# Moving Least Square Approximations for Solution of Differential Equations

R.L. Taylor
O.C. Zienkiewicz
E. Oñate
S. Idelsohn

Publication CIMNE Nº 74, December 1995

# Moving Least Square Approximations for Solution of Differential Equations

R.L. Taylor
O.C. Zienkiewicz
E. Oñate
S. Idelsohn

Publication CIMNE Nº 74, December 1995

International Center for Numerical Methods in Engineering
Gran Capitán s/n, 08034 Barcelona, Spain

# MOVING LEAST SQUARE APPROXIMATIONS FOR SOLUTION OF DIFFERENTIAL EQUATIONS

R. L. Taylor[1]
O. C. Zienkiewicz[2]
E. Oñate[3]
S. Idelsohn[4]

International Center for Numerical Methods in Engineering
Universitat Politècnica de Catalunya
Edificio C1, Gran Capitán s/n,
08034, Barcelona, Spain

## 1. Introduction

The use of local least square approximations to fit sets of data has been reported in several studies [8,18,21]. Recently, several authors have explored the use of least square approximations for the solution of differential equations associated with solid and fluid mechanics [2,9,16,17,19]. Generally, the introduction of weighting functions leads to improved results as shown in Oñate et. al. for compressible fluid flow solutions [15]. One variant of a weighted approach is the *moving least square method* (MLS) introduced by Lancaster and Salkauskas [7]. In MLS a weighting function is introduced and allowed to move continuously to the location where a least square fit is to be obtained. The method can lead to continuous interpolation in terms of a set of discrete parameters and thus provides a basis for constructing discrete approximate solutions to differential equations, as well as, for data fitting. Belytschko and co-authors have presented a number of studies applied to problems in solid mechanics and calls the approach an *element free Galerkin method* (EFG) [3,4,10-14,22].

In this report we review the moving least square method in Section 2 and in Section 3 a recent extension based on hierarchical approximations as proposed independently by Babuška and Melenk [1] and by Duarte and Oden [6]. We call the extension a *hierarchical moving least square method* (HMLS). The effects of the continuity of a moving least square weighting function and its domain of influence is investigated in Section 4. Based on this assessment it is concluded that weighting functions which have continuous derivatives up to the order necessary in the approximating procedure are an essential ingredient for accurate

[1] Professor of Civil Engineering, University of California, Berkeley. Visiting Professor, CIMNE

[2] Professor of Civil Engineering, University College, Swansea. UNESCO Professor, UPC

[3] Director, CIMNE

[4] Professor, Universidad del Litoral, Sante Fe, Argentina. Visiting Professor, CIMNE

representations of solutions. In Section 5 we illustrate this by considering an application of MLS and HMLS to approximately solve ordinary differential equations by a *finite point method* (FPM) proposed by Oñate et. al. [15]. Finally, in Section 6 an application of MLS and HMLS using a Galerkin approach is presented.

## 2. Moving Least Square Approximation

The moving least square method is a weighted least square approximation in which the weighting function is allowed to move based on the point where the fit is to be obtained. To describe the method, consider the problem of fitting an approximation to a set of data items $u_i, i = 1, n$ defined at the $n$ points $x_i$. We assume the approximating function is to be described by the relation

$$u(x) \approx \hat{u}(x) = \sum_{i=1}^{m} p_i(x)\alpha_i = \boldsymbol{p}^T(x)\boldsymbol{\alpha}$$

where $p_i$ are a set of linearly independent functions and $\alpha_i$ are unknown quantities to be determined by the fit algorithm. A moving least square fit to the given data may be defined for each point $\hat{x}$ in the domain by solving the problem

$$J(\hat{x}) = \sum_{k=1}^{n} \varphi_{\hat{x}}(x_k - \hat{x})[u_k - \boldsymbol{p}^T(x_k)\boldsymbol{\alpha}(\hat{x})]^2 = min$$

where $\varphi_{\hat{x}}$ can in general change its shape and span depending on the position of the point $\hat{x}$. We note that the point $\hat{x}$ is an arbitrary position and the $\alpha_i$ now depend on the position selected. After the $\alpha_i(\hat{x})$ are computed, substitution into the expression for $u(x)$ above is achieved with

$$u(x) \approx \hat{u}(x) = \sum_{i=1}^{m} p_i(x)\alpha_i(\hat{x})|_{\hat{x}=x}$$

We retain the form $\varphi_{\hat{x}}(x_k - \hat{x})$ to emphasize the possibility of a changing function at each position where the minimization is to occur. In the simplest case, with constant spacing of the data points $x_k$, it is possible to take

$$\varphi_{\hat{x}}(x_k - \hat{x}) = \varphi(x_k - \hat{x})$$

and assume the shape and span of the weighting is invariant with position.

In general with an arbitrary spacing of the points $x_k$, the problem of specifying $\varphi_{\hat{x}}$ at every position $\hat{x}$ is very difficult and presents an infinite number of possibilities. It is convenient instead to assume weighting functions associated to the data points and take

$$\varphi_{\hat{x}}(x_k - \hat{x}) \;=\; \phi_k(\hat{x} - x_k)$$

where we again emphasize that $\hat{x}$ denotes an arbitrary point whereas $x_k$ are fixed points in the domain. With this assumption the weight functions $\phi_k$ are fixed at each $x_k$ and the function to be minimized is now

$$J(\hat{x}) = \sum_{k=1}^{n} \phi_k(\hat{x} - x_k)[u_k - \boldsymbol{p}^T(x_k)\boldsymbol{\alpha}(\hat{x})]^2 = min$$

If each weighting function is defined such that

$$\phi_k(y) = \begin{cases} f_k(y), & \text{if } |y| < r_k; \\ 0, & \text{otherwise.} \end{cases}$$

then the terms in the sum are zero whenever $y = \hat{x} - x_k$ and $|y| < r_k$. The parameter $r_k$ defines a radius of a ball around each point, $x_k$; inside the ball the weighting function is non zero while outside the radius it is zero. Each point may have a different weighting function or radius of the ball around its defining point. Note that the weighting function is defined such that it is zero on the boundary of the ball (i.e., $y = r_k$). This class of function may be denoted as $C_0^q(B_{r_k})$, where the subscript denotes the boundary value of zero, the superscript denotes the highest derivative for which $C^0$ continuity is achieved, and $B_{r_k}$ denotes a ball of radius $r_k$. The solution to the least square problem leads to

$$\boldsymbol{\alpha}(\hat{x}) = \boldsymbol{A}^{-1}(\hat{x}) \sum_{j=1}^{n} \boldsymbol{B}_j(\hat{x})\, u_j$$

where

$$\boldsymbol{A}(\hat{x}) = \sum_{k=1}^{n} \phi_k(\hat{x} - x_k) \boldsymbol{p}(x_k) \boldsymbol{p}^T(x_k)$$

and

$$\boldsymbol{B}_j(\hat{x}) = \phi_j(\hat{x} - x_j) \boldsymbol{p}(x_j)$$

In matrix form the array $\boldsymbol{A}(\hat{x})$ may be written as

$$\boldsymbol{A}(\hat{x}) = [\,\boldsymbol{p}_1 \quad \boldsymbol{p}_2 \quad \cdots \quad \boldsymbol{p}_N\,] \begin{pmatrix} \phi_1(\hat{x} - x_1) & 0 & \cdots & \cdots \\ 0 & \phi_2(\hat{x} - x_2) & 0 & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ \cdots & \cdots & 0 & \phi_N(\hat{x} - x_N) \end{pmatrix} \begin{bmatrix} \boldsymbol{p}_1^T \\ \boldsymbol{p}_2^T \\ \vdots \\ \boldsymbol{p}_N^T \end{bmatrix}$$

where

$$\boldsymbol{p}_k = \boldsymbol{p}(x_k) = \begin{pmatrix} p_1(x_k) \\ p_2(x_k) \\ \vdots \end{pmatrix}$$

From the above it is evident that use of weighting functions $\phi_k(\hat{x} - x_k)$ which are zero outside a ball of radius $r_k$ results in only a limited number of terms contributing to the summation. Note also that the moving least square algorithm produces solutions for $\boldsymbol{\alpha}$ which depend on the point selected for each approximation.

