Approximation of Function and Its Derivatives Using Radial Basis Function Networks

Nam Mai-Duy and Thanh Tran-Cong*

Faculty of Engineering and Surveying,

University of Southern Queensland, Toowoomba, QLD 4350, Australia

Submitted to *Applied Mathematical Modelling*, December 2000; revised August 2002

Abstract. This paper presents a numerical approach, based on Radial Basis Function Networks (RBFNs), for the approximation of a function and its derivatives (scattered data interpolation). The approach proposed here is called the indirect radial basis function network (IRBFN) approximation which is compared with the usual direct approach. In the direct method (DRBFN) the closed form RBFN approximating function is first obtained from a set of training points and the derivative functions are then calculated directly by differentiating such closed form RBFN. In the indirect method (IRBFN) the formulation of the problem starts with the decomposition of the derivative of the function into RBFs.

*Corresponding author: Telephone +61 7 46312539, Fax +61 7 46 312526, E-mail trancong@usq.edu.au

The derivative expression is then integrated to yield an expression for the original function, which is then solved via the general linear least squares principle, given an appropriate set of discrete data points. The IRBFN method allows the filtering of noise arisen from the interpolation of the original function from a discrete set of data points and produces a greatly improved approximation of its derivatives. In both cases the input data consists of a set of unstructured discrete data points (function values), which eliminates the need for a discretisation of the domain into a number of finite elements (FE). The results obtained are compared with those obtained by the Feed Forward Neural Network (FFNN) approach where appropriate and the "Finite Element" methods. In all examples considered, the IRBFN approach yields a superior accuracy. For example, all partial derivatives up to second order of the function of three variables $y = x_1^2 + x_1x_2 - 2x_2^2 - x_2x_3 + x_3^2$ are approximated with at least an order of magnitude better in the L_2-norm in comparison with the usual DRBFN approach.

Keywords: Radial basis function networks, function approximation, derivative approximation, scattered data interpolation, global approximation.

1 Introduction

Numerical methods for differentiation are of significant interest and importance in the study of numerical solutions of many problems in engineering and science. For example, the approximation of derivatives is needed either to convert the relevant governing equations into a discrete form or to numerically estimate various terms from a set of discrete or scattered data. This is commonly achieved by discretising the domain of analysis into a number of elements which are defined by a small number of nodes. The interpolation of a function and its derivatives over such an element from the nodal values can then be achieved analytically via the chosen shape functions. Examples of elements include finite elements (FE) and boundary elements (BE) associated with the Finite Element Method (FEM) (e.g. [1]) and the Boundary Element Method (BEM) (e.g. [2]). Element-based methods are referred to as conventional methods in this paper. Common shape functions for one, two and three-dimensional elements can be found in most texts on Finite Element Method and Boundary Element Method (e.g. [1-2]). However, the element technique requires mesh generation which is time consuming and therefore accounts for a high proportion of the analysis cost, especially for problems with moving or unknown boundaries. For practical analysis, automatic discretisation or meshing is a highly desirable feature but rarely available in general. Thus there are great interests in element-free numerical methods in both engineering and scientific communities. In particular, neural networks have been developed and become one of the main fields of research in numerical analysis. Radial Basis Function Networks (RBFNs) [3-11] can be used for a wide range of applications primarily because it can approximate any regular function [6,7,10] and its training

is faster than that of a multilayer perceptron when the RBFN combines self-organised and supervised learning [11]. The design of an RBFN is considered as a curve-fitting (approximation) problem in a high-dimensional space. Correspondingly, the generalization of the approach is equivalent to the use of a multidimensional surface to interpolate the test data [9]. The networks just need an unstructured distribution of collocation points throughout a volume for the approximation and hence the need for discretisation of the volume of the analysis domain is eliminated. In this paper new approximation methods based on RBFNs are reported. The primary aim of the presented methods is the achievement of a more accurate approximation of a target function's derivatives. From the results obtained here it is suggested that the present IRBFN approach could be, in addition to its ability to approximate scattered data, a potential candidate for future development of element-free methods for engineering modelling and analyses. The paper is organized as follows. In section 2, the problem is defined. A brief review of RBFNs is given in section 3 and then, in sections 4 and 5, a direct RBFN (DRBFN) and an indirect RBFN (IRBFN) method for the approximation of a function and its derivatives are discussed. The DRBFN method is included to provide the basis for the assessment of the presently proposed IRBFN approach. Both methods are illustrated with the aid of three numerical examples of function of one, two and three variables in section 6. Section 7 concludes the paper.

2 Description of Problem

The problem considered in this paper is described as follows (superscripts are used to index elements of a set of neurons and subscripts denote scalar components of a p-dimensional vector):

- Given a set of data points whose elements consist of paired values of the independent variables (a vector $\boldsymbol{x}$) and the dependent variable (a scalar y), denoted by $\{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^{n}$ where n is the number of input points and $\boldsymbol{x} = [x_1, x_2, ..., x_p]^T$ where p is the number of dimensions and the superscript ^{T} denotes the transpose operation,
- find a closed form approximate function f of the dependent variable y and its closed form approximate derivative functions.

3 Function Approximation by RBFNs

An RBFN represents a map from the p-dimensional input space to the 1-dimensional output space $f : R^p \rightarrow R^1$ that consists of a set of weights $\{w^{(i)}\}_{i=1}^{m}$, and a set of radial basis functions $\{g^{(i)}\}_{i=1}^{m}$ where $m \leq n$. There is a large class of radial basis functions which can be written in a general form $g^{(i)}(\boldsymbol{x}) = \phi^{(i)}(\|\boldsymbol{x} - \boldsymbol{c}^{(i)}\|)$, where $\|.\|$ denotes the Euclidean norm and $\{\boldsymbol{c}^{(i)}\}_{i=1}^{m}$ is a set of the centers that can be chosen from among the data points. The following are some common types of radial basis functions that are of particular interest in the study of RBFNs [9]:

1. multiquadrics

$$\phi^{(i)}(r) = \phi^{(i)}(\|\boldsymbol{x} - \boldsymbol{c}^{(i)}\|) = \sqrt{r^2 + a^{(i)2}} \qquad \text{for some } a^{(i)} > 0, \tag{1}$$

2. inverse multiquadrics

$$\phi^{(i)}(r) = \phi^{(i)}(\|\boldsymbol{x} - \boldsymbol{c}^{(i)}\|) = \frac{1}{\sqrt{r^2 + a^{(i)2}}} \qquad \text{for some } a^{(i)} > 0, \tag{2}$$

3. Gaussians

$$\phi^{(i)}(r) = \phi^{(i)}(\|\boldsymbol{x} - \boldsymbol{c}^{(i)}\|) = \exp\left(-\frac{r^2}{a^{(i)2}}\right) \qquad \text{for some } a^{(i)} > 0, \tag{3}$$

where $a^{(i)}$ is usually referred to as the width of the ith basis function and $r = \|\boldsymbol{x} - \boldsymbol{c}^{(i)}\| = \sqrt{(\boldsymbol{x} - \boldsymbol{c}^{(i)}) \cdot (\boldsymbol{x} - \boldsymbol{c}^{(i)})}$.

The inverse multiquadrics (2) and Gaussians function (3) have a local response, i.e. they decrease monotonically with increasing distance from the center (localized function). In contrast, the multiquadrics (1) increases with increasing distance from the center and therefore exhibits a global response (non-localized function). An important property of the RBFN is that it is a linearly weighted network in the sense that the output is a linear combination of m radial basis functions written as

$$f(\boldsymbol{x}) = \sum_{i=1}^{m} w^{(i)} g^{(i)}(\boldsymbol{x}). \tag{4}$$

With the model f constructed as a linear combination of m fixed functions in a given family, the problem is to find the unknown weights $\{w^{(i)}\}_{i=1}^{m}$. For this purpose, the general least squares principle is used to minimise the sum squared error

$$SSE = \sum_{i=1}^{n} \left[y^{(i)} - f(\boldsymbol{x}^{(i)}) \right]^2, \tag{5}$$

with respect to the weights of f, resulting in a set of m simultaneous linear algebraic equations (normal equations) in the m unknown weights

$$(\boldsymbol{G}^T \boldsymbol{G})\boldsymbol{w} = \boldsymbol{G}^T \boldsymbol{y}, \tag{6}$$

where

$$\boldsymbol{G} = \begin{bmatrix} g^{(1)}(\boldsymbol{x}^{(1)}) & g^{(2)}(\boldsymbol{x}^{(1)}) & \cdots & g^{(m)}(\boldsymbol{x}^{(1)}) \\ g^{(1)}(\boldsymbol{x}^{(2)}) & g^{(2)}(\boldsymbol{x}^{(2)}) & \cdots & g^{(m)}(\boldsymbol{x}^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ g^{(1)}(\boldsymbol{x}^{(n)}) & g^{(2)}(\boldsymbol{x}^{(n)}) & \cdots & g^{(m)}(\boldsymbol{x}^{(n)}) \end{bmatrix},$$

$$\boldsymbol{w} = [w^{(1)}, w^{(2)}, ..., w^{(m)}]^T,$$

$$\boldsymbol{y} = [y^{(1)}, y^{(2)}, ..., y^{(n)}]^T.$$

However, in the special case where $n = m$, the resultant system is just

$$\boldsymbol{G}\boldsymbol{w} = \boldsymbol{y}. \tag{7}$$

4 Function derivatives by direct RBFN method

In an arbitrary RBFN where the basis functions are fixed and the weights are adaptable, the derivative of the function computed by the network is also a linear combination of fixed functions (the derivatives of the radial basis functions). The partial derivatives of the approximate function $f(\boldsymbol{x})$ (4) can be calculated as follows

$$\frac{\partial^k f}{\partial x_j ... \partial x_l} = f_{,j...l}(\boldsymbol{x}) = \sum_{i=1}^{m} w^{(i)} \frac{\partial^k g^{(i)}}{\partial x_j ... \partial x_l}, \tag{8}$$

where $\frac{\partial^k g^{(i)}}{\partial x_j ... \partial x_l}$ is the corresponding basis function for the derivative function $f_{,j...l}(\boldsymbol{x})$, which is obtained by differentiating the original basis function $g^{(i)}(\boldsymbol{x})$ which is continuously differentiable. For example, considering the first order derivative of function $f(\boldsymbol{x})$ with respect to x_j, denoted by $f_{,j}$, the corresponding basis functions are found analytically as follows:

