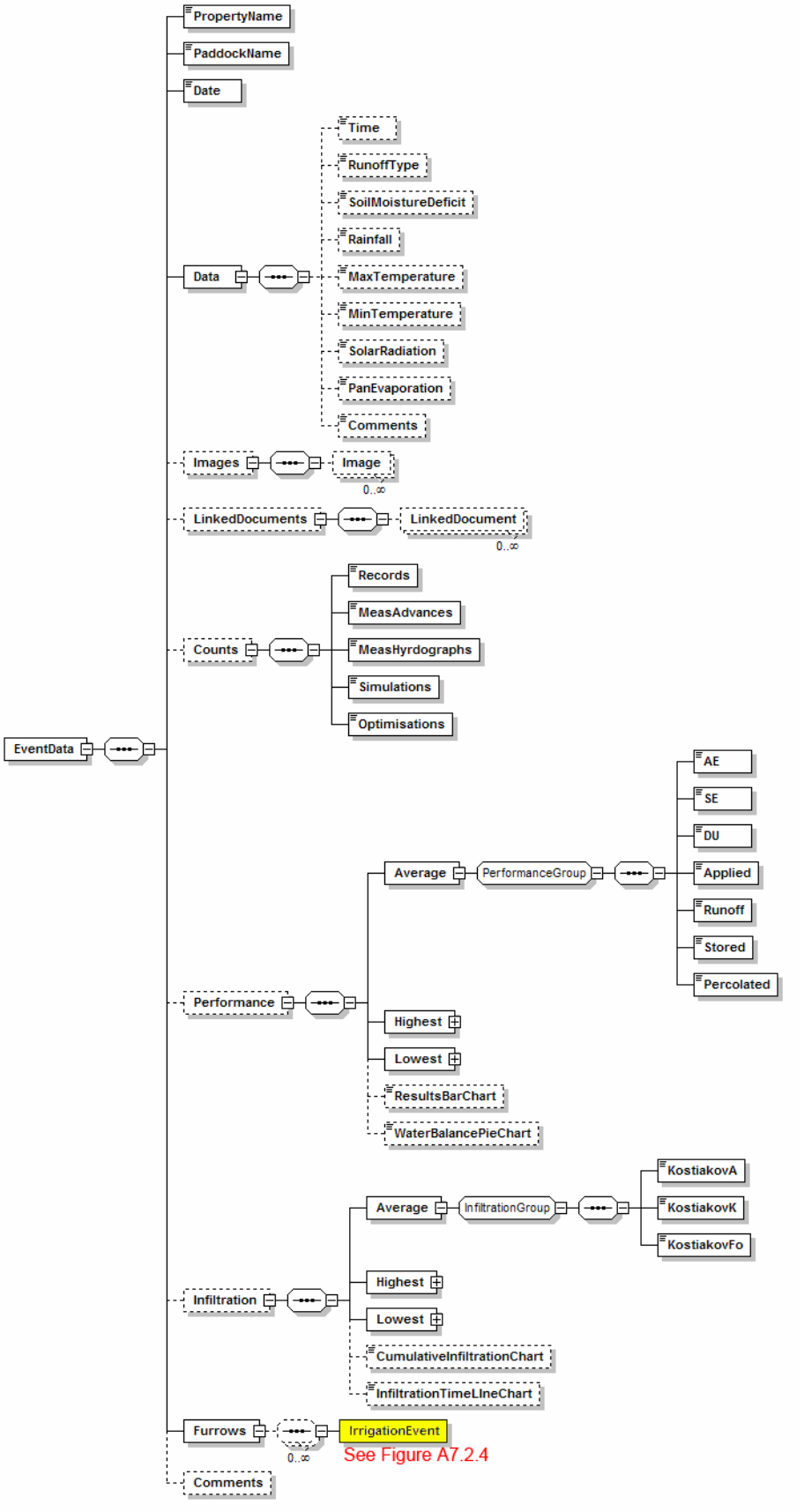


Figure A7.2.3: Event Data Structure

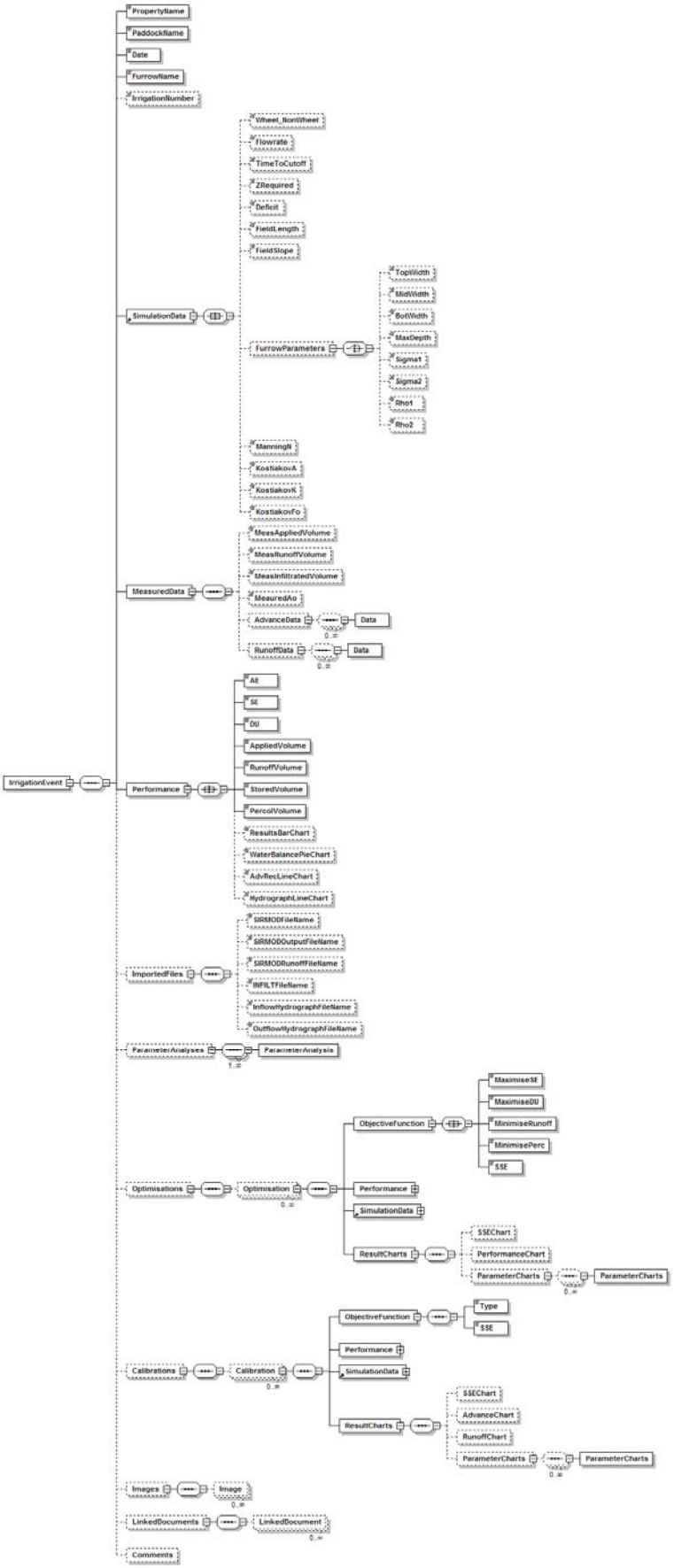


Figure A7.2.4: Event Data Structure

Appendix 7.3 Evolution of *FIDO*'s simulation GUI

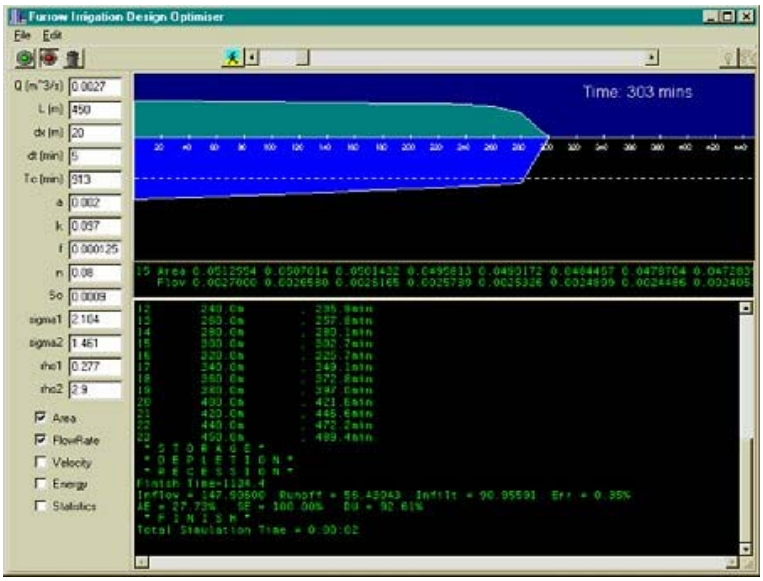


Figure A7.3.1: Main interface version 1. This is the very first version of *FIDO* with textural outputs and simple animation.

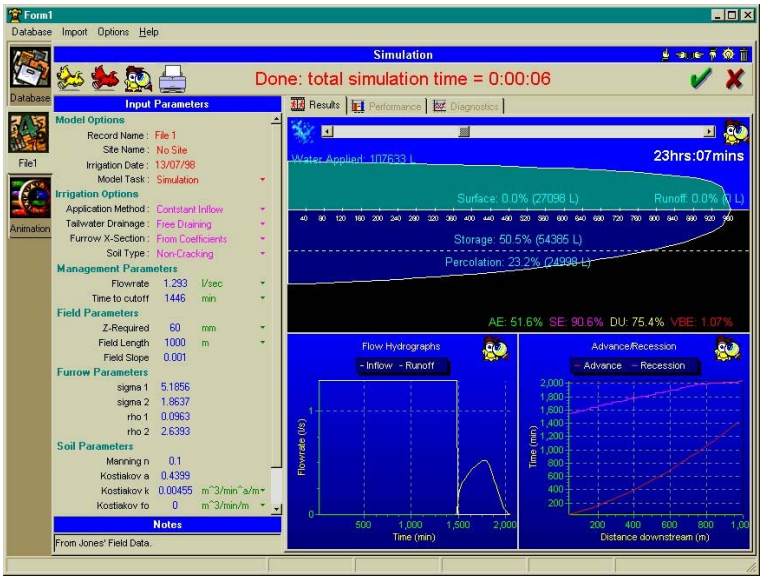


Figure A7.3.2: Simulation Animation, early version. First full working version of the *FIDO* simulation with detailed simulation outputs. Database was primitive with many controls that were only there for show.

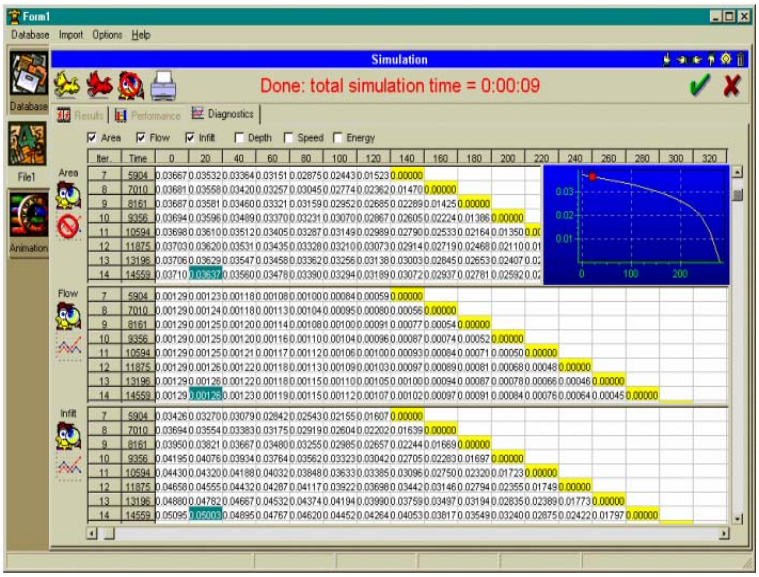