Based on the moving least square solution the approximation for the function $u(x)$ now may be written as

$$\hat{u}(x) = \sum_{j=1}^{n} N_j(x)\, u_j$$

where

$$N_j(x) = \boldsymbol{p}^T(x)\boldsymbol{A}^{-1}(\hat{x})\boldsymbol{B}_j(\hat{x})|_{\hat{x}=x}$$

define interpolation functions for each data item $u_j$.

It is easy to show that the interpolations defined by the above can approximate all the functions used to define $\boldsymbol{p}(x)$. To do so, consider the set of approximations

$$\boldsymbol{U} = \sum_{j=1}^{n} N_j(x)\ \boldsymbol{U}_j$$

where

$$\boldsymbol{U} = [\,\hat{u}_1(x) \quad \hat{u}_2(x) \quad \ldots \quad \hat{u}_n(x)\,]$$

and

$$\boldsymbol{U}_j = [\,u_{j1} \quad u_{j2} \quad \ldots \quad u_{jn}\,]$$

and assign to each $u_{jk}$ the value of the funtion $p_k(x_j)$ (i.e., the $k$-th entry in $\boldsymbol{p}$) so that

$$\boldsymbol{U}_j = \boldsymbol{p}^T(x_j)$$

Now, using the definition of the interpolation functions we have

$$\boldsymbol{U} = \sum_{j=1}^{n} N_j(x)\ \boldsymbol{p}^T(x_j) = \sum_{j=1}^{n} \boldsymbol{p}^T(x)\boldsymbol{A}^{-1}(x)\boldsymbol{B}_j(x)\boldsymbol{p}^T(x_j)$$

which after substitution of the definition of $\boldsymbol{B}_j(x)$ yields

$$\boldsymbol{U} = \sum_{j=1}^{n} \boldsymbol{p}^T(x)\boldsymbol{A}^{-1}(x)\phi_j(x - x_j)\boldsymbol{p}(x_j)\boldsymbol{p}^T(x_j)$$

Rewriting as

$$\boldsymbol{U} = \boldsymbol{p}^T(x)\boldsymbol{A}^{-1} \sum_{j=1}^{n} \phi_j(x - x_j)\boldsymbol{p}(x_j)\boldsymbol{p}^T(x_j)$$

yields

$$\boldsymbol{U} = \boldsymbol{p}^T(x)\boldsymbol{A}^{-1}\,\boldsymbol{A}(x) = \boldsymbol{p}^T(x)$$

which shows that the form can interpolate exactly any function included as part of the definition of $\boldsymbol{p}(x)$. If polynomials are used to define the functions, the interpolation always includes exact representations for each included polynomial. Inclusion of the zero-order polynomial (i.e., 1), implies that

$$\sum_{j=1}^{n} N_j(x) = 1$$

In the mathematical literature this is known as a *partition of unity* (provided it is true for all points, $x$, in the domain) [20]. It is easy to recognize that this is the same requirement as applies to most finite element shape functions.

Derivatives of the interpolation function may be constructed by defining

$$\boldsymbol{\alpha}(\hat{x})|_{\hat{x}=x} = \sum_{j=1}^{n} \boldsymbol{w}_j(x) u_j$$

and using the representation

$$N_j(x) = \boldsymbol{p}^T(x)\boldsymbol{w}_j(x)$$

where

$$\boldsymbol{A}(x)\boldsymbol{w}_j(x) = \boldsymbol{B}_j(x)$$

For example, the first derivative is given by

$$\frac{dN_j}{dx} = \frac{d\boldsymbol{p}^T}{dx}\boldsymbol{w}_j + \boldsymbol{p}^T\frac{d\boldsymbol{w}_j}{dx}$$

and

$$\boldsymbol{A}\frac{d\boldsymbol{w}_j}{dx} + \frac{d\boldsymbol{A}}{dx}\boldsymbol{w}_j = \frac{d\boldsymbol{B}_j}{dx}$$

where

$$\frac{d\boldsymbol{A}}{dx} = \sum_{k=1}^{n} \frac{d\phi_k(x-x_k)}{dx}\boldsymbol{p}(x_k)\boldsymbol{p}^T(x_k)$$

and

$$\frac{d\boldsymbol{B}_j}{dx} = \frac{d\phi_j(x-x_j)}{dx}\boldsymbol{p}(x_j)$$

Solving for $\frac{d\boldsymbol{w}_j}{dx}$ and substituting into the equation for the derivative yields

$$\frac{dN_j}{dx} = \left(\frac{d\boldsymbol{p}^T}{dx} - \boldsymbol{p}^T\boldsymbol{A}^{-1}\frac{d\boldsymbol{A}}{dx}\right)\boldsymbol{w}_j + \boldsymbol{p}^T\boldsymbol{A}^{-1}\frac{d\boldsymbol{B}_j}{dx}$$

Higher derivatives may be computed by repeating the above process to define the higher derivatives of $\boldsymbol{w}_j$.

Nayroles et.al. suggest that approximations ignoring the derivatives of $\boldsymbol{\alpha}$ may be used to define the derivatives of the interpolation functions [16-18]. Accordingly, as an approximation they suggest the use of

$$\frac{dN_j}{dx} \approx \frac{d\hat{N}_j}{dx} = \frac{d\boldsymbol{p}^T}{dx}\boldsymbol{w}_j$$

for the first derivative. This approximation simplifies the construction of derivatives as it is no longer necessary to compute the derivatives for $\boldsymbol{A}$ and $\boldsymbol{B}_j$. However, there is little

additional effort required to compute the derivatives of the weighting function. Furthermore, for the lowest order approximation using only a constant for the $\boldsymbol{p}$ ignoring the derivative results in no derivatives being available. Consequently, we have not utilized the above approximation for any of the results reported below.

## 3. Hierarchical Enhancement of Moving Least Square Approximation

In the previous section the moving least square solution for the approximation of the function $u(x)$ was given as

$$\hat{u}(x) = \sum_{j=1}^{n} N_j(x)\ u_j$$

where $N_j(x)$ define interpolation functions based on linearly dependent functions prescribed by $\boldsymbol{p}(x)$. In the sequel we employ polynomial functions to describe $\boldsymbol{p}(x)$, with a polynomial of degree $k$ given by

$$\boldsymbol{p}(x) = [\,1 \quad x \quad x^2 \quad \ldots \quad x^k\,]$$

For this case we will denote the resulting interpolation functions using the notation $N_j^k(x)$, where $j$ is associated with the location of the point where the parameter $u_j$ is given and $k$ denotes the order of the polynomial approximating functions. Duarte and Oden suggest using Legendre polynomials instead of the form given above [6]; however, conceptionally the two are equivalent. Duarte and Oden also present a hierarchical construction based on $N_j^k(x)$ which increases the order of the polynomial to $p$. In their construction, the interpolation is written as

$$\hat{u}(x) = \sum_{j=1}^{n} \left( N_j^k(x)\ u_j + N_j^k(x)\,[\,x^{k+1} \quad x^{k+2} \quad \ldots \quad x^p\,] \begin{pmatrix} b_{j1} \\ b_{j2} \\ \vdots \\ b_{jq} \end{pmatrix} \right)$$

$$= \sum_{j=1}^{n} N_j^k(x)\,(\ u_j + \boldsymbol{q}^T(x)\boldsymbol{b}_j)$$

where $q = p - k$ and $b_{jl}$, for $l = 1, \ldots, q$, are additional parameters for the approximation. Derivatives of the interpolation function may be constructed using the method given in the previous section.