1. for multiquadrics

$$h^{(i)}(\boldsymbol{x}) = \frac{\partial g^{(i)}}{\partial x_j} = \frac{x_j - c_j^{(i)}}{(r^2 + a^{(i)2})^{0.5}}, \tag{9}$$

2. for inverse multiquadrics

$$h^{(i)}(\boldsymbol{x}) = \frac{\partial g^{(i)}}{\partial x_j} = -\frac{x_j - c_j^{(i)}}{(r^2 + a^{(i)2})^{1.5}}, \tag{10}$$

3. for Gaussians

$$h^{(i)}(\boldsymbol{x}) = \frac{\partial g^{(i)}}{\partial x_j} = \frac{-2(x_j - c_j^{(i)})}{a^{(i)2}} \exp\left(-\frac{r^2}{a^{(i)2}}\right). \tag{11}$$

Considering the second order derivative of function $f(\boldsymbol{x})$ with respect to x_j, denoted by $f_{,jj}$, the corresponding basis functions will be

1. for multiquadrics

$$\bar{h}^{(i)}(\boldsymbol{x}) = \frac{\partial h^{(i)}}{\partial x_j} = \frac{r^2 + a^{(i)2} - (x_j - c_j^{(i)})^2}{(r^2 + a^{(i)2})^{1.5}}, \tag{12}$$

2. for inverse multiquadrics

$$\bar{h}^{(i)}(\boldsymbol{x}) = \frac{\partial h^{(i)}}{\partial x_j} = \frac{3(x_j - c_j^{(i)})^2}{(r^2 + a^{(i)2})^{2.5}} - \frac{1}{(r^2 + a^{(i)2})^{1.5}}, \tag{13}$$

3. for Gaussians

$$\bar{h}^{(i)}(\boldsymbol{x}) = \frac{\partial h^{(i)}}{\partial x_j} = \frac{2}{a^{(i)2}} \left[\frac{2}{a^{(i)2}}(x_j - c_j^{(i)})^2 - 1\right] \exp\left(-\frac{r^2}{a^{(i)2}}\right). \tag{14}$$

Similarly, the basis functions for $f_{,kj}$ are as follows:

1. for multiquadrics

$$\bar{h}^{(i)}(\boldsymbol{x}) = \frac{\partial h^{(i)}}{\partial x_k} = -\frac{(x_j - c_j^{(i)})(x_k - c_k^{(i)})}{(r^2 + a^{(i)2})^{1.5}}, \tag{15}$$

2. for inverse multiquadrics

$$\bar{h}^{(i)}(\boldsymbol{x}) = \frac{\partial h^{(i)}}{\partial x_k} = \frac{3(x_j - c_j^{(i)})(x_k - c_k^{(i)})}{(r^2 + a^{(i)2})^{2.5}}, \tag{16}$$

3. for Gaussians

$$\bar{h}^{(i)}(\boldsymbol{x}) = \frac{\partial h^{(i)}}{\partial x_k} = \frac{4(x_j - c_j^{(i)})(x_k - c_k^{(i)})}{a^{(i)4}} \exp\left(-\frac{r^2}{a^{(i)2}}\right). \tag{17}$$

Once $f(\boldsymbol{x})$ is determined by solving (6) or (7) for the unknown weights, which is referred to as network training, it is straightforward and economical to compute its derivatives according to (8). However, this direct method has some drawbacks that are illustrated in the following example.

4.1 Example to illustrate the drawbacks of the DRBFN method

The function

$$y(x) = x^3 + x + 0.5, \quad -3 \leq x \leq 2,$$

is sampled at 50 uniformly spaced training points as depicted in Figure 1. Parameters to be decided before the start of network training are the number of centers m, their locations $\{\boldsymbol{c}^{(i)}\}_{i=1}^m$ and a set of the corresponding widths $\{a^{(i)}\}_{i=1}^m$. The ideal data points used here are not corrupted by noise. According to Cover's Theorem [9], the more basis functions are used, the better the approximation will be and so all data points will be taken to be the centers of the network ($m = n$) in this study. Thus $\{\boldsymbol{c}^{(i)} = \boldsymbol{x}^{(i)}\}_{i=1}^n$. The

width of the ith basis function is determined according to the following relation [11]

$$a^{(i)} = \beta d^{(i)}, \tag{18}$$

where β is a factor, $\beta > 0$, and $d^{(i)}$ is the distance from the ith center to the nearest neighbouring center. As a measure of the accuracy of different approximate schemes, a norm of the error of the solution, N_e, is defined as

$$N_e = \sqrt{\sum_{i=1}^{nt} (y^{(i)} - f^{(i)})^2}, \tag{19}$$

where $f^{(i)}$ and $y^{(i)}$ are the calculated and exact function values at the point i, and nt is the total number of test nodes. Smaller N_es indicate more accurate approximations.

Table 1 shows the error norms N_es of the approximate function and its first and second derivatives that are obtained from the networks using different types of radial basis function based on a set of 250 test nodes. It can be seen that errors in the approximate function obtained from all networks are quite low and hence the global shape of the original function is well captured as shown in Figure 1. However, the derivative functions, especially higher order ones, are strongly influenced by the local behaviour of the approximant. The nature of a bad local behaviour despite a good global approximation is illustrated in Figure 2a. The errors in the function approximation amplify in the process of differentiation as shown in Figures 2b-c with the corresponding error norms N_es shown in Table 1. It is remarkable that multiquadrics RBFs (1) produce greater accuracy than other basis functions. This surprising result was discussed by Franke [12] and Powell [5].

However, the norms N_es of the derivative functions estimated using multiquadrics RBFs are still quite high (Table 1). To improve accuracy, a new indirect method is proposed and presented in the next section.

5 Function derivatives by indirect RBFN method

It can be seen that the differentiation process is very sensitive to even a small level of noise as illustrated in the previous section. In contrast it is expected that on average the integration process is much less sensitive to noise. Based on this observation, it is proposed here that the approximation procedure starts with the derivative function using RBFNs. The original function is then obtained by integration. Here the generic nature of "derivative function" and "original function" is illustrated as follows. Suppose a function $f(x)$ and its derivatives $f'(x)$ and $f''(x)$ are to be approximated. The procedure consists of two stages. In the first stage, $f(x)$ corresponds to the "original function" and $f'(x)$ the "derivative function". In the second stage the $f'(x)$ obtained in stage 1 corresponds to the "original function" and $f''(x)$ the "derivative function". The procedure just discussed is here referred to as the first indirect method or IRBFN1. Alternatively, the procedure can start with the second derivative. First, the second order derivative is approximated by a RBFN, then the first order derivative is obtained by integration. Finally the original function is similarly obtained, i.e. by integrating the first derivative function. This second method is here referred to as the second indirect method or IRBFN2. The detail of IRBFN1 and IRBFN2 is described in the next two sections for function of one and two or more variables respectively, followed by some numerical results in section 6.

5.1 Functions of one variable

5.1.1 IRBFN1 method

In this method, the first order derivative function is decomposed into radial basis functions as

$$f'(x) = \sum_{i=1}^{m} w^{(i)} g^{(i)}(x), \tag{20}$$

where $\{g^{(i)}(x)\}_{i=1}^{m}$ is a set of radial basis functions and $\{w^{(i)}\}_{i=1}^{m}$ is the set of corresponding weights. With this approximation, the original function can be calculated as

$$f(x) = \int f'(x)dx = \int \sum_{i=1}^{m} w^{(i)} g^{(i)}(x)dx = \sum_{i=1}^{m} w^{(i)} \int g^{(i)}(x)dx = \sum_{i=1}^{m} w^{(i)} H^{(i)}(x) + C_1, \tag{21}$$

where C_1 is the constant of integration and $\{H^{(i)}(x)\}_{i=1}^{m}$ is the set of corresponding basis functions for the original function with $H^{(i)}(x) = \int g^{(i)}(x)dx$. The radial basis functions $\{g^{(i)}\}_{i=1}^{m}$ are continuously integrable, but only two basis functions $\{H^{(i)}(x)\}_{i=1}^{m}$ corresponding to the multiquadric (1) and the inverse multiquadric (2) are able to be obtained analytically here. This paper focuses on the use of these two RBFs in the indirect method. The corresponding basis functions are:

1. for multiquadrics

$$H^{(i)}(x) = \frac{(x - c^{(i)})\sqrt{(x - c^{(i)})^2 + a^{(i)2}}}{2} + \frac{a^{(i)2}}{2} \ln\left((x - c^{(i)}) + \sqrt{(x - c^{(i)})^2 + a^{(i)2}}\right), \tag{22}$$

2. for inverse multiquadrics

$$H^{(i)}(x) = \ln\left((x - c^{(i)}) + \sqrt{(x - c^{(i)})^2 + a^{(i)2}}\right). \tag{23}$$

The training to determine the weights in (20) and (21) is equivalent to a minimisation of the following sum squared error

$$SSE = \sum_{i=1}^{n} \left[y^{(i)} - f(x^{(i)})\right]^2. \tag{24}$$

Equation (21) is used in equation (24) in the minimisation procedure, which results in a system of equations in terms of the unknown weights $w^{(i)}$. The data used in training the network for the derivative and original functions just consists of a set of discrete values $\{y^{(i)}\}_{i=1}^{n}$ of the dependent variable y and the closed form of the derivative function (20). The minimisation of (24) can be achieved by solving the corresponding normal equations [13]. However in practice the normal equations method of solution can produce less than optimum solution, i.e. the norm of the solution (in the least square sense) is not the smallest. Fortunately, Singular Value Decomposition (SVD) method [13] can overcome this difficulty and will be used to solve (24) for the unknown weights and the constant of integration in the remainder of this paper. The SVD method provides a solution whose

norm is the smallest in the least-squares sense, i.e. any combination of basis functions irrelevant to the fit is driven down to a small value. After solving (24), a set of the weights is obtained and used for approximating the derivative function via (20) and together with the constant C_1 for estimating the original function via (21). The example in section 4.1 is reconsidered here using the IRBFN1 method. The N_es over a set of 250 test nodes are decreased considerably as shown in Tables 2 - 4. There is a significant improvement in the results obtained by the IRBFN1 over those obtained by the DRBFN not only for the derivative functions but also for the original function. The improvement factor is defined as follows

$$\text{Improvement factor} = \frac{\text{DRBFN } N_e}{\text{IRBFN1 } N_e}. \tag{25}$$

The improvement factors are 102.6, 85.9 and 93.6 corresponding to the original, 1st derivative and 2nd derivative functions respectively when the multiquadric is used and 49.4, 40.6 and 44.7 when the inverse multiquadric is used (Tables 2-4).