Figure A7.3.3: Textural Outputs, early version. Textural outputs for flow-area, flowrate and infiltration. This was a very powerful feature of the early version and proved very helpful in debugging the simulation. However, it was also poorly written using a large amount of code and proved difficult to extend and maintain.

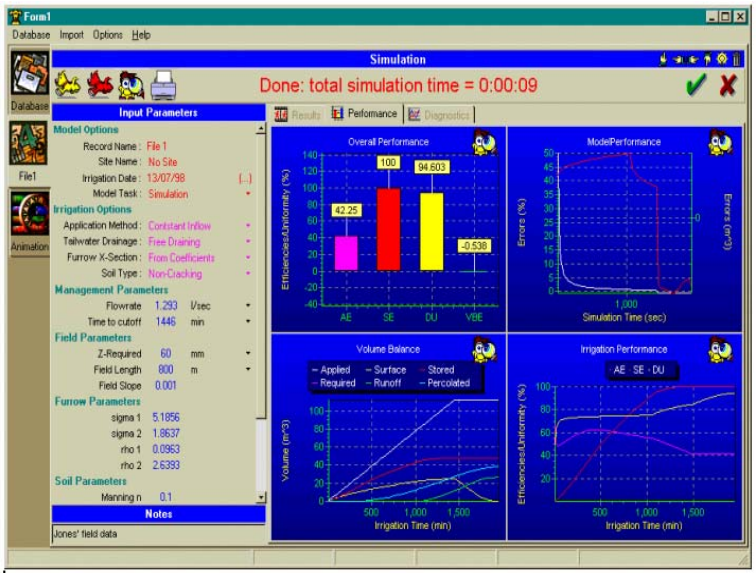


Figure A7.3.4: Performance outputs, early version. Another output from the early version showing how irrigation performance varies with simulation time.

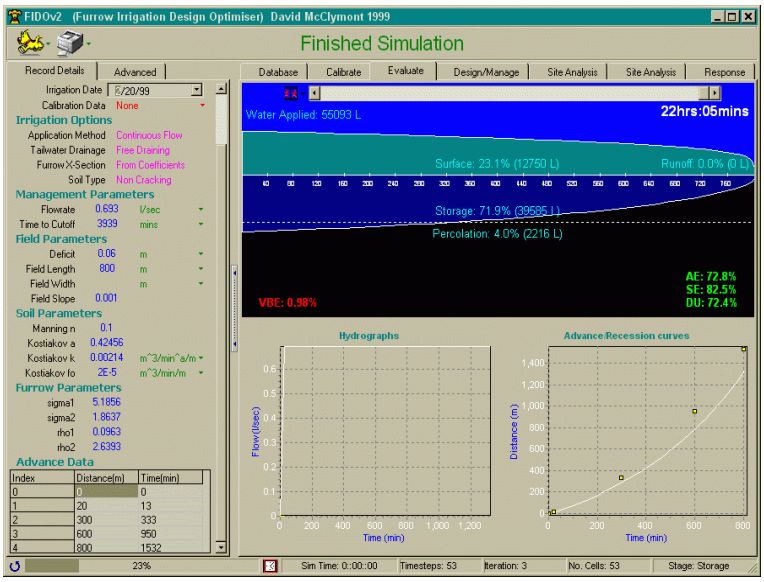


Figure A7.3.5: Simulation animation, early version. Early attempt at integrating all decision support components into a single program. Interface appears clean, many features remained operational. Measured advance data is incorporated in inputs and outputs.