The advantage of the above method lies in the reduced cost of computing the interpolation function $N_j^k(x)$ compared to that required to compute the $p$-order interpolations $N_j^p(x)$. For example, use of the functions $N_j^0(x)$, which are called Shepard interpolations [21], leads to a scalar matrix $\boldsymbol{A}$ which is trivial to invert to define the $N_j^0$. Specfically, the Shepard interpolations are

$$N_j^0(x) = A^{-1}(x)B_j(x)$$

where

$$A(x) = \sum_{k=1}^{n} \phi_k(x - x_k)$$

and

$$B_j(x) = \phi_j(x - x_j)$$

The fact that the hierarchical interpolations include polynomials up to order $p$ is easy to demonstrate. Based on the results from the previous section the interpolation $\phi_k(y)$ contains all the polynomials up to order $k$. The higher polynomials may be constructed from

$$\hat{u}(x) = \sum_{j=1}^{n} \left( N_j^k(x)\ u_j + N_j^k(x) \begin{bmatrix} x^{k+1} & x^{k+2} & \dots & x^p \end{bmatrix} \begin{pmatrix} b_{j1} \\ b_{j2} \\ \vdots \\ b_{jq} \end{pmatrix} \right)$$

by setting all the $u_j$ to zero and then for each interpolation term $j$ setting one of the $b_{jk}$ to unity with the remaing values set to zero. For example, setting $b_{j1}$ to unity results in the expansion

$$\hat{u}(x) = \sum_{j=1}^{n} N_j^k(x) x^{k+1} = x^{k+1}$$

This result requires only that

$$\sum_{j=1}^{n} N_j^k(x) = 1$$

which was already verified. The remaining polynomials are obtained by setting the other values of $b_{jk}$ to unity one at a time. Calculations provided by Duarte and Oden show that the same order approximation is obtained with $k = 0, 1$ or $p$.

The above hierarchical form has parameters which do not relate to approximate values of the interpolation function. For the case where $k = 0$, Babuška and Melenk [1] suggest that an alternate expression be used in which

$$\hat{u}(x) = \sum_{j=1}^{n} N_j^0(x) \begin{bmatrix} 1 & x^1 & x^2 & \dots & x^p \end{bmatrix} \begin{pmatrix} u_j \\ b_{j1} \\ b_{j2} \\ \vdots \\ b_{jq} \end{pmatrix}$$

be written as

$$\hat{u}(x) = \sum_{j=1}^{n} N_j^0(x) \left( \sum_{k=0}^{p} L_k^p(x) u_{jk} \right)$$

where $L_k^p(x)$ are Lagrange interpolation polynomials and $u_{jk}$ are parameters with dimensions of $u$ for the $j$-th term at point $x_k$ of the Lagrange interpolation. The above result follows since the Lagrange interpolation polynomials have the property

$$L_k(x_i) = \delta_{ki} = \begin{cases} 1, & \text{if } k = i; \\ 0, & \text{otherwise} \end{cases}$$

Babuška and Melenk also note that any functions which satisfy the partition of unity may be used to define the $N_j^k(x)$. In particular finite element interpolations may be used for this purpose. Consequently, a hierarchical interpolation may be generated for standard finite element forms, for the moving least square approximations, or for any other functions which satisfy the partition of unity. An example, which can be used for any finite element implementation requiring only first derivatives is to define

$$\hat{u}(x) = \sum_{j=1}^{n} N_j^1(x) \begin{bmatrix} 1 & x^2 \end{bmatrix} \begin{bmatrix} u_j \\ b_j \end{bmatrix}$$

where $N_J^1$ are standard linear finite element interpolations for elements which include the polynomials 1 and $x$. Similarly, for two dimensions one can use interpolations over triangular elements as

$$\hat{u}(x,y) = \sum_{j=1}^{n} N_j^1(x,y) \begin{bmatrix} 1 & x^2 & xy & y^2 \end{bmatrix} \begin{bmatrix} u_j \\ b_{j1} \\ b_{j2} \\ b_{j3} \end{bmatrix}$$

where $N_j^1(x,y)$ are the *tent* functions resulting from linear interpolations on each element connected to node $j$.

We should also note that in addition to the options for choosing $N_j^k(x)$, a similar freedom holds to define functions used for the $\boldsymbol{q}$. Thus, for any function $q_i(x)$ we can set the associated $b_{ji}$ to unity (with all others and $u_j$ set to zero) and obtain

$$\hat{u}(x) = \sum_{j=1}^{n} N_J^k(x)\ q_i(x) = q_i(x)$$

Again the only requirement being that

$$\sum_{j=1}^{n} N_j^k(x) = 1$$

Thus, for any basic functions satisfying the partition of unity a hierarchical enrichment may be added using any type of functions. For example, if one knows the structure of the solution involves exponential functions in $x$ it is possible to include them as members

of the $\boldsymbol{q}(x)$ functions and, thus, capture the essential part of the solution with just a few terms. This is especially important for problems which involve solutions with different length scales. Large length scale can be included in the basic functions, $N_j^k(x)$, and while other smaller length scales may be included in the functions $\boldsymbol{q}(x)$.

## 4. Behavior of Interpolation Functions and Derivatives

To assess aspects of moving least square interpolation functions we consider a set of test examples for uniformly spaced points. We first illustrate the importance of the weight function by considering four different forms. These are:

1. Piecewise constant interpolation given by:

$$f_{k1}(y) = 1 \quad ; \quad | y | < r_k$$

2. Hat function given by:

$$f_{k2}(y) = 1 - \frac{| y |}{r_k} \quad ; \quad | y | < r_k$$

3. Truncated Gaussian function given by:

$$f_{k3}(y) = \frac{\exp(-\xi^2) - \exp(-\xi_k^2)}{1 - \exp(-\xi_k^2)} \quad ; \quad | y | < r_k$$

where

$$\xi = \frac{y}{c}$$

$$\xi_k = \frac{r_k}{c}$$

and

$$c = \kappa r_k$$

with $\kappa$ a specified parameter (in our tests we set $\kappa$ to one-half).

4. A smooth polynomial given by:

$$f_{k4}(y) = [1 - (\frac{y}{r_k})^2]^m \quad ; \quad | y | < r_k$$

where $m$ is a specified positive integer.

The weighting function for each of the above is taken, as defined previously, by:

$$\phi_k(y) = \begin{cases} f_{ki}(y), & \text{if } | y | < r_k; \\ 0, & \text{otherwise.} \end{cases}$$

for $i = 1, 4$; consequently, all of the above weighting functions are zero outside the range $\mid y \mid < r_k$. The above weighting functions have the property that $f_{k1}$ has continuity $C_0^{-1}(r_k)$; $f_{k2}$ and $f_{k3}$ have $C_0^0(r_k)$; and $f_{k4}$ has continuity $C_0^{m-1}(r_k)$. Functions 2 and 3 differ in that $f_{k3}$ has all derivatives continuous at $y = 0$ whereas $f_{k2}$ is only $C^0$ continuous at this point.

To demonstrate the differences the above weighting functions produce in generating shape functions we consider the class of quadratic hierarchical interpolations generated from the $N_j^0$ Shepard basic functions. All of the results shown are for a 7 point equally spaced region using a weighting function with a total span of 4.1 mesh spaces (i.e. $r_k = 2.05$). We show in all the comparisons the function $N_4^0$ with the first and second derivatives denoted as $dN$ and $d2N$, respectively.

The results for $N_4^0$ produced by a uniform type 1 weight function are shown in Figure 4.1a. Figures 4.1b and 4.1c show results for $xN_4^0$ and $x^2N_4^0$, respectively.