5.1.2 IRBFN2 method

As an alternative indirect method for approximating function and its derivatives, the second order derivative function $f''(x)$ is first approximated in terms of radial basis functions as follows

$$f''(x) = \sum_{i=1}^{m} w^{(i)} g^{(i)}(x). \tag{26}$$

Then the first derivative function $f'(x)$ is given by (21) as

$$f'(x) = \int f''(x)dx = \sum_{i=1}^{m} w^{(i)} H^{(i)}(x) + C_1, \tag{27}$$

with the basis functions given by (22) or (23). The original function is calculated as

$$f(x) = \int f'(x)dx = \sum_{i=1}^{m} w^{(i)} \bar{H}^{(i)}(x) + C_1 x + C_2, \tag{28}$$

where C_1 and C_2 are constants of integration and the corresponding basis functions are obtained by integrating (22) or (23) as shown below

1. for multiquadrics

$$\bar{H}^{(i)}(x) = \int H^{(i)}(x)dx = \frac{((x-c^{(i)})^2 + a^{(i)2})^{1.5}}{6} + \frac{a^{(i)2}}{2}(x-c^{(i)}) \ln\left((x-c^{(i)}) + \sqrt{(x-c^{(i)})^2 + a^{(i)2}}\right) - \frac{a^{(i)2}}{2}\sqrt{(x-c^{(i)})^2 + a^{(i)2}}, \tag{29}$$

2. for inverse multiquadrics

$$\bar{H}^{(i)}(x) = \int H^{(i)}(x)dx = (x-c^{(i)}) \ln\left((x-c^{(i)}) + \sqrt{(x-c^{(i)})^2 + a^{(i)2}}\right) - \sqrt{(x-c^{(i)})^2 + a^{(i)2}}. \tag{30}$$

In the present IRBFN2 method, the improvement factors have increased (Tables 5-6) for both the original function and its derivatives in comparison with the first indirect method IRBFN1. It is remarkable here that the improvement in the case of multiquadrics is very

significant for all approximate functions (more than 16 times). Thus, the multiquadric function maintains its superior performance in terms of accuracy among the radial basis functions used in IRBFN2.

5.1.3 The role of "constants" of integration

"Constants" of integration in equations (21) and (28) appear naturally in the present indirect formulation. The structure of the approximant therefore looks like

$$f(x) = \sum_{i=1}^{m} w^{(i)} \bar{H}^{(i)}(x) + \text{polynomial.} \tag{31}$$

As a result, if $y(x)$ is flat or closer to a polynomial fit, the above structure (31) has the ability for better accuracy. This is in addition to the inherent smoothing of error in the process of integration.

5.2 Functions of two or more variables

In this section the indirect methods discussed in section 5.1 are extended to the case of functions of many variables. The case of functions of two variables is discussed in detail and the procedure for functions of three or more variables can be similarly developed.

5.2.1 IRBFN1 method

Consider the approximation of a function of two variables $f(x_1, x_2)$. In the IRBFN1 method, the first order partial derivative of $f(x_1, x_2)$ with respect to x_1, denoted by $f_{,1}$, is first approximated in terms of radial basis functions

$$f_{,1}(x_1, x_2) = \sum_{i=1}^{m} w^{(i)} g^{(i)}(x_1, x_2), \tag{32}$$

where $\{g^{(i)}(x_1, x_2)\}_{i=1}^{m}$ is a set of radial basis functions and $\{w^{(i)}\}_{i=1}^{m}$ is the set of corresponding weights.

The original function can be calculated as

$$\begin{aligned} f(x_1, x_2) = \int f_{,1}(x_1, x_2) dx_1 &= \int \sum_{i=1}^{m} w^{(i)} g^{(i)}(x_1, x_2) dx_1 \\ &= \sum_{i=1}^{m} w^{(i)} \int g^{(i)}(x_1, x_2) dx_1 = \sum_{i=1}^{m} w^{(i)} H^{(i)}(x_1, x_2) + C_1(x_2), \end{aligned} \tag{33}$$

where $C_1(x_2)$ is a function of the variable x_2 and $\{H^{(i)}(x_1, x_2)\}_{i=1}^{m}$ is the set of corresponding basis functions for the original function and given below

1. for multiquadrics

$$\begin{aligned} H^{(i)}(x_1, x_2) = &\frac{(x_1 - c_1^{(i)})\sqrt{r^2 + a^{(i)2}}}{2} \\ &+ \frac{r^2 - (x_1 - c_1^{(i)})^2 + a^{(i)2}}{2} \ln\left((x_1 - c_1^{(i)}) + \sqrt{r^2 + a^{(i)2}}\right), \end{aligned} \tag{34}$$

2. for inverse multiquadrics

$$H^{(i)}(x_1, x_2) = \ln\left((x_1 - c_1^{(i)}) + \sqrt{r^2 + a^{(i)2}}\right). \tag{35}$$

The added term on the right hand side of (33) is a function of the variable x_2 only. Thus $C_1(x_2)$ can be interpolated using the IRBFN2 method for univariate functions as follows (in the previous section, IRBFN2 is shown to be the better alternative among the methods investigated in this work.)

$$C_1''(x_2) = \sum_{i=1}^{M} \bar{w}^{(i)} g^{(i)}(x_2), \tag{36}$$

$$C_1'(x_2) = \sum_{i=1}^{M} \bar{w}^{(i)} H^{(i)}(x_2) + \widehat{C}_1, \tag{37}$$

$$C_1(x_2) = \sum_{i=1}^{M} \bar{w}^{(i)} \bar{H}^{(i)}(x_2) + \widehat{C}_1 x_2 + \widehat{C}_2, \tag{38}$$

where $\widehat{C}_1$ and $\widehat{C}_2$ are constants of integration; $\bar{w}^{(i)}$ are the corresponding weights; and M is the number of centres whose x_2 coordinates are distinct. Upon applying the general linear least squares principle, a system of linear algebraic equations is obtained. The unknown of the system which is found by the SVD method as mentioned earlier, consists of the set of weights in (32), the second set of weights in (36) and the constants of integration $\widehat{C}_1$, $\widehat{C}_2$. The strategy of approximation is the same for the derivative function of $f(x_1, x_2)$ with respect to the variable x_2 ($f_{,2}(x_1, x_2)$).

5.2.2 IRBFN2 method

In this method, the second order derivative functions are first approximated in terms of radial basis functions. For example, in the case of $f_{,11}$ the basis functions for the first derivative function, $f_{,1}$, are given by (34) or (35) while for the original function f, the basis functions are obtained by integrating (34) or (35) and shown below

1. for multiquadrics

$$\bar{H}^{(i)}(x_1, x_2) = \int H^{(i)}(x_1, x_2)dx_1 = \frac{(r^2 + a^{(i)2})^{1.5}}{6} + \frac{r^2 - (x_1 - c_1^{(i)})^2 + a^{(i)2}}{2}(x_1 - c_1)\ln\left((x_1 - c_1^{(i)}) + \sqrt{r^2 + a^{(i)2}}\right) - \frac{r^2 - (x_j - c_j^{(i)})^2 + a^{(i)2}}{2}\sqrt{r^2 + a^{(i)2}}, \quad (39)$$

2. for inverse multiquadrics

$$\bar{H}^{(i)}(x_1, x_2) = \int H^{(i)}(x_1, x_2)dx_1 = (x_1 - c_1)\ln\left((x_1 - c_1^{(i)}) + \sqrt{r^2 + a^{(i)2}}\right) - \sqrt{r^2 + a^{(i)2}}. \quad (40)$$

The original function is calculated as

$$f(x_1, x_2) = \sum_{i=1}^{m} w^{(i)}\overline{H}^{(i)}(x_1, x_2) + C_1(x_2)x_1 + C_2(x_2), \quad (41)$$

where $C_1(x_2)$ and $C_2(x_2)$ are constants of integration which are interpolated in the same maner as shown by (36)-(38).

For the purpose of illustration, some numerical results are presented in the next section.

6 Numerical results

In this section, examples of approximation of functions of one, two and three variables are given. As mentioned, the multiquadric function appears to be the better one in terms of accuracy among the basis functions considered and will be used to solve the example problems. The factor β that influences the accuracy of the solution is just chosen to be 2.0 until now. In the following examples, for the purpose of investigation of its effect, β will take values over a wide range with an increment of 0.2. From the numerical experiments discussed shortly, it appears that there is an upper limit for β above which the system of equations (6) or (7) is ill-conditioned, which is also observed by Tarwater [14]. In the present work, the value of β is considered to reach an upper limit when the condition of the system matrix is $O(10^{17})$, i.e. the estimate for the reciprocal of the condition of the matrix in 1-norm using LINPACK condition estimator [15] is of $O(10^{-17})$.