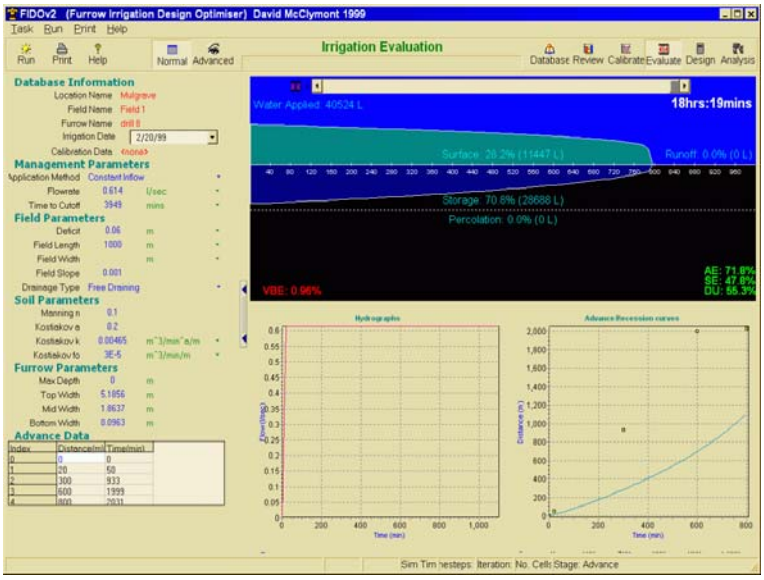


Figure A7.3.6: Simulation animation, early version. This version attempts to simplify the interface through using task-buttons, and remove many of the unnecessary controls presented in the previous figure.

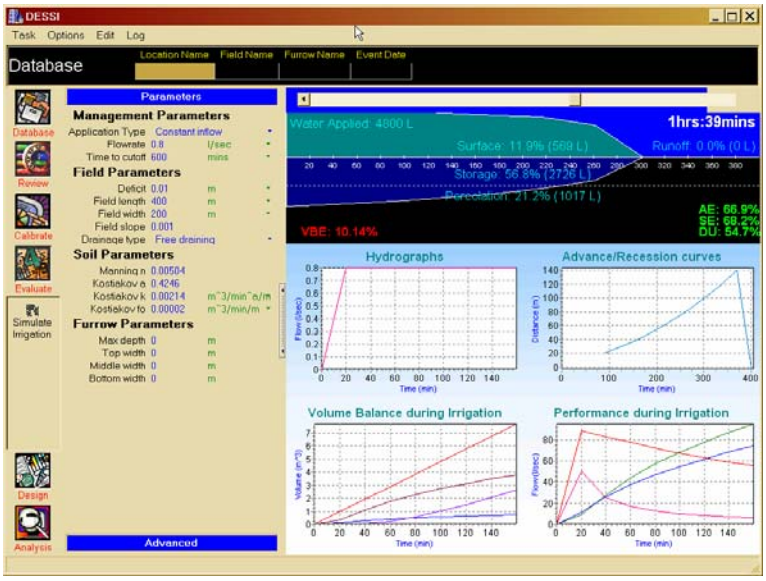


Figure A7.3.7: Advanced simulation outputs. This version tries to simplify the interface further by placing task buttons along the left hand side of the screen. Extra simulation outputs are presented. It appears that there was an error during the simulation. The software name was temporarily changed to *DESSI* (Decision Support for Surface Irrigation), but was later changed back to *FIDO* (proving that the initial name adopted by software is hard to change).

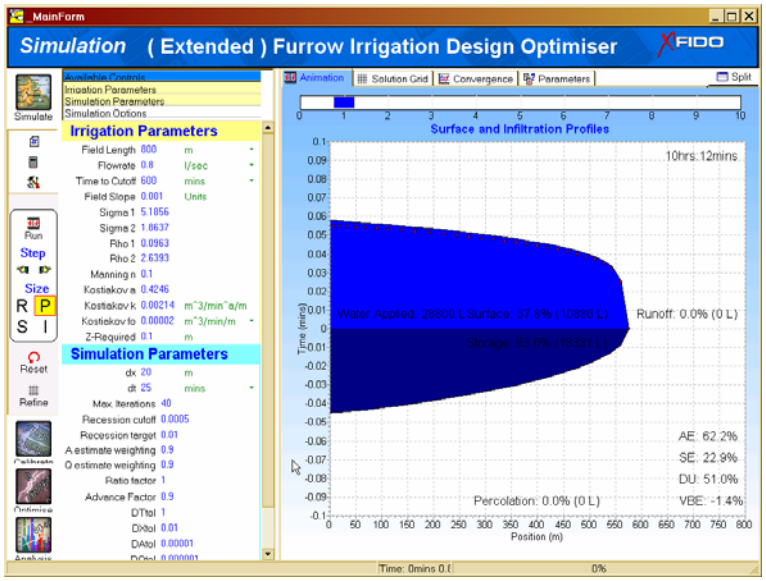


Figure A7.3.8: X*FIDO* simulation. Advanced interface developed to help debug the simulation engine. Code-named X*FIDO* (Extended *FIDO*) it presented previously hidden simulation parameters, and new controls for stepping through and debugging the simulation. Advanced outputs were also presented including initial estimates for flow-area at each time-step. These estimates are shown as red dots on the animation which could be changed by dragging the points with the mouse.

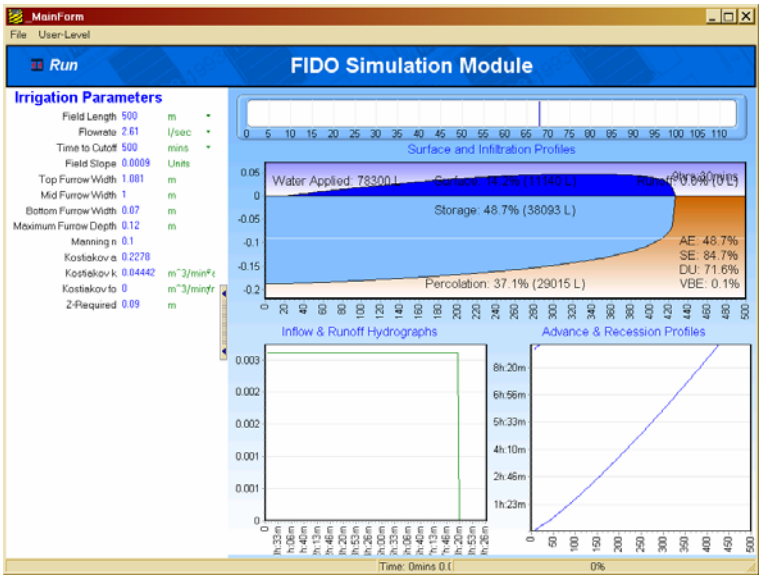


Figure A7.3.9: Advanced X*FIDO* outputs. This version of the software was developed as a compact version with a minimum of features presented. This was created when the decision support software was becoming very large and difficult to maintain. It was therefore split into separate models for development purposes.