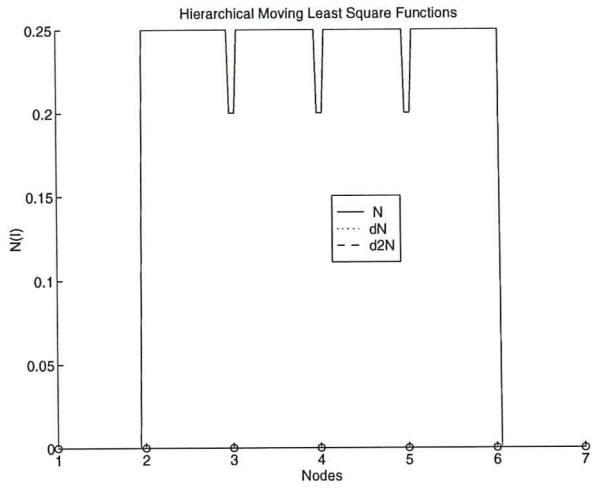


Figure 4.1a. Shepard Interpolation: $N_4^0$. Type 1 Weight Function, $r_k = 2.05$

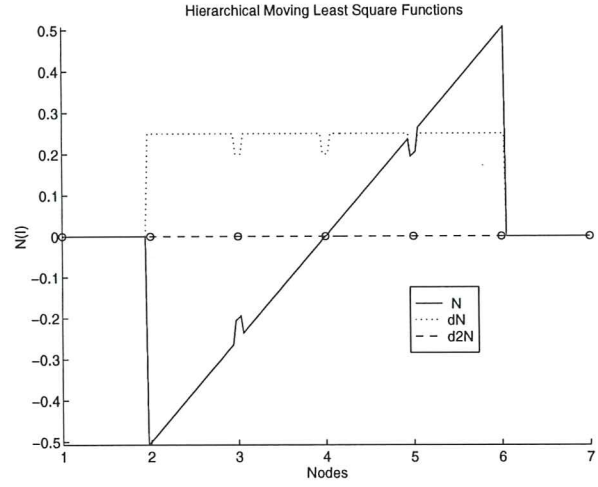


Figure 4.1b. Hierarchical Interpolation: $xN_4^0$. Type 1 Weight Function, $r_k = 2.05$

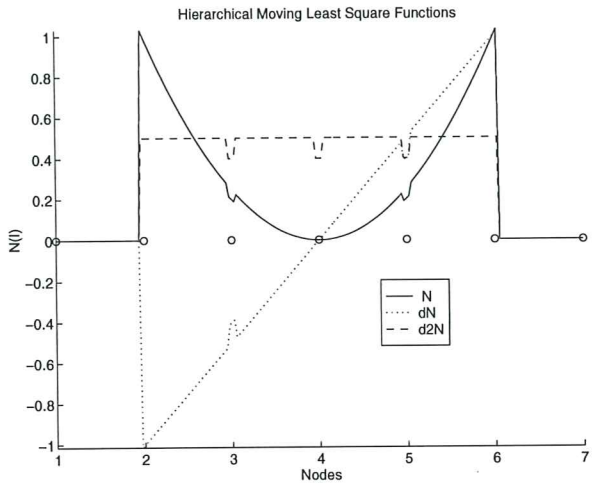


Figure 4.1c. Hierarchical Interpolation: $x^2 N_4^0$. Type 1 Weight Function, $r_k = 2.05$

The results for $N_4^0$ produced by the hat type 2 weight function are shown in Figure 4.2a. Figures 4.2b and 4.2c show results for $xN_4^0$ and $x^2 N_4^0$, respectively.

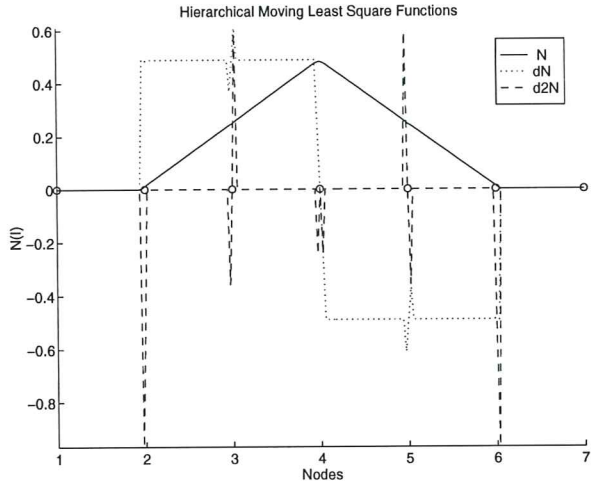


Figure 4.2a. Shepard Interpolation: $N_4^0$. Type 2 Weight Function, $r_k = 2.05$

The results for $N_4^0$ produced by the truncated Gaussian type 3 weight function are shown in Figure 4.3a. Figures 4.3b and 4.3c show results for $xN_4^0$ and $x^2 N_4^0$, respectively.

The results for $N_4^0$ produced by the $C_0^m$ type 4 weight function are shown in Figure 4.4a. The parameter $m$ is set to 4. Figures 4.4b and 4.4c show results for $xN_4^0$ and $x^2 N_4^0$, respectively.

It is evident from the above results that the use of smooth weighting functions is necessary to produce interpolations free from discontinuities in the derivatives. For example, use of type 4 weight function with an exponent $m$ greater than 3 and span greater than 4 together with quadratic polynomial interpolations will produce moving least square functions $N_k^p, p = 2$ with continuous second derivatives. The required span of the interpolation

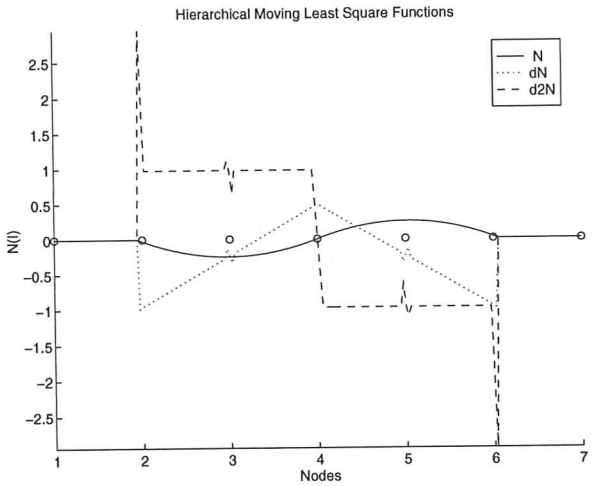


Figure 4.2b. Hierarchical Interpolation: $xN_4^0$. Type 2 Weight Function, $r_k = 2.05$

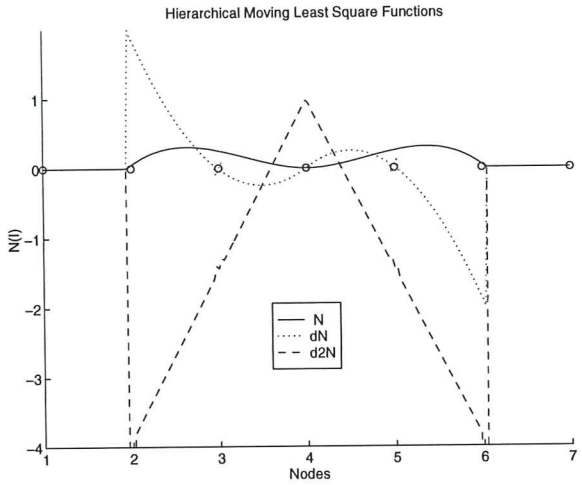


Figure 4.2c. Hierarchical Interpolation: $x^2N_4^0$. Type 2 Weight Function, $r_k = 2.05$