6.1 Example 1

Consider the following function of one variable

$$y = 0.02(12 + 3x - 3.5x^2 + 7.2x^3)(1 + \cos 4\pi x)(1 + 0.8 \sin 3\pi x),$$

with $0 \leq x \leq 1$, a problem studied by Hashem and Schmeiser [16]. They reported a method, namely Mean Squared Error-Optimal Linear Combinations of Trained Feedforward Neural Netwoks (MSE-OLC), for an approximation of the function and its derivatives. The authors suggest that the usual approach is to try a multiple of networks with possibly different structures and values for training parameters and the "best" network (based on some optimality criterion) is selected. Instead of the usual approach just described the authors investigated a new approach where a combination of the trained networks is constructed by forming the weighted sum of the corresponding outputs of the trained networks. The authors claim that their MSE-OLC method yields more accurate approximations in comparison with the best trained FFNN. For this problem, with a set of 200 training nodes and $10,000$ test nodes, the resultant MSEs for the original, the first and second order derivative functions produced by MSE-OLC are 0.000017, 0.1 and 133.3, respectively, which are 87.3%, 69.7% and 64.5%, respectively, less than the MSEs produced by the best FFNN [16]. Here, both the direct and indirect RBFN methods are applied to solve this problem using 200 training points and 10,000 test nodes, uniformly spaced along the x-axis. The training points are displayed in Figure 4a. In contrast, Hashem and Schmeiser [16] used the same number of data points but randomly distributed. In order to compare the present results with those obtained by Hashem and Schmeiser [16] the latter's MSEs are converted into norms N_es as defined in this paper. Thus the norms N_es corresponding to the original, first derivative and second derivative are 0.41, 31.62 and 1154.56, respectively. Figure 3 compares the quality of approximation obtained by the DRBFN, IRBFN1, IRBFN2, MSE-OLC and the conventional element method (with linear element) in the range $0.2 \leq \beta \leq 9.0$ and indicates that the quality

of approximation improves significantly with RBFN, and particularly that the IRBFN1 yields superior results over the whole range of values of β (e.g. with $\beta = 9$ the N_es for the second derivative are 0.0877 (IRBFN1), 5.6869 (DRBFN), 209.89 (conventional) and 1154.56 (MSE-OLC) [16]). Even more accurate results can be obtained by using the second indirect method IRBFN2 ($N_e = 0.0517$), as shown in the same Figure 3. Figure 4 shows the plots of the function and its derivative at $\beta = 0.2$ obtained with the IRBFN2 where the "worst" value of β is used to demonstrate the superior performance of the IRBFN2.

6.2 Example 2

Consider the following bivariate function

$$y = x_1^2 x_2 + x_2^3/3 + x_2^2/2,$$

where $-3 \leq x_1 \leq 3$ and $-3 \leq x_2 \leq 3$. This is a non-trivial example which has a complicated root structure [17]. The data consist of 441 points, uniformly spaced along both axes x_1 and x_2 for training and 1764 points for testing. The results obtained from both DRBFN and IRBFN methods are compared with the accuracies achieved by the conventional method using linear shape function over triangular elements. Figure 5 shows the quality of the approximation of the function $f(x_1, x_2)$ and its first derivatives while Figure 6 shows the quality of the approximation of second order derivatives using the DRBFN, IRBFN1 and IRBFN2 with β in the range $0.2 \leq \beta \leq 7.0$. Again, the results

are more accurate with IRBFN2 as shown in Figures 5-6. Thus it can be seen that the IRBFN2 yields better performance than the IRBFN1 which in turn performs better than the DRBFN.

6.3 Example 3

Consider the following function of three variables

$$y = x_1^2 + x_1 x_2 - 2x_2^2 - x_2 x_3 + x_3^2,$$

where $0 \leq x_1 \leq 0.5$, $0 \leq x_2 \leq 0.5$ and $0 \leq x_3 \leq 0.5$. In this example, 216 points, uniformly spaced along the axes x_1, x_2 and x_3, are used for training and 1728 points for testing. Figure 7 shows the quality of the approximation of the original function. Figure 8 shows the plots of norm N_e as function of β ($0.2 \leq \beta \leq 7.6$) for first order derivative functions $f_{,1}$, $f_{,2}$ and $f_{,3}$, while Figures 9-10 are for second order derivative functions $f_{,11}$, $f_{,22}$, $f_{,33}$, $f_{,12}$, $f_{,23}$ and $f_{,31}$. Figures 7-10 again show that the IRBFN2 method exhibits superior performance over other methods.

7 Concluding Remarks

This paper reports the successful development and implementation of function approximation methods based on RBFNs for functions of one, two and three variables and their derivatives (scattered data interpolation). Both the direct RBFN method and the indi-

rect RBFN method are able to offer better results in comparison with the conventional method using linear shape functions. The present RBFN methods also eliminate the need for FE-type discretisation of the domain of analysis. Among the RBFs considered, multiquadrics RBF offers the best performance in accuracy in both DRBFN and IRBFN method. Numerical results show that IRBFNs, especially IRBFN2, achieve greater accuracy than DRBFN in the approximation of both function and especially its derivatives. Furthermore, this superior accuracy is maintained over a wide range of RBF's width ($0.2 < \beta < 9$). A formal theoretical proof of the superior accuracy of the present IRBFN method cannot be offered at this stage, at least by the present authors. However, a heuristic argument can be presented as follows. In the direct methods, the starting point is the decomposition of the unknown functions into some finite basis and all derivatives are obtained as a consequence. Any inaccuracy in the assumed decomposition is usually magnified in the process of differentiation. In contrast, in the indirect approach the starting point is the decomposition of the highest derivatives into some finite basis. Lower derivatives and finally the function itself are obtained by integration which has the property of damping out or at least containing any inherent inaccuracy in the assumed shape of the derivatives. At this stage, it is recommended that the IRBFN2 method is the better one among the methods considered for an accurate approximation of a function and its derivative. In a subsequent study, the application of the DRBFN and IRBFN methods in solving differential equations will be reported.

Acknowledgements

This work is supported by a Special USQ Research Grant to Thanh Tran-Cong (Grant No

179-310). Nam Mai-Duy is supported by a USQ scholarship. This support is gratefully acknowledged. The authors would like to thank the referees for their helpful comments.

References

[**1**] R.D. Cook, D.S. Malkus, M.E. Plesha, Concepts and Applications of Finite Element Analysis, John Wiley & Sons, Toronto, 1989.

[**2**] C.A. Brebbia, J.C.F. Telles, L.C. Wrobel, Boundary Element Techniques: Theory and Applications in Engineering, Springer-Verlag, Berlin, 1984.

[**3**] M.J.D. Powell, Radial basis functions for multivariable interpolation: a review, in: J.C. Watson, M.G. Cox (Eds), IMA Conference on Algorithms for the Approximation of Function and Data, Royal Military College of Science, Shrivenham, England, 1985, pp. 143-167.

[**4**] D.S. Broomhead, D. Lowe, Multivariable functional interpolation and adative networks, Complex Systems 2 (1988) 321-355.

[**5**] M.J.D. Powell, Radial basis function approximations to polynomial, in: D.F. Griffiths, G.A. Watson (Eds), Numerical Analysis 1987 Proceedings, University of Dundee, Dundee, UK, 1988, pp. 223-241.

[**6**] T. Poggio, F. Girosi, Networks for approximation and learning, in: Proceedings of the IEEE 78, 1990, pp.1481-1497.

[**7**] F. Girosi, T. Poggio, Networks and the best approximation property, Biological Cybernetics 63 (1990) 169-176.

[**8**] S. Chen, F.N. Cowan, P.M. Grant, Orthogonal least squares learning algorithm for radial basis function networks, IEEE Transaction on Neural Networks 2 (1991) 302-309.

[**9**] S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice-Hall, New Jersey, 1999.

[**10**] J. Park, I.W. Sandberg, Approximation and radial basis function networks, Neural Computation 5 (1993) 305-316.

[**11**] J. Moody, C.J. Darken, Fast learning in networks of locally-tuned processing units, Neural Computation 1 (1989) 281-294.

[**12**] R. Franke, Scattered data interpolation: tests of some methods, Mathematics of Computation 38(157) (1982) 181-200.

[**13**] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, Numerical Recipes in C: The Art of Scientific Computing, Cambridge University Press, Cambridge, 1988.

[**14**] A.E. Tarwater, A parameter study of Hardy's multiquadrics method for scattered data interpolation, Technical Report UCRL-563670, Lawrence Livemore National Laboratory, 1985.

[**15**] J.J. Dongarra, J.R. Bunch, C.B. Moler, G.W. Stewart, LINPACK User's Guide, SIAM, Philadelphia, 1979.

[**16**] S. Hashem, B. Schmeiser, Approximating a function and its derivatives using MSE-optimal linear combinations of trained feedforward neural networks, in: Proceedings of the 1993 World Congress on Neural Networks, vol 1, Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1993, pp. 617-620.

[**17**] S.V. Chakravarthy, J. Ghosh, Function emulation using radial basis function networks, Neural Networks 10 (1997) 459-478.

Table 1: N_e of the approximate function and its derivatives for $\beta = 2.0$ with the direct RBFN (DRBFN) approach. The quality of approximation deteriorates with higher derivatives.

	Gaussians	multiquadrics	Inverse multiquadrics
Original function	$7.002e-01$	$9.570e-02$	$7.459e-01$
1st derivative	$3.035e+01$	$4.053e+00$	$3.086e+01$
2nd derivative	$1.214e+03$	$1.564e+02$	$1.141e+03$

Table 2: Comparison of N_es between the DRBFN and IRBFN1 for the original function, $\beta = 2.0$.

	multiquadrics	Inverse multiquadrics
DRBFN	$9.570e-02$	$7.459e-01$
IRBFN1	$9.324e-04$	$1.510e-02$
Improvement factor	102.6	49.4

Table 3: Comparison of N_es between the DRBFN and IRBFN1 for the 1st derivative function, $\beta = 2.0$.

	multiquadrics	Inverse multiquadrics
DRBFN	$4.053e+00$	$3.086e+01$
IRBFN1	$4.720e-02$	$7.603e-01$
Improvement factor	85.9	40.6

Table 4: Comparison of N_es between the DRBFN and IRBFN1 for the 2nd derivative function, $\beta = 2.0$.

	multiquadrics	Inverse multiquadrics
DRBFN	$1.564e+02$	$1.141e+03$
IRBFN1	$1.671e+00$	$2.550e+01$
Improvement factor	93.6	44.7

Table 5: Comparison of N_es between the two indirect methods using multiquadrics for $\beta = 2.0$. Here the improvement factor is defined as the improvement of IRBFN2 relative to IRBFN1.