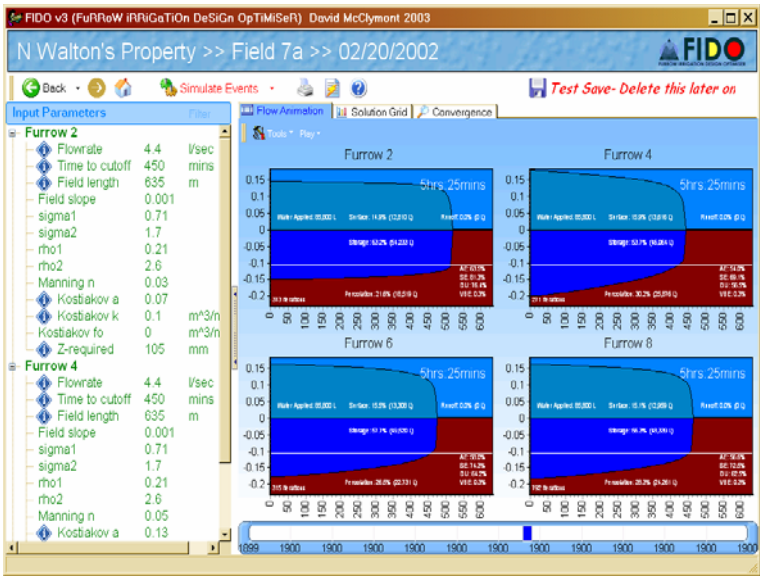


Figure A7.3.10: Multiple simulations. Prototype of the current version of *FIDO*. This prototype version was never operational being plagued by instability problems. This version was the first to simultaneously display and compare multiple outputs.

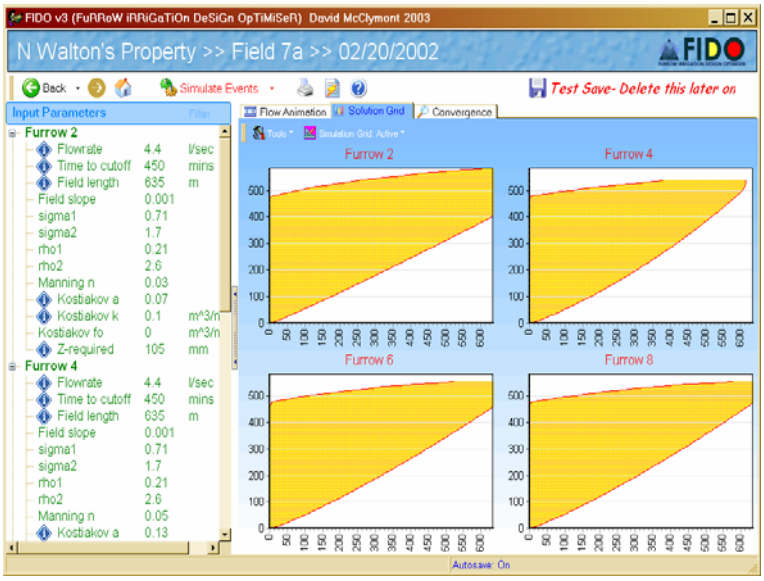


Figure A7.3.11: Solution grids. Solution grid analysis from the prototype of the current *FIDO* version. This proved to be a useful development tool for debugging the simulation engine with the capability to zoom into the grid to study irregularities.

Appendix 7.4 Evolution of *FIDO*'s calibration GUI

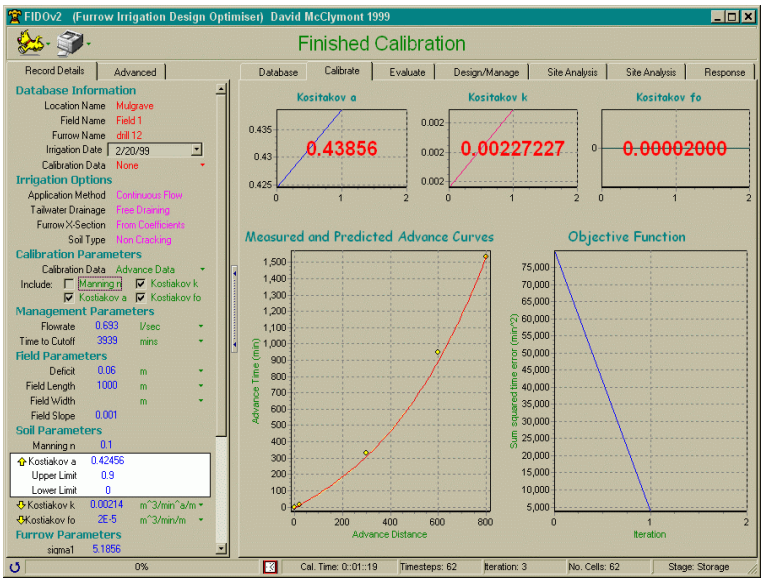


Figure A7.4.1: First calibration attempt. First attempt at calibration using the hydrodynamic model. Calibration input options were permanently presented to the user.

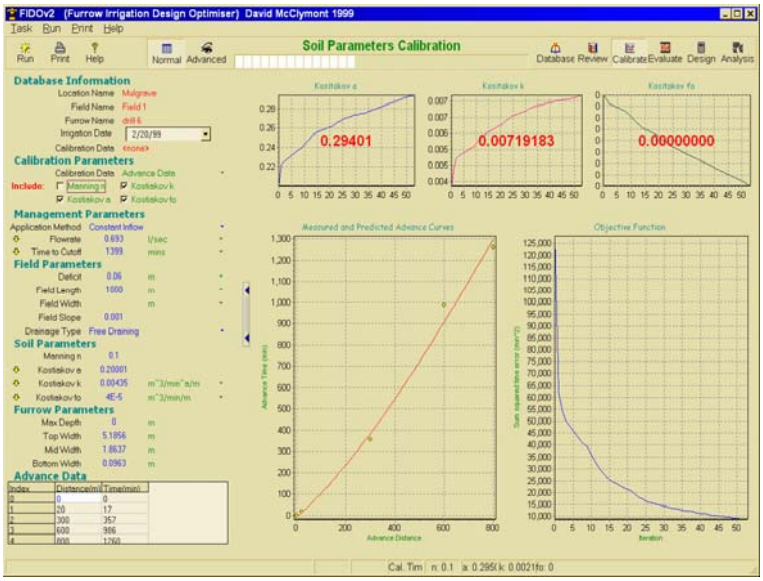


Figure A7.4.2: Early calibration interface. Early version of calibration interface, which has undergone structural simplifications since the previous version. Both “normal” and “advanced” calibration inputs were available.