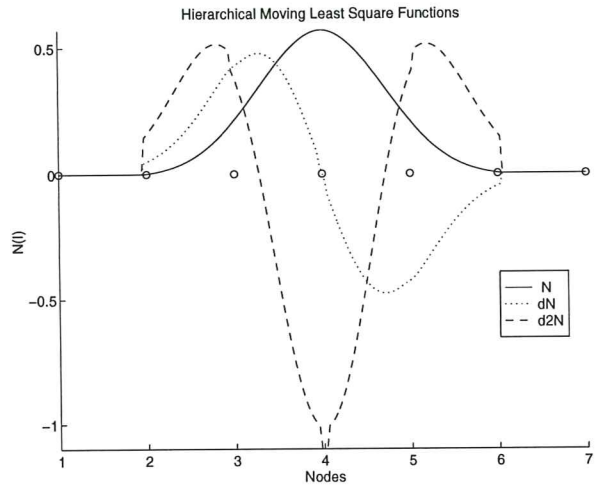


Figure 4.3a. Shepard Interpolation: $N_4^0$. Type 3 Weight Function, $r_k = 2.05$

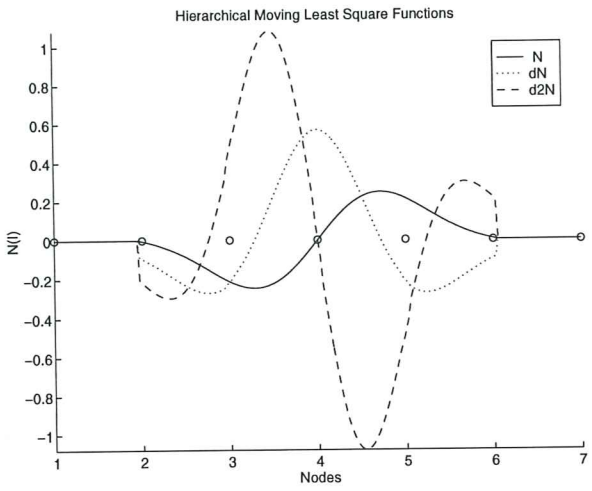


Figure 4.3b. Hierarchical Interpolation: $xN_4^0$. Type 3 Weight Function, $r_k = 2.05$

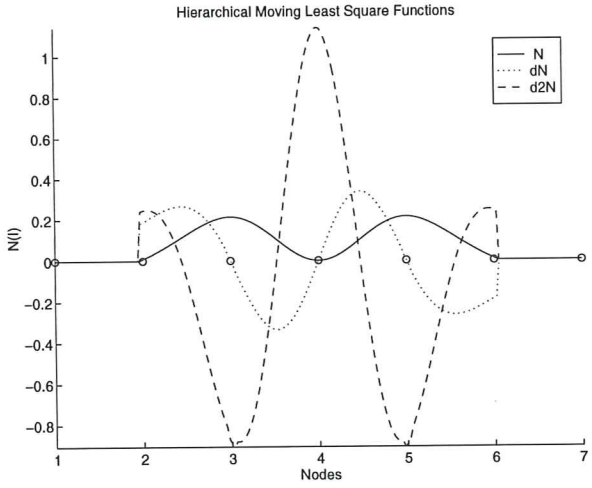


Figure 4.3c. Hierarchical Interpolation: $x^2N_4^0$. Type 3 Weight Function, $r_k = 2.05$

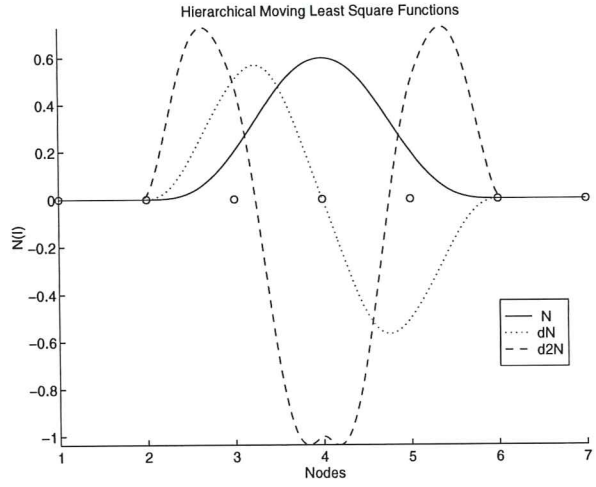


Figure 4.4a. Shepard Interpolation: $N_4^0$. Type 4 Weight Function, $r_k = 2.05$

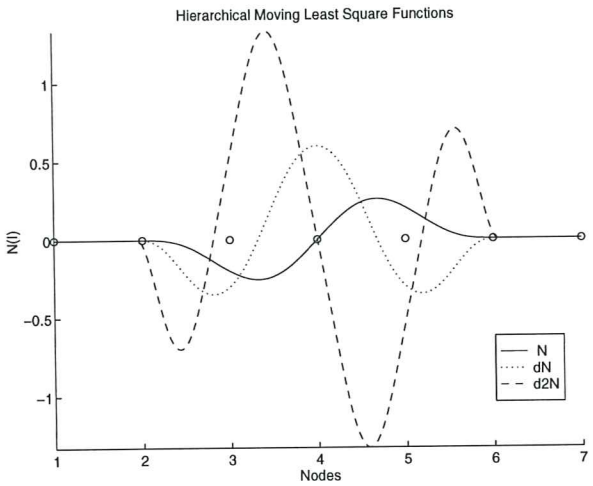


Figure 4.4b. Hierarchical Interpolation: $xN_4^0$. Type 4 Weight Function, $r_k = 2.05$

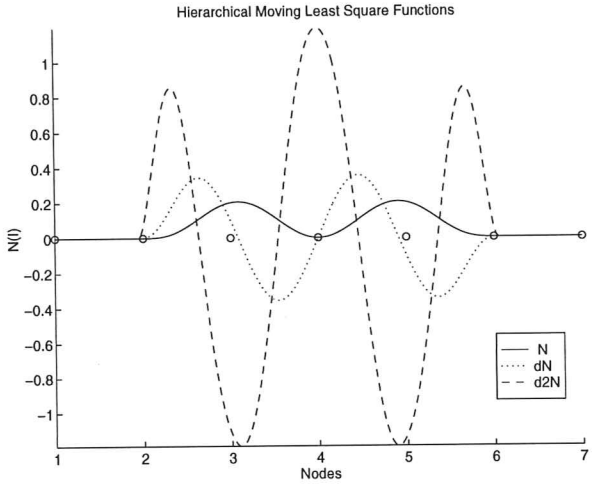


Figure 4.4c. Hierarchical Interpolation: $x^2N_4^0$. Type 4 Weight Function, $r_k = 2.05$

may be reduced using the hierarchical representation. Thus, the span of the functions based on Shepard interpolation (i.e., $N_k^0(x)$) need only be greater than 1 ($r_k > 0.5$). Using a span 1.2 (i.e., $r_k = 0.6$) for type 4 weight functions produces the quadratic functions shown in Figures 4.5a to 4.5c.

The extremely sharp discontinuities near the ends of the interpolation lead to very large values for derivatives which renders approximation techniques sensitive to points used in the calculations, as well as producing flat spots. For example, Galerkin methods may require high order quadrature formula to produce good results (e.g., see results in Section 6). Collocation methods may be equally sensitive. Increasing the span to 2.01 ($r_k = 1.005$) produces the results shown in Figures 4.6a to 4.6c. The increased smoothness of the interpolations produced by the larger span is clearly evident.