	Original	1st derivative	2nd derivative
IRBFN1	$9.324e-04$	$4.720e-02$	$1.671e+00$
IRBFN2	$3.968e-05$	$2.100e-03$	$1.022e-01$
Improvement factor	23.5	22.5	16.3

Table 6: Comparison of N_es between the two indirect methods using inverse multiquadrics for $\beta = 2.0$. Here the improvement factor is defined as the improvement of IRBFN2 relative to IRBFN1.

	Original	1st derivative	2nd derivative
IRBFN1	$1.510e-02$	$7.603e-01$	$2.550e+01$
IRBFN2	$2.200e-03$	$9.760e-02$	$4.073e+00$
Improvement factor	6.9	7.8	6.3

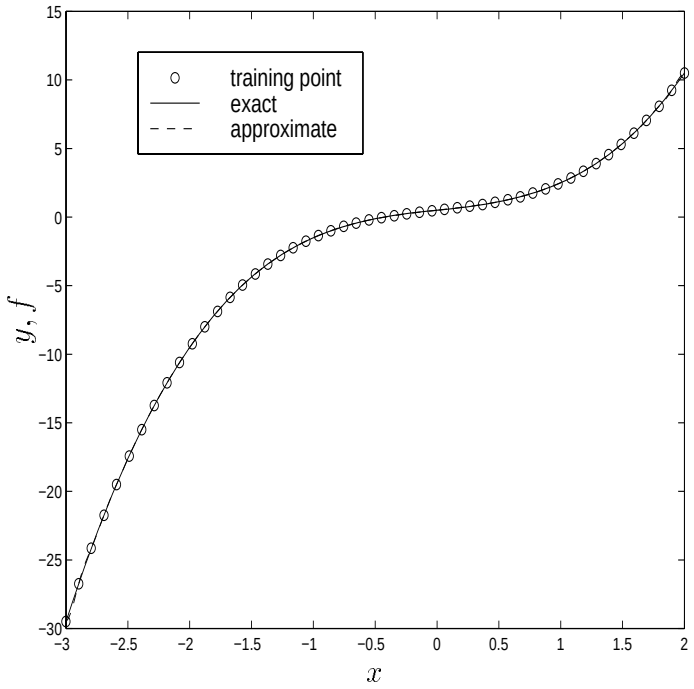

Figure 1: Function $y(x) = x^3 + x + 0.5$: plot of training points, the exact function and the approximate function obtained by the direct RBFN using inverse multiquadrics basis functions (DRBFN) with $\beta = 2.0$. Note that the accuracy of the approximation of the function is such that the error (i.e. the difference between the dashed and the solid lines) is not discernible on this plot. However, the goodness of the global shape might not be good enough in obtaining accurate function derivatives as illustrated in the next Figure 2.

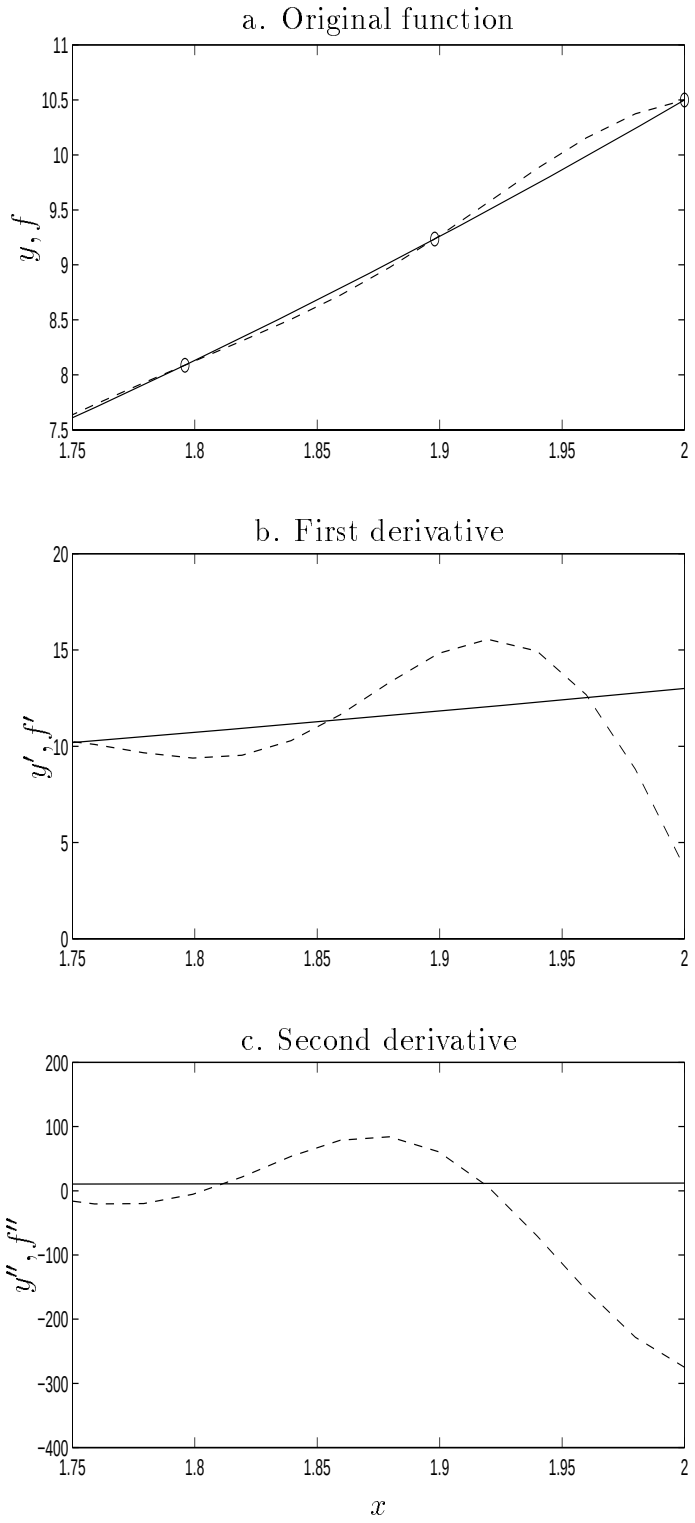

Figure 2: Function $y(x) = x^3 + x + 0.5$: Zoom in on the original, first derivative and second derivative functions ($\beta = 2.0$). Solid line: exact function and dashed line: DRBFN approximation using inverse multiquadrics. The plots illustrate the shortcomings of the DRBFN approach where the associated error norms are $7.459e{-}1$, $3.086e{+}1$ and $1.141e{+}3$ for the approximation of the function, its first derivative and second derivative respectively.

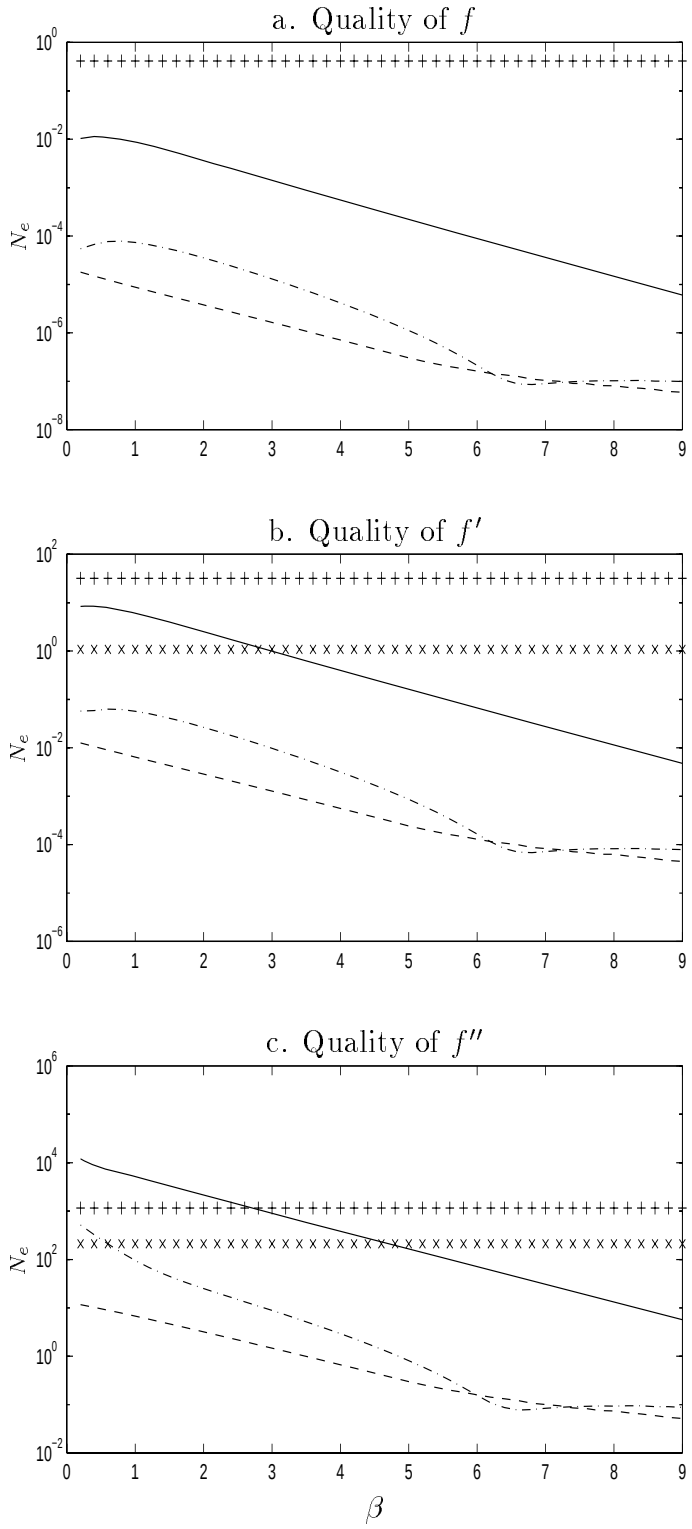

Figure 3: Approximant f of the function $y = 0.02(12+3x-3.5x^2+7.2x^3)(1+\cos 4\pi x)(1+0.8\sin 3\pi x)$ and its derivatives: plots of the norm N_e as a function of β. Legends +: MSE-OLC, x: conventional element method, solid line: DRBFN, dashdot line: IRBFN1 and dashed line: IRBFN2.