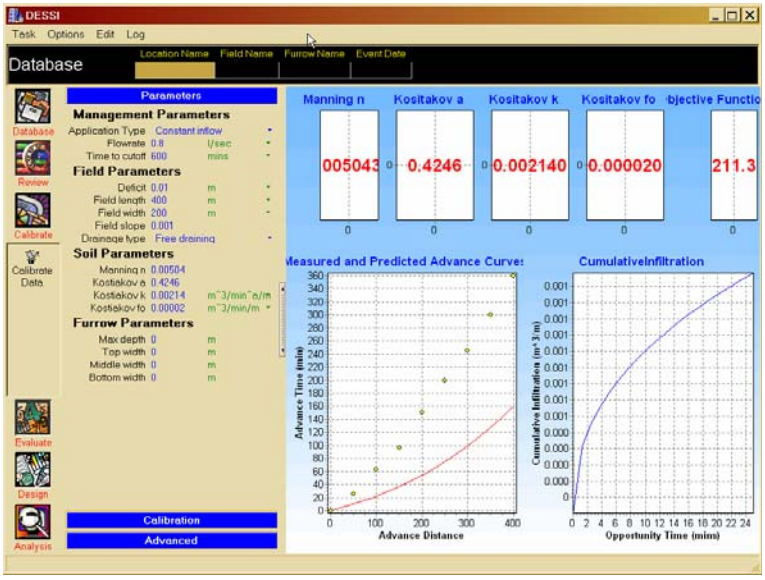


Figure A7.4.3: Early attempt at calibration. Calibration inputs are located in a separate pane in the input panel. Cumulative infiltration is presented as an output of the calibration.

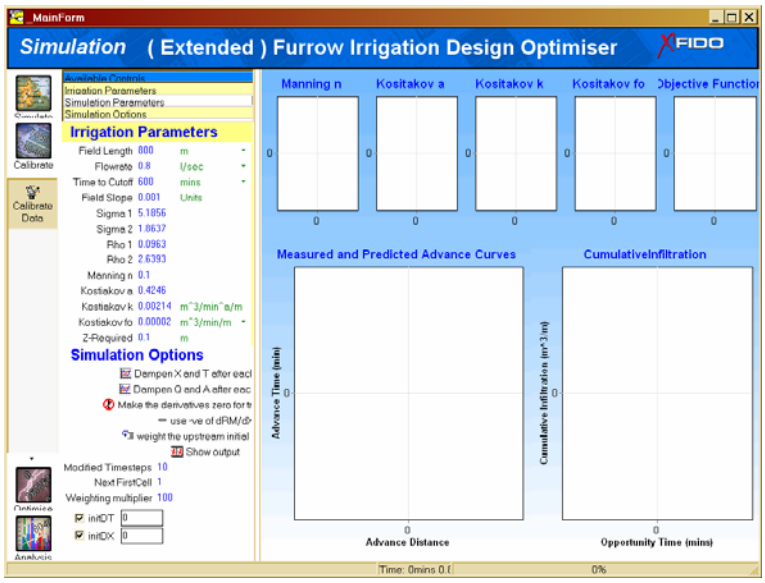


Figure A7.4.4: Advanced calibration interface used in X*FIDO* (Extended *FIDO*). This version was never operational, but was developed to help debug the calibration, with the ability to try different simulation stability measures and see the effect on the calibration response.

Appendix 7.5 Evolution of *FIDO*'s optimisation GUI

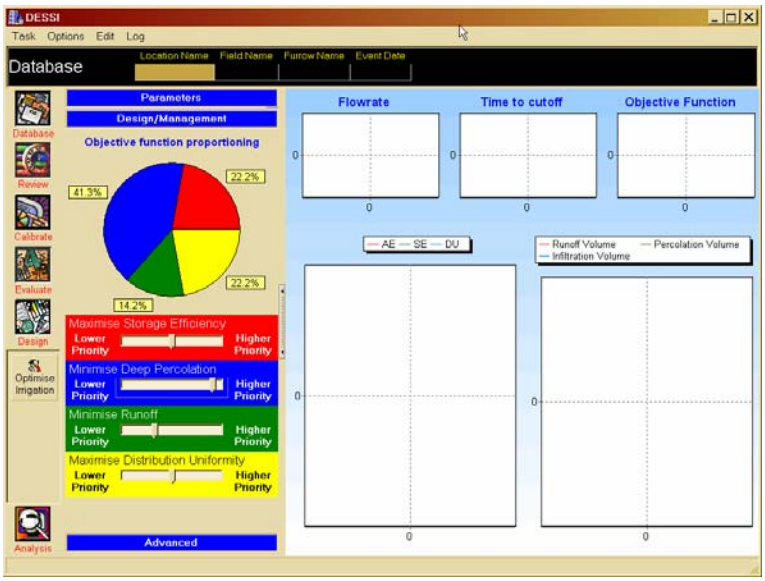


Figure A7.5.1: Early optimisation interface. Early version of the optimisation interface showing the optimisation priority setter. This was the second version of the setter tool adding a pie chart to help visualise priority proportions. This solution caused several problems relating to the proportional nature of the task. Changing the setting of one bar inadvertently changed the other weightings automatically as the relative position between the bars changed. Also several different setting arrangements could be used to achieve the same result.

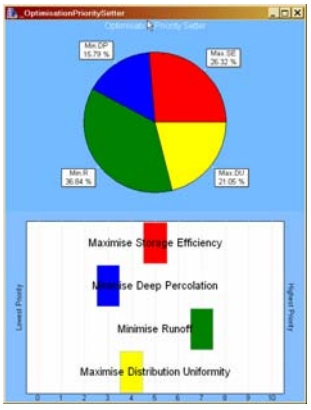


Figure A7.5.2: Early objective-function setter. Updated version of the objective-function priority setter presented as a popup dialog. This version was created from two *TeeChart* objects, rather than a series of standard controls. This tool was designed so that moving one slider bar automatically updated the position of the other slider bars in proportion. In practice, this proved to be an awkward solution with an unnatural feel.

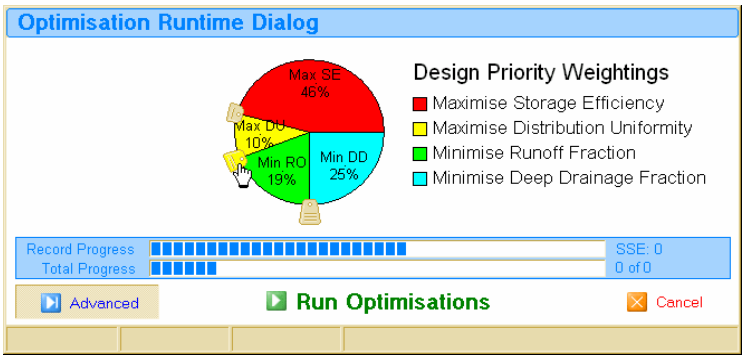


Figure A7.5.3: Advanced objective-function setter. Prototype version of the current objective-function priority setter. This version is a custom made component with “handles” that appear when the mouse is moved over the pie-chart. In this way, the user has direct control over adjusting priority proportions, overcoming problems associated with the previous slider controls.

Appendix 7.6 Evolution of *FIDO*'s parameter analysis GUI

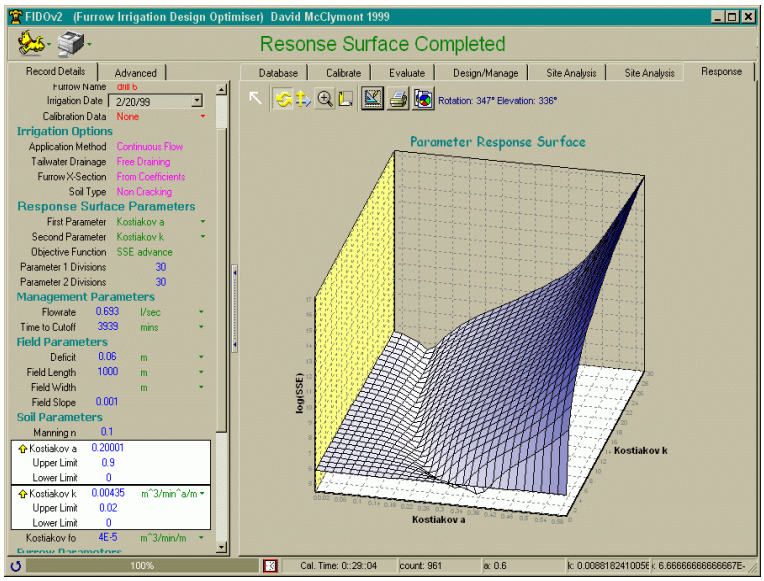


Figure A7.6.1: First version of parameter analysis interface. First version of the parameter analysis interface showing a calibration response-surface. This version was limited in that only one output could be shown at a time.