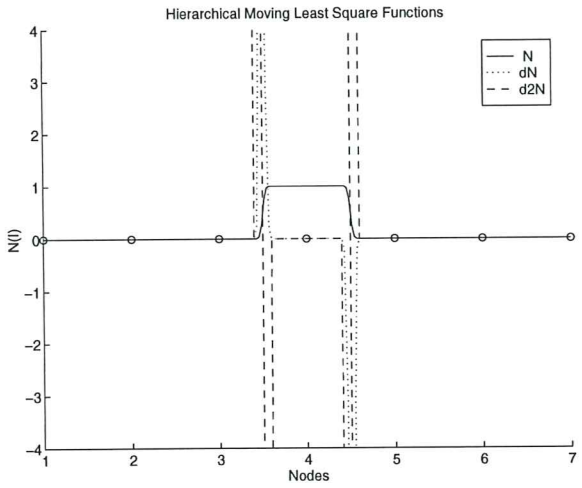


Figure 4.5a. Shepard Interpolation: $N_4^0$. Type 4 Weight Function, $r_k = 0.6$

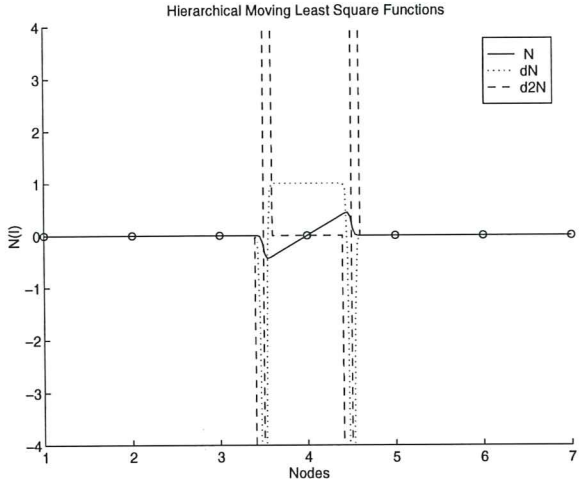


Figure 4.5b. Hierarchical Interpolation: $xN_4^0$. Type 4 Weight Function, $r_k = 0.6$

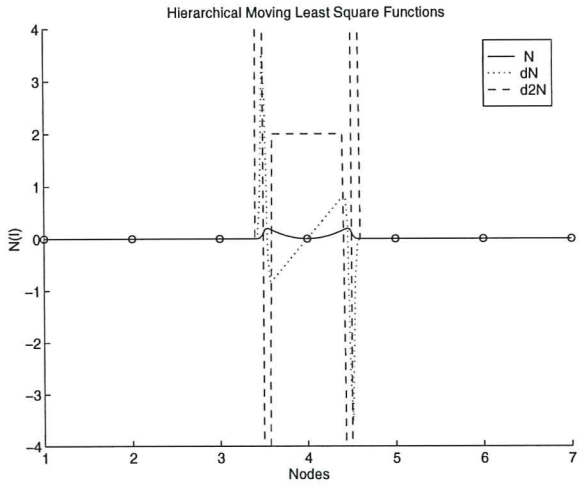


Figure 4.5c. Hierarchical Interpolation: $x^2N_4^0$. Type 4 Weight Function, $r_k = 0.6$

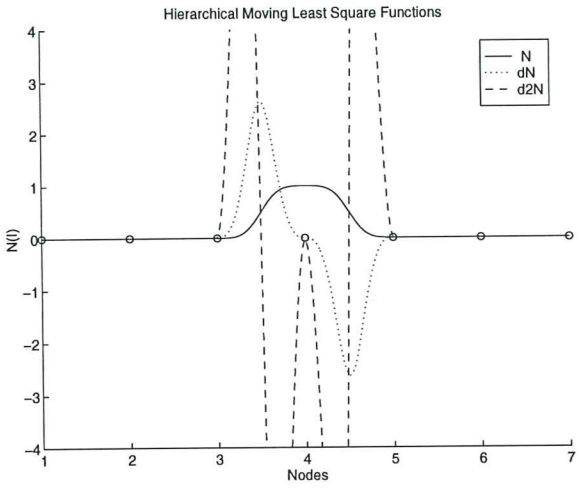


Figure 4.6a. Shepard Interpolation: $N_4^0$. Type 4 Weight Function, $r_k = 1.005$

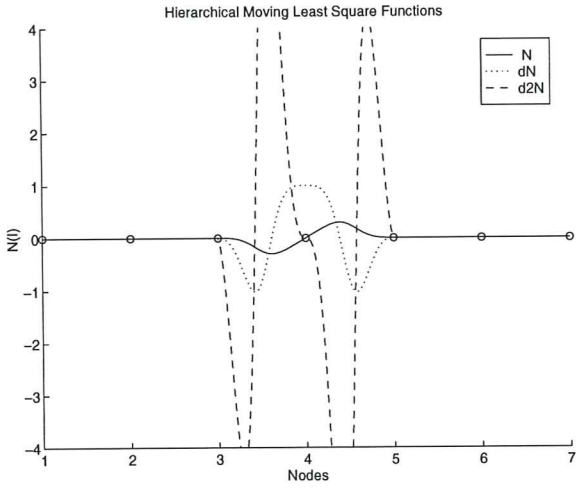


Figure 4.6b. Hierarchical Interpolation: $xN_4^0$. Type 4 Weight Function, $r_k = 1.005$

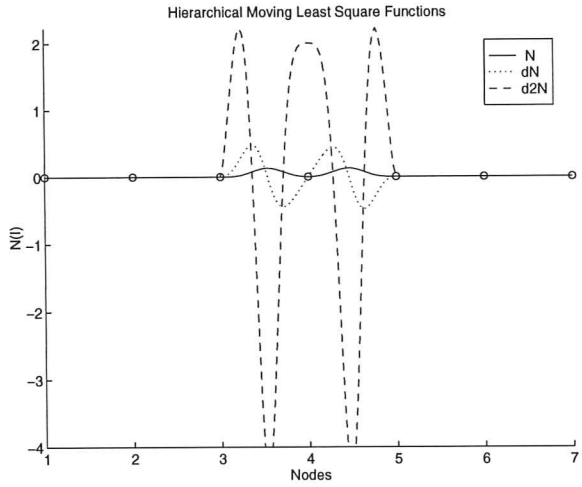


Figure 4.6c. Hierarchical Interpolation: $x^2N_4^0$. Type 4 Weight Function, $r_k = 1.005$

## 5. Solution of ODE by a Finite Point Method

To assess the performance of moving least square approximations described in Sections 2 and 3, we consider the solution of second order ordinary differential equations using a finite point method [15]. Accordingly, the differential equation is taken as

$$- a \frac{d^2u}{dx^2} + b \frac{du}{dx} + c\, u = f(x)$$

on the domain $0 < x < L$ with constant coefficients $a$, $b$, $c$, and subject to the boundary conditions

$$u(0) = g_1$$

and

$$u(L) = g_2$$

The domain is divided into a set of points located at $x_i$, $i = 1, n$. The moving least square approximation described in Section 2 is used to write difference equations at each of the interior points (i.e., $i = 2, n-1$). The boundary conditions are also written in terms of discrete approximations using the moving least square approximation. Accordingly, for the approximate solution using $p$-order polynomials to define the $\boldsymbol{p}(x)$ in the interpolations

$$\hat{u}(x) = \sum_{j=1}^{n} N_i^p(x)\, u_i$$

we have the set of $n$-equations in $n$-unknowns:

$$\sum_{i=1}^{n} N_i^p(x_1)\, u_i = g_1$$

$$\sum_{i=1}^{n} \left( - a\frac{d^2 N_i^p}{dx^2} + b\frac{d^2 N_i^p}{dx^2} + cN_i^p \right)_{x=x_j} u_i = f(x_j) \quad ; \qquad j = 2, n-1$$

and

$$\sum_{i=1}^{n} N_i^p(x_n)\, u_i = g_2$$

A solution using the approximate interpolation functions as first suggested by Nayroles et.al. may be constructed by replacing the derivatives of $N_i^p(x_j)$ by those of $\hat{N}_i^p(x_j)$ as defined in Section 2. We have noted, however, that the extra cost of including the terms involving the derivatives of $\phi_j$ is not large compared to other parts of a solutions. Accordingly, all results reported here retain all terms in the derivatives.