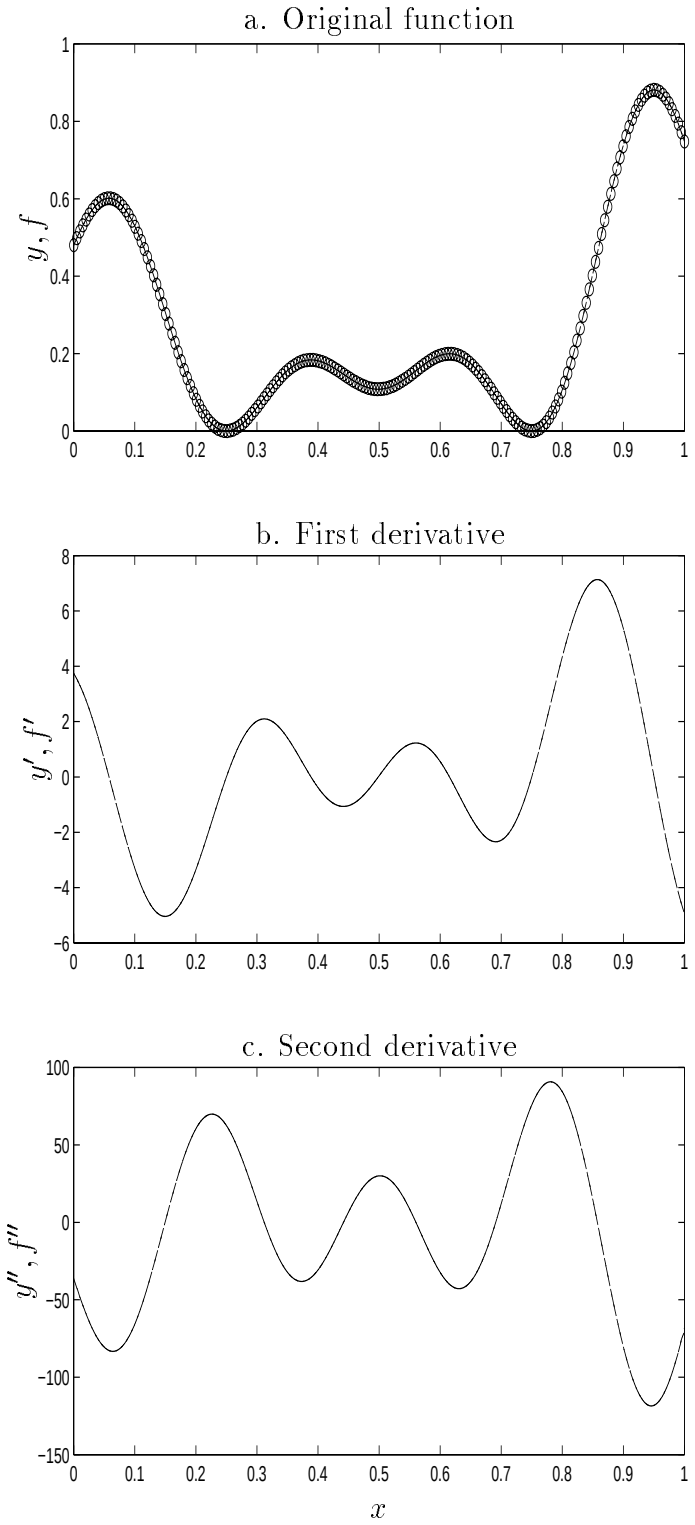

Figure 4: Function $y = 0.02(12 + 3x - 3.5x^2 + 7.2x^3)(1 + \cos 4\pi x)(1 + 0.8 \sin 3\pi x)$ and its derivatives: plots of function and its derivatives at the "worst" value of β (0.2). Dashed line: exact and dashdot line: IRBFN2. The quality of the IRBFN2 approximation is such that the numerical approximation and the analytical plots are not discernible. The data points are also shown as $\circ$.

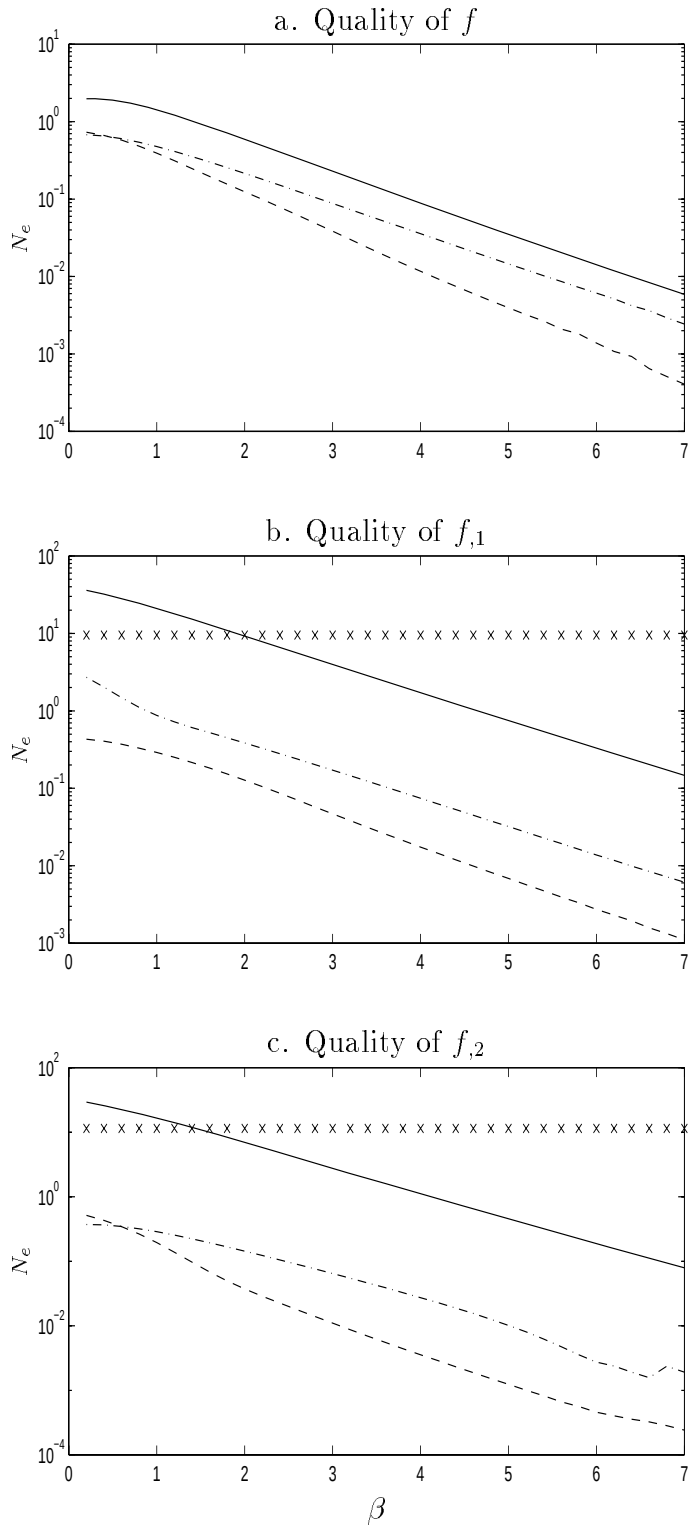

Figure 5: Approximant f of the function $y(x_1, x_2) = x_1^2 x_2 + x_2^3/3 + x_2^2/2$ and its derivatives: plots of the norm N_e as a function of β. Legends x: conventional element method, solid line: DRBFN, dashdot line: IRBFN1 and dashed line: IRBFN2. It can be seen that the quality of the approximation for the derivatives is much better with the IRBFN approach.