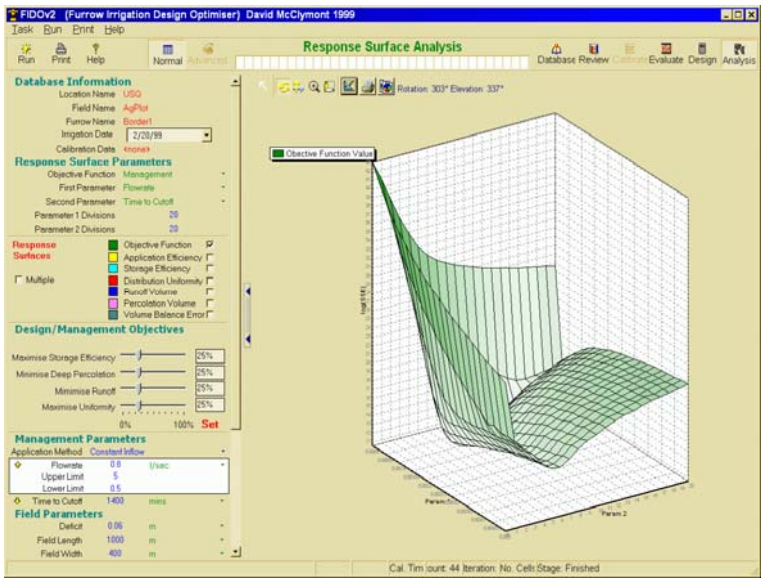


Figure A7.6.2: Updated parameter analysis interface. An updated version of the parameter analysis interface with an objective-function priority-setter to dynamically change the objective-function response-surface in real time. Different outputs are selectable via different options in the input panel. Different input options are also presented. These options are only shown when the parameter analysis tab is selected.

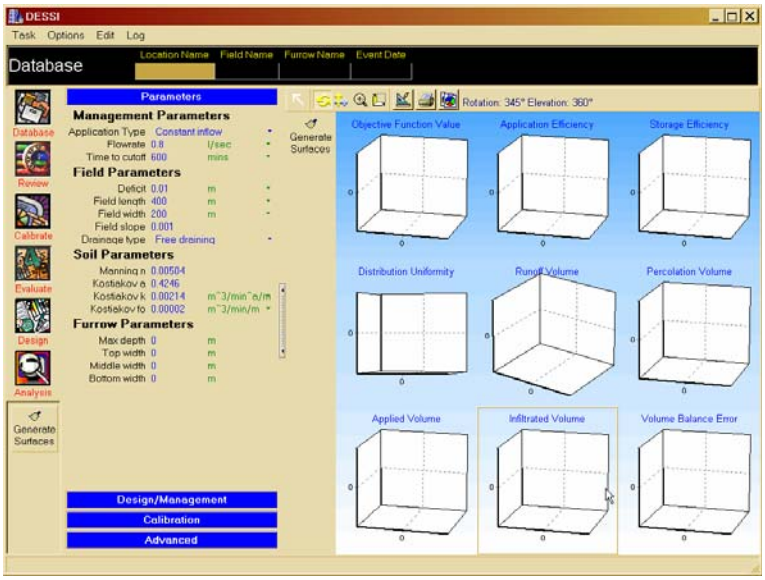


Figure A7.6.3: Multiple views in parameter analysis interface. Updated version of the previous interface with advanced options hidden in different input panel views. Multiple outputs can now be compared simultaneously.

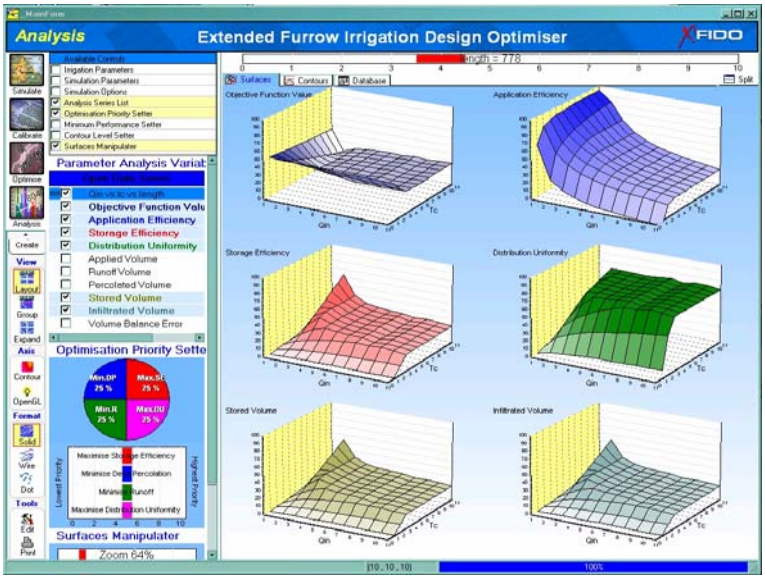


Figure A7.6.4: Multiple outputs in parameter analysis interface. Advanced interface from the X*FIDO* (Extended *FIDO*) version. This version was developed as a debugging tool for the simulation. This was the first version to present a slider bar to study output for a third parameter. Outputs can then be animated through adjusting the slider bar position.

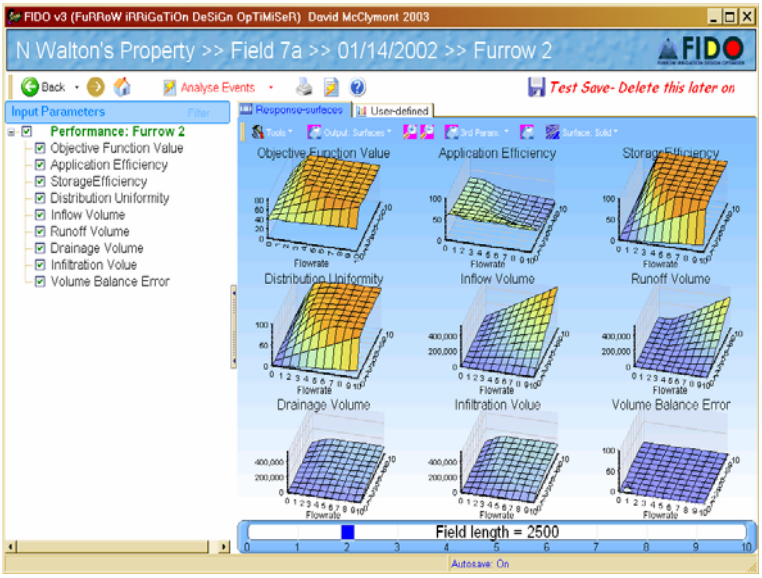


Figure A7.6.5: Prototype of current parameter analysis interface. Prototype interface for the current version of *FIDO*. In this example, field-length is represented by the slider-bar at the bottom of the screen. Optimisation priority was temporarily omitted from this version.