The discrete equations for the differential equation and boundary conditions may be written in matrix form as:

$$\boldsymbol{K}\,\boldsymbol{u} \;=\; \boldsymbol{f}$$

where $\boldsymbol{K}$ is a square coefficient matrix, $\boldsymbol{f}$ is a load vector consisting of the entries from $g_i$ and $f(x_j)$, and $\boldsymbol{u}$ is the vector of unknown parameters defining the approxiate solution $\hat{u}(x)$. A unique solution to this set of equations requires $\boldsymbol{K}$ to be non-singular (i.e., $rank(\boldsymbol{K}) = n$). The rank of $\boldsymbol{K}$ depends both on the weighting function used to construct the least square approximation as well as the number of functions used to define the polynomials $\boldsymbol{p}$. In order to keep the least square matrices as well conditioned as possible, we use the shifted polynomial approximation

$$\boldsymbol{p}(x) = [\,1 \quad x - x_j \quad (x - x_j)^2 \quad \ldots \quad (x - x_j)^p\,]^T$$

to define the interpolations associated with $N_j^p(x)$. The matrix $\boldsymbol{K}$ will be of correct rank provided the weighting function can generate linearly independent equations. Since local polynomials are used, this will always be achieved provided that the span of the weighting function is appropriately defined.

As shown in the previous section, the accurate approximation of second derivatives requires use of at least quadratic order polynomials in $\boldsymbol{p}(x)$. In addition, the span of the weighting function must be sufficient to keep the least squares matrix $\boldsymbol{A}$ non-singular *at every point in the domain* and boundary points. Thus, the minimum span needed to define quadratic interpolations of $\boldsymbol{p}(x)$ (i.e., $p = k = 2$) must include at least three mesh points with non-zero contributions. If a weighting function which is zero at its boundary is placed so that one of its boundary points lies exactly on a node, any span which is bigger than three mesh spaces will be sufficient to have three non-zero terms in the difference quotient. At the problem boundaries, however, only half of the associated weighting function span will be used. Consequently, for weighting functions which go smoothly to zero at their boundary, a span larger than four mesh spaces is required. The span should not be made too large however, since the sparse structure of $\boldsymbol{K}$ will be lost. Furthermore, overly diffuse solutions may result.

Use of hierarchical interpolations reduces the required span of the weighting function. For example use of interpolations with $k = 0$ requires only a span at each point for which the domain is just covered (since any span will include its defining point, $x_k$, the $\boldsymbol{A}$ matrix will always be non-singular). For a uniformly spaced set of points this is any span greater than one mesh spacing.
For the results shown below, unless stated otherwise, we have used weighting function type 4 with a weight span 4.4 times the largest adjacent mesh space for the quadratic interpolations with $k = p = 2$ and a weight span 2.01 times the mesh space for the hierarchical quadratic interpolations with $k = 0 \;\; p = 2$.

**Example 1.**

As a first example consider the solution of the differential equation

$$-\frac{d^2u}{dx^2} = 1$$

in the domain $0 < x < 1$ with the boundary conditions

$$u(0) = u(1) = 0$$

The exact solution is given by

$$u(x) = \frac{1}{2}x(1-x)$$

and has a maximum value of 0.125 at the midspan. For quadratic order interpolations for $\boldsymbol{p}(x)$ the exact solution is recovered for all weighting functions with span greater than $4h_k$, where $h_k$ is the largest distance between adjacent mesh points $x_{k-1}$ and $x_{k+1}$. The solution is also recovered exactly when the approximate interpolation function derivatives are employed.

**Example 2.**

As a second example consider the solution of the convection-diffusion equation

$$-\frac{d^2u}{dx^2} + b\,\frac{du}{dx} = 0$$

in the domain $0 < x < 1$ with the boundary conditions

$$u(0) = 1; \quad \text{and} \quad u(1) = 0$$

The exact solution is given by

$$u(x) = \frac{exp(b) - exp(bx)}{exp(b) - 1}$$

For large $b$ the solution converges to a step from 1 to 0 near $x = 1$. A solution for $b = 10$ for a problem with 9 points is shown in Figure 5.1 and was computed using quadratic interpolations for each point (i.e., $k = p = 2$). The solution was then computed using 27 points and results are shown in Figure 5.2.

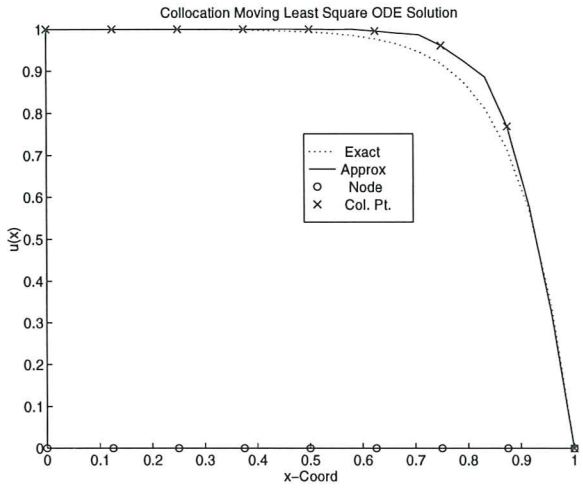


Figure 5.1. Convection Diffusion Solution: 9 Points with $k = p = 2$.

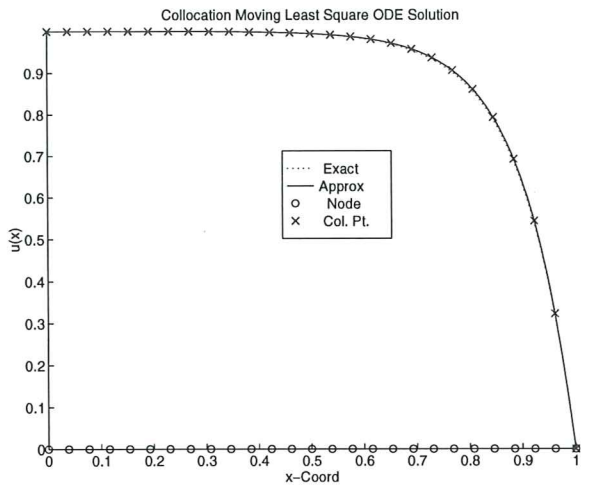


Figure 5.2. Convection Diffusion Solution: 27 Points with $k = p = 2$.

A quadratic approximate solution using the $p$-order hierarchical interpolation described in Section 3 leads to

$$\hat{u}(x) = \sum_{j=1}^{n} N_j^0(x)(u_j + \boldsymbol{q}_{02}\boldsymbol{b}_j^2)$$

where

$$\boldsymbol{q}_{02} = [\, x \quad x^2 \,]$$

and

$$\boldsymbol{b}_j^2 = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

Using this interpolation in the differential equation and boundary conditions we have:

$$\sum_{i=1}^{n} N_i^0(x_1)\ (u_i + \boldsymbol{q}_{02}\boldsymbol{b}_j^2) = g_1$$

$$\sum_{i=1}^{n} \left( -\ a\ \frac{d^2 N_i^0}{dx^2} + b\ \frac{d^2 N_i^0}{dx^2} + cN_i^0 \right)_{x=x_j} (u_i + \boldsymbol{q}_{02}\boldsymbol{b}_i^2) = f(x_j) \quad ; \qquad j = 2, nn-1$$

and

$$\sum_{i=1}^{n} N_i^0(x_n)\ (u_i + \boldsymbol{q}_{02}\boldsymbol{b}_i^2) = g_2$$

For a $p$-order polynomial approximation the total number of parameters at each point is $p - k + 1$ (i.e., $p - k$ parameters in each $\boldsymbol{b}_i^p$ and the $u_i$ parameter); thus, for a quadratic interpolation there are three parameters per point. Due to the presence of the additional parameters it is necessary to construct the difference equation for $p$-order interpolations at a total of $n(p - k + 1) - 2$ points instead of $n$ points. Thus, it is now necessary in the finite point collocation to select additional points for approximation. In the results shown

below the points have been spaced equally within each interval defined half-way between the point immediately to the left and the right of the $i$-point. For the boundary points the points are placed within the half interval near the end. This selection ensures that no two points have the same coordinate and thus ensures that the approximations remain linearly independent.