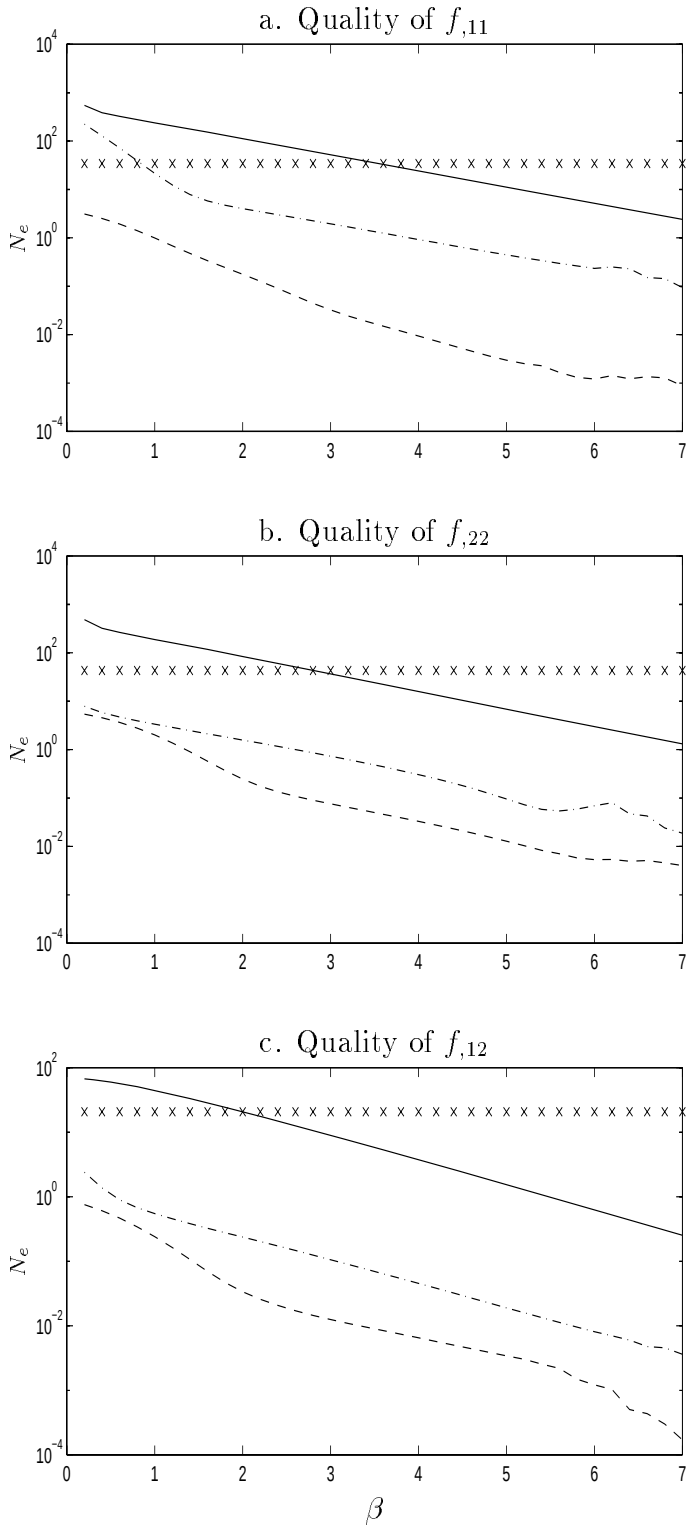

Figure 6: Approximant of the derivatives of the function $y(x_1, x_2) = x_1^2 x_2 + x_2^3/3 + x_2^2/2$: plots of the norm N_e as a function of β. Legends x: conventional element method, solid line: DRBFN, dashdot line: IRBFN1 and dashed line: IRBFN2. It can be seen that the quality of the approximation for the derivatives is much better with the IRBFN approach.

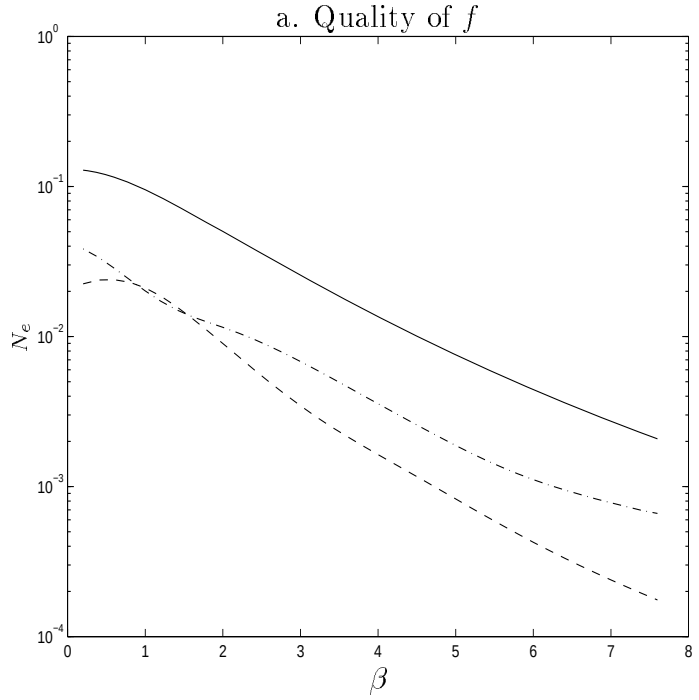

Figure 7: Approximant of the function $y = x_1^2 + x_1x_2 - 2x_2^2 - x_2x_3 + x_3^2$: plots of the norm N_e as a function of β. Solid line: DRBFN, dashdot line: IRBFN1 and dashed line: IRBFN2. It can be seen that the quality of the approximation is much better with the IRBFN approach.

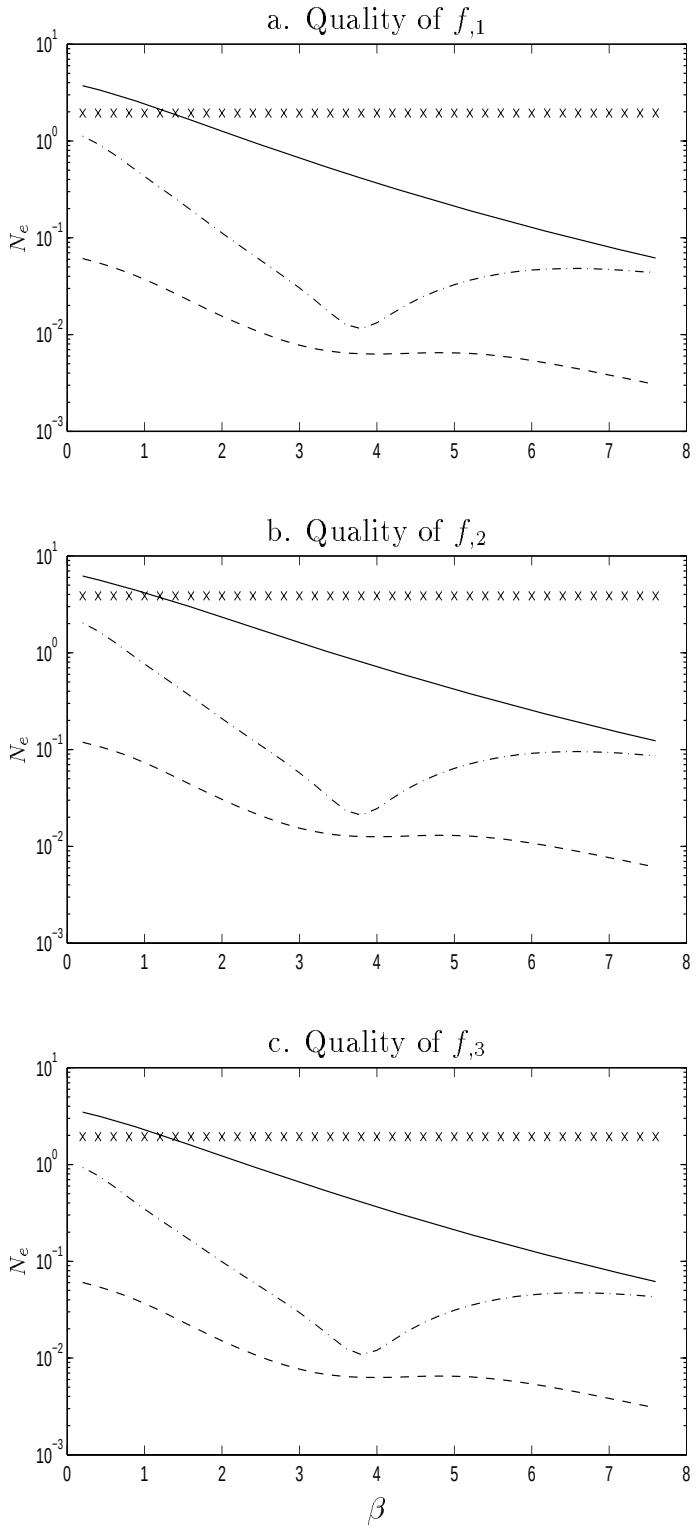

Figure 8: Approximant of the derivatives of the function $y = x_1^2 + x_1x_2 - 2x_2^2 - x_2x_3 + x_3^2$: plots of the norm N_e as a function of β. Legends x: conventional element method, solid line: DRBFN, dashdot line: IRBFN1 and dashed line: IRBFN2. It can be seen that the quality of the approximation is much better with the IRBFN approach.