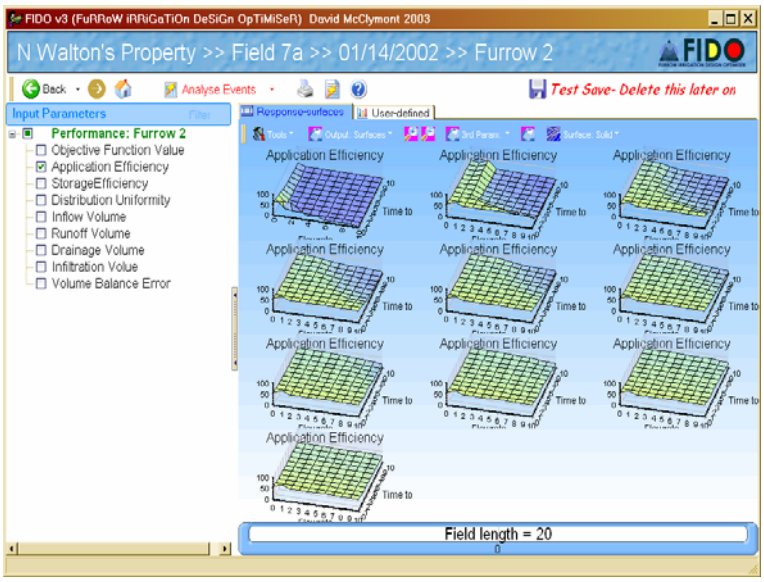


Figure A7.6.6: Prototype showing 3rd parameter expansion. Prototype of current version showing the third parameter expanded as a series of charts of application efficiency with each chart representing a different field-length.

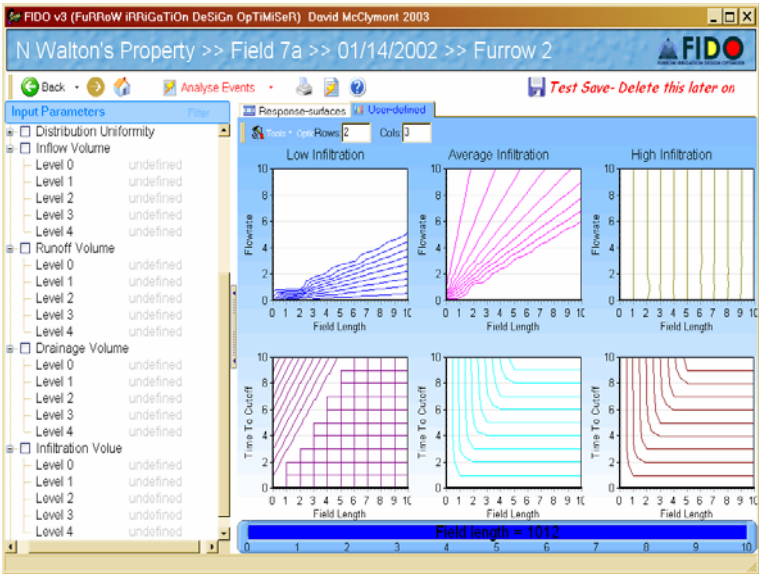


Figure A7.6.7: Prototype contour plotter. First version of the user-defined parameter analysis in which the user "drags and drops" performance outputs onto a user-defined grid of charts. In this example, the different columns represent different infiltration properties.

Appendix 7.7 Evolution of *FIDO*'s database GUI

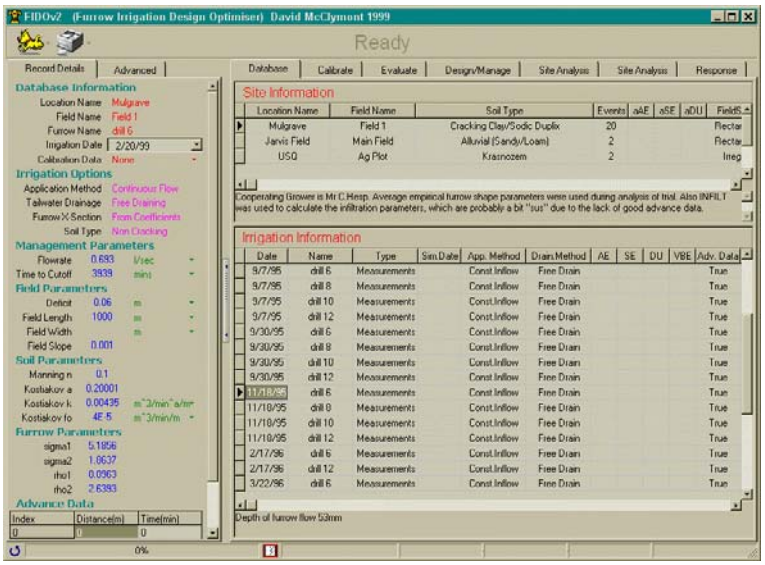


Figure A7.7.1: Original FIDO database. First version of the *FIDO* database interface. This version used the *Microsoft Access* database engine with specialised database controls. Data was separated into a site database and an event database which were displayed in tables. This interface was seen as overly complex, with too much information being presented at once.

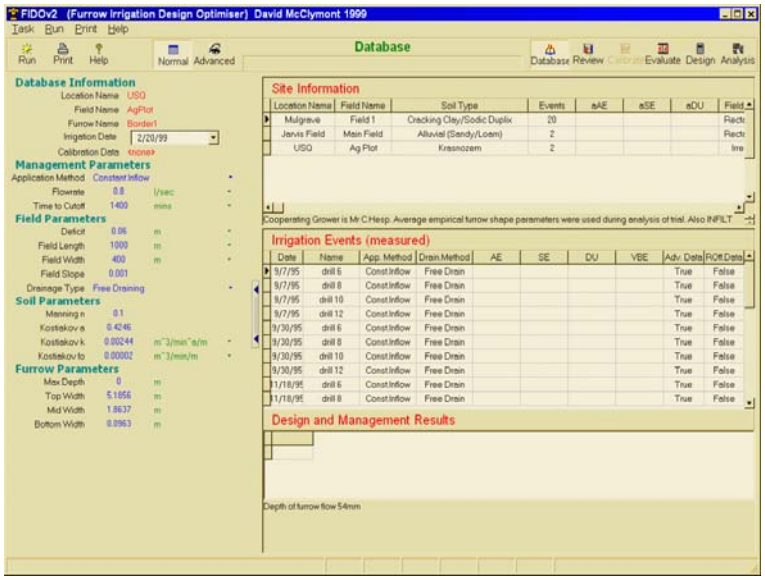


Figure A7.7.2: Updated FIDO Database. An updated version of the database interface with an extra table for design and management results.

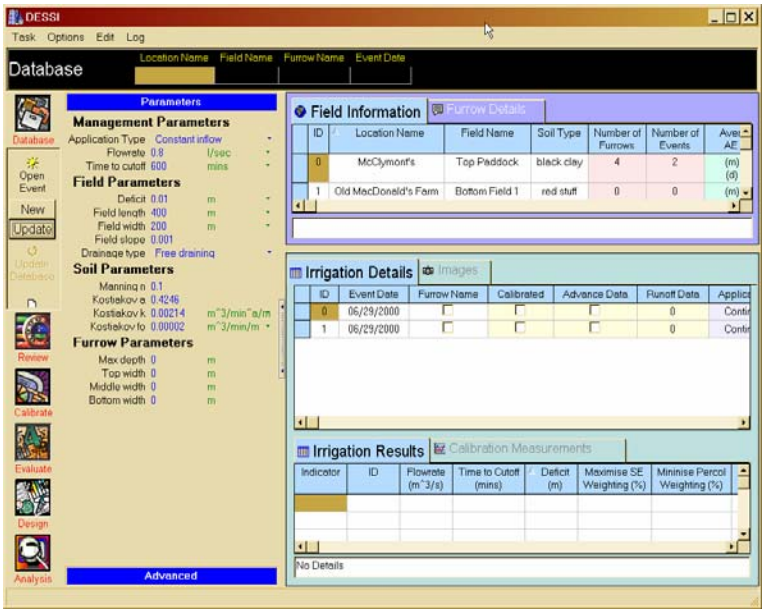


Figure A7.7.3: Tab filtering in early FIDO database. This version tries to simplify the database interface by using tabs to filter field information, furrow details, irrigation details, irrigation results and calibration measurements. This was ultimately seen as a failed attempt at simplification as the end result was just as complex as the previous versions.