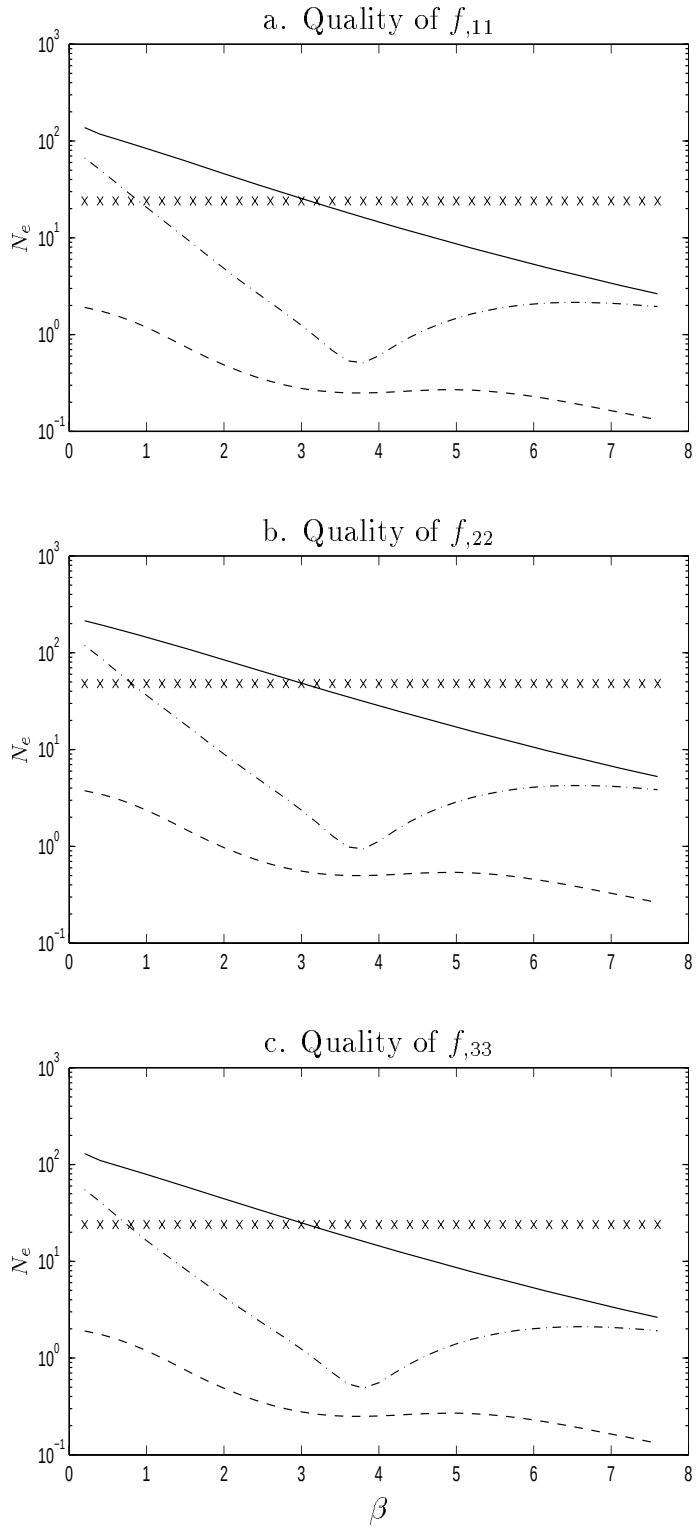

Figure 9: Approximant of the derivatives of the function $y = x_1^2 + x_1x_2 - 2x_2^2 - x_2x_3 + x_3^2$: plots of the norm N_e as a function of β. Legends x: conventional element method, solid line: DRBFN, dashdot line: IRBFN1 and dashed line: IRBFN2. It can be seen that the quality of the approximation is much better with the IRBFN approach.

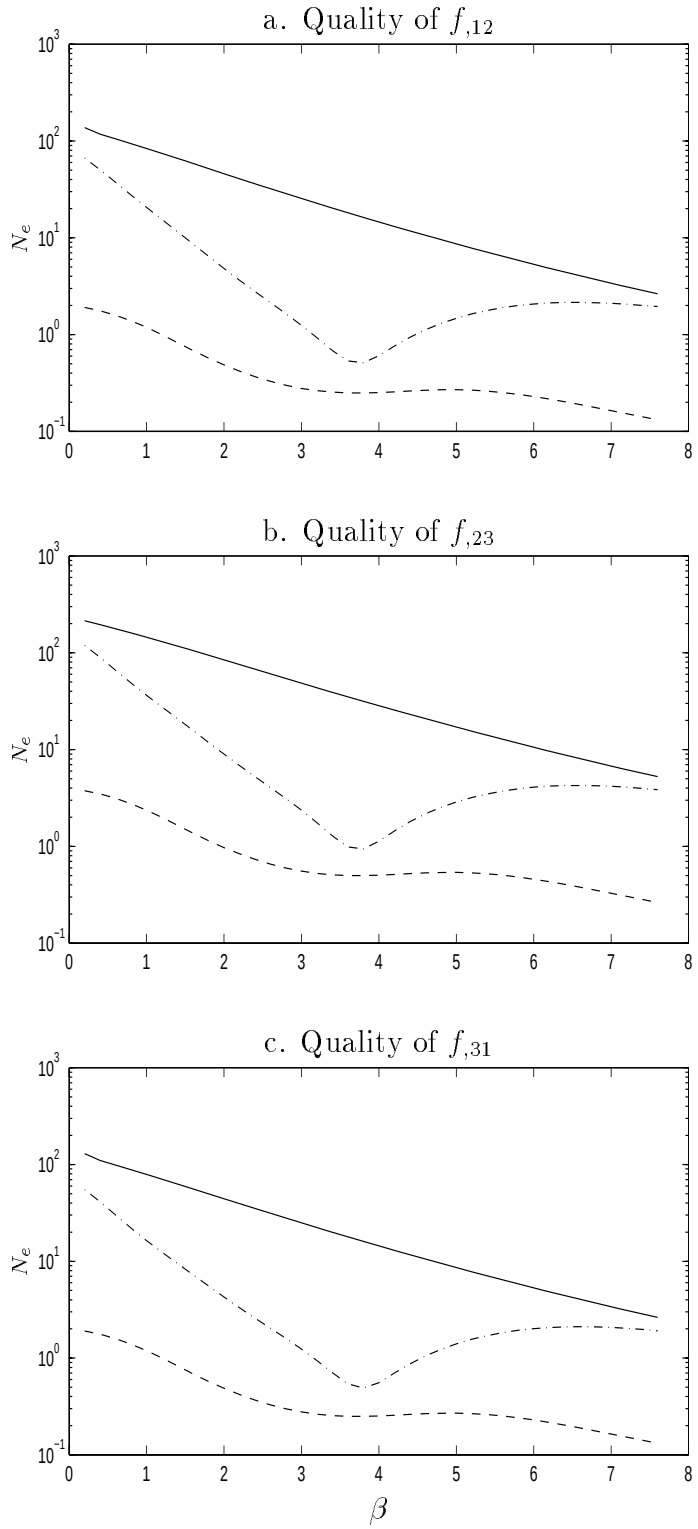

Figure 10: Approximant of the derivatives of the function $y = x_1^2 + x_1x_2 - 2x_2^2 - x_2x_3 + x_3^2$: plots of the norm N_e as a function of β. Solid line: DRBFN, dashdot line: IRBFN1 and dashed line: IRBFN2. It can be seen that the quality of the approximation is much better with the IRBFN approach.

Abbreviated title for running headline

Function approximation by RBFNs

Figure Captions

Figure 1: Function $y(x) = x^3 + x + 0.5$: plot of training points, the exact function and the approximate function obtained by the direct RBFN using inverse multiquadrics basis functions (DRBFN) with $\beta = 2.0$. Note that the accuracy of the approximation of the function is such that the error (i.e. the difference between the dashed and the solid lines) is not discernible on this plot. However, the goodness of the global shape might not be good enough in obtaining accurate function derivatives as illustrated in the next Figure 2.

Figure 2: Function $y(x) = x^3 + x + 0.5$: Zoom in on the original, first derivative and second derivative functions ($\beta = 2.0$). Solid line: exact function and dashed line: DRBFN approximation using inverse multiquadrics. The plots illustrate the shortcomings of the DRBFN approach where the associated error norms are $7.459e{-}1$, $3.086e{+}1$ and $1.141e{+}3$ for the approximation of the function, its first derivative and second derivative respectively.

Figure 3: Approximant f of the function $y = 0.02(12+3x-3.5x^2+7.2x^3)(1+\cos 4\pi x)(1+0.8\sin 3\pi x)$ and its derivatives: plots of the norm N_e as a function of β. Legends +: MSE-OLC, x: conventional element method, solid line: DRBFN, dashdot line: IRBFN1 and dashed line: IRBFN2.

Figure 4: Function $y = 0.02(12 + 3x - 3.5x^2 + 7.2x^3)(1 + \cos 4\pi x)(1 + 0.8\sin 3\pi x)$ and its derivatives: plots of function and its derivatives at the "worst" value of β (0.2). Dashed line: exact and dashdot line: IRBFN2. The quality of the IRBFN2 approximation is such

that the numerical approximation and the analytical plots are not discernible. The data points are also shown as $\circ$.

Figure 5: Approximant f of the function $y(x_1, x_2) = x_1^2 x_2 + x_2^3/3 + x_2^2/2$ and its derivatives: plots of the norm N_e as a function of β. Legends x: conventional element method, solid line: DRBFN, dashdot line: IRBFN1 and dashed line: IRBFN2. It can be seen that the quality of the approximation for the derivatives is much better with the IRBFN approach.

Figure 6: Approximant of the derivatives of the function $y(x_1, x_2) = x_1^2 x_2 + x_2^3/3 + x_2^2/2$: plots of the norm N_e as a function of β. Legends x: conventional element method, solid line: DRBFN, dashdot line: IRBFN1 and dashed line: IRBFN2. It can be seen that the quality of the approximation for the derivatives is much better with the IRBFN approach.

Figure 7: Approximant of the function $y = x_1^2 + x_1 x_2 - 2x_2^2 - x_2 x_3 + x_3^2$: plots of the norm N_e as a function of β. Solid line: DRBFN, dashdot line: IRBFN1 and dashed line: IRBFN2. It can be seen that the quality of the approximation is much better with the IRBFN approach.

Figure 8: Approximant of the derivatives of the function $y = x_1^2 + x_1 x_2 - 2x_2^2 - x_2 x_3 + x_3^2$: plots of the norm N_e as a function of β. Legends x: conventional element method, solid line: DRBFN, dashdot line: IRBFN1 and dashed line: IRBFN2. It can be seen that the quality of the approximation is much better with the IRBFN approach.

Figure 9: Approximant of the derivatives of the function $y = x_1^2 + x_1 x_2 - 2x_2^2 - x_2 x_3 + x_3^2$: plots of the norm N_e as a function of β. Legends x: conventional element method, solid line: DRBFN, dashdot line: IRBFN1 and dashed line: IRBFN2. It can be seen that the

quality of the approximation is much better with the IRBFN approach.

Figure 10: Approximant of the derivatives of the function $y = x_1^2 + x_1 x_2 - 2x_2^2 - x_2 x_3 + x_3^2$: plots of the norm N_e as a function of β. Solid line: DRBFN, dashdot line: IRBFN1 and dashed line: IRBFN2. It can be seen that the quality of the approximation is much better with the IRBFN approach.