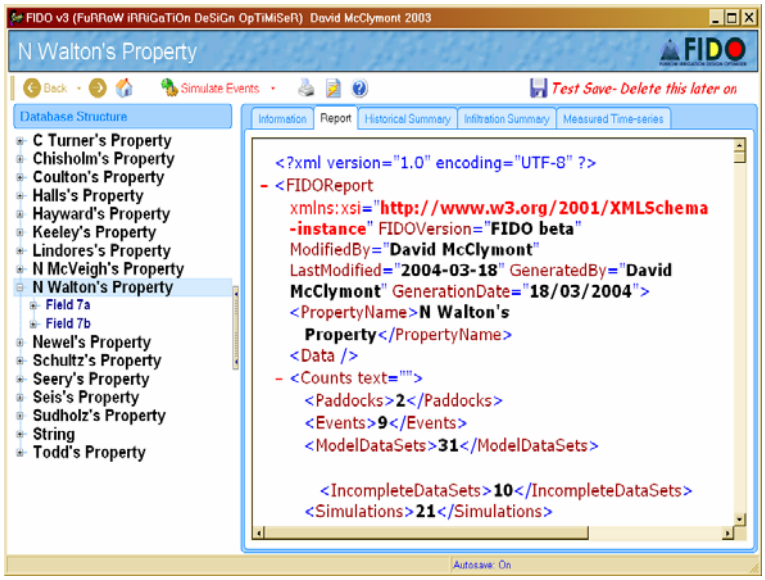


Figure A7.7.4: Prototype of current database. This is a prototype of the current version of *FIDO* using the *XML*-based database structure. This structure has four levels including property, paddock, event and simulation information. The database output presented in the right hand window is in raw *XML* format with no stylesheet transformation applied.

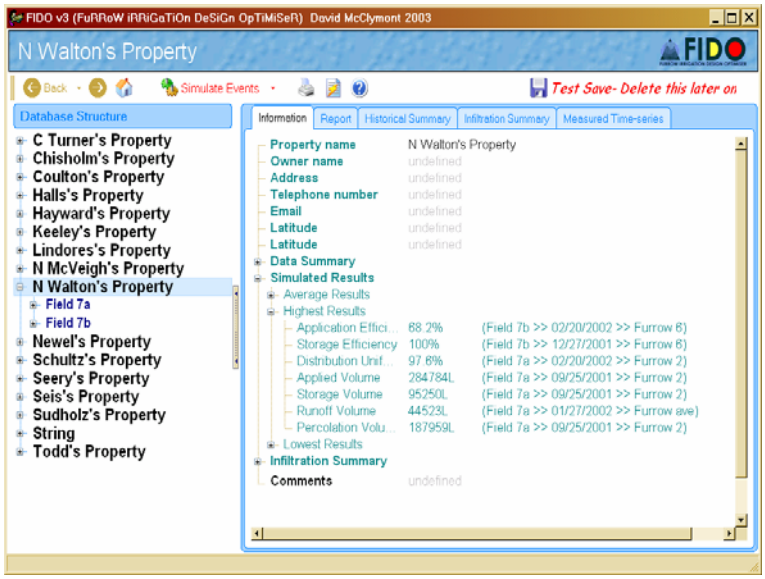


Figure A7.7.5: Prototype data editor. Prototype data editor window for the current *FIDO* version. The *XML* data cannot be directly edited so a tree based data editor was incorporated into this prototype version. The same system exists in the current version but with more refined formatting.

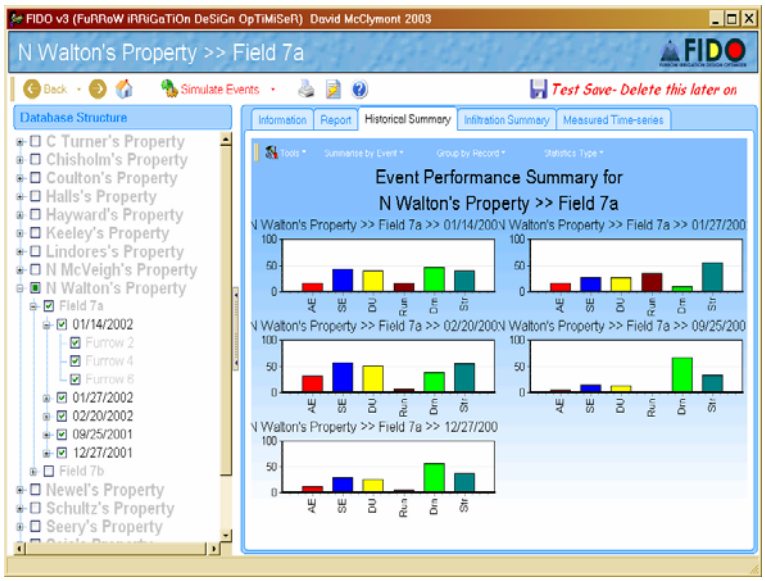


Figure A7.7.6: Database performance summaries. Prototype summary analysis for performance results. This analysis provides a range of options to investigate spatial and temporal performance of the irrigation.

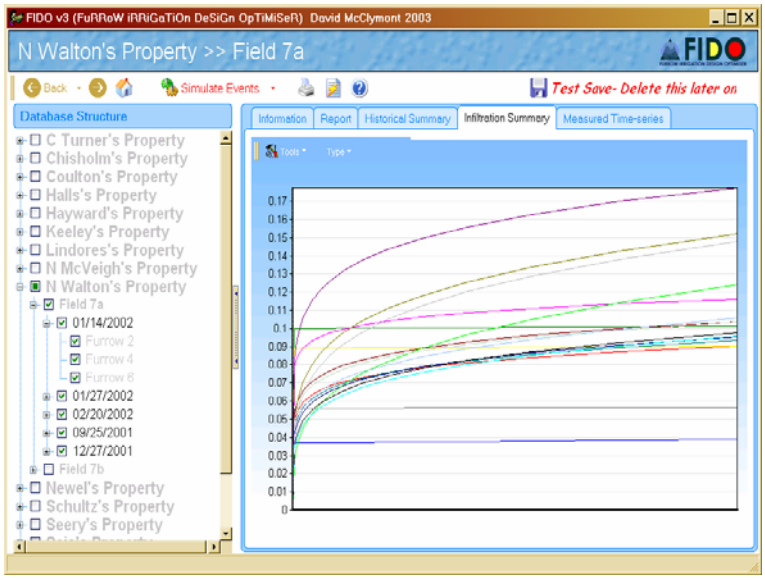


Figure A7.7.7: Infiltration summaries. Infiltration summary analysis for investigating spatial and temporal variations in infiltration.