Repeating the solution for Example 1 using $N_i^0$ (i.e., Shepard approximation) with quadratic hierarchical functions given by $\boldsymbol{q}_{02}$ again produces exact results. Repeating the solution for Example 2 using the same type of approxaimation produces the results shown in Figure 5.3 for 3 points (9 parameters) and in Figure 5.4 for 9 points (27 parameters). The location of each collocation point is shown by an $\times$ in the figures.

While quite accurate results are obtained for this example by using a sufficient number of parameters, use of yet larger values of $b$ requires the introduction of some form of numerical dissipation to mitigate unwanted oscillations in the approximate solution. This has been noted previously for finite point methods by Oñate et. al. [15].

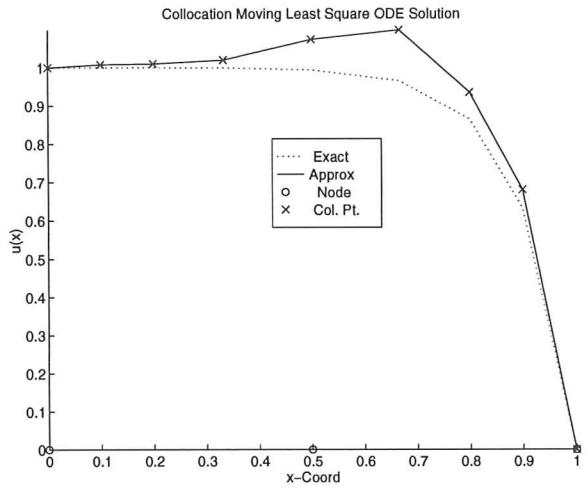


Figure 5.3. Convection Diffusion Solution: 3 Points $k = 0$ $p = 2$.

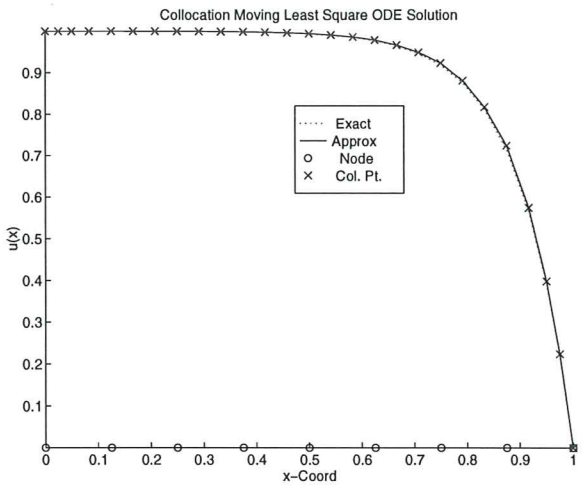


Figure 5.4. Convection Diffusion Solution: 9 Points $k = 0 \;\; p = 2$.

**Example 3.**

As a third example consider the solution of the differential equation representing a string on an elastic foundation. Accordingly,

$$-\, a\, \frac{d^2u}{dx^2} + c\, u = f$$

in the domain $0 < x < 1$ with the boundary conditions

$$u(0) = u(1) = 0$$

The exact solution is given by

$$\frac{u(x)}{u_p} = 1 - cosh(mx) + (1 - cosh(m))\frac{sinh(mx)}{sinh(m)}$$

where for constant loading $f$ the particular solution is

$$u_p = \frac{f}{c}$$

The data for the solution is taken as

$$a = 0.01 \;\; c = f = 1$$

The problem is solved using the finite point method with 27 points and $k = p = 2$ producing the results shown in Figure 5.5.

The process was repeated using the hierarchical interpolations with $k = 0$ and $p = 2$ using 9 points (which results in 27 parameters - the same as for the first case). The results are shown in Figure 5.6. The hierarchical interpolations permit the solution to be obtained using as few as two points. A solution with two points using an interpolations with $k = 0$ and $p = 3$ and 5 are shown in Figures 5.7 and 5.8.

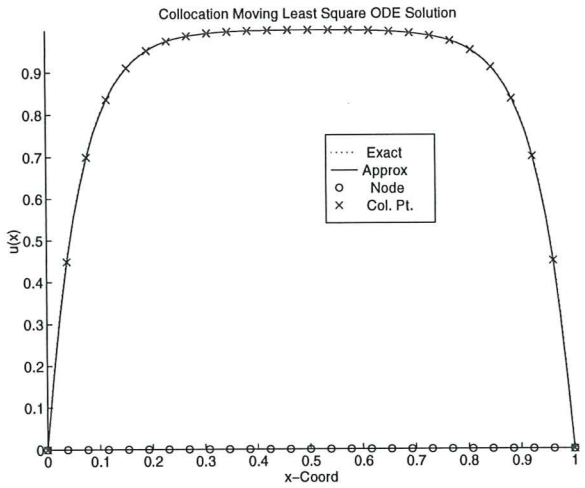


Figure 5.5. String on Elastic Foundation Solution: 27 Points, $k = p = 2$

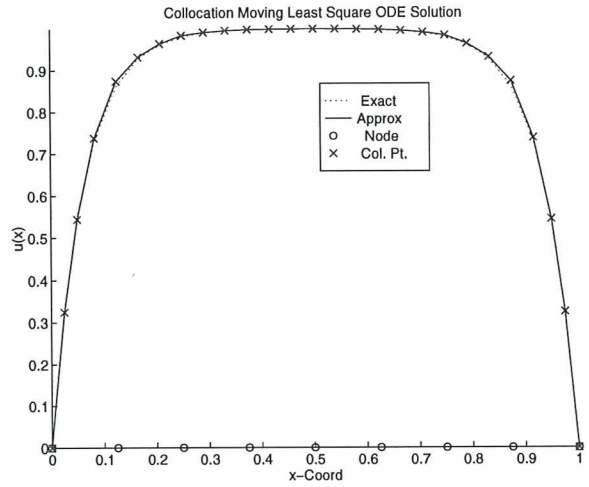


Figure 5.6. String on Elastic Foundation Solution: 9 Points, $k = 0 \;\; p = 2$

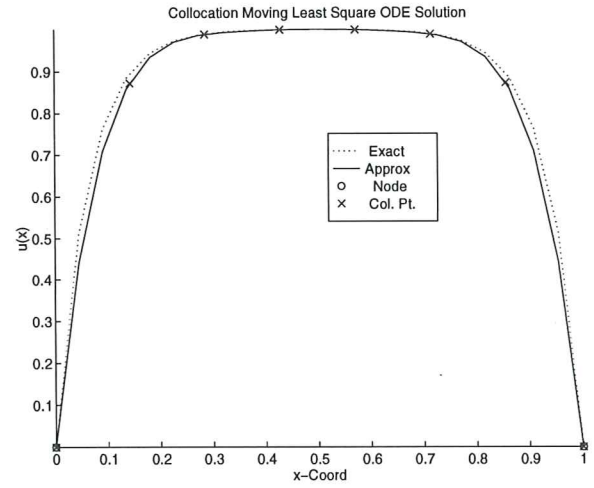


Figure 5.7. String on Elastic Foundation Solution: 2 Points, $k = 0 \;\; p = 3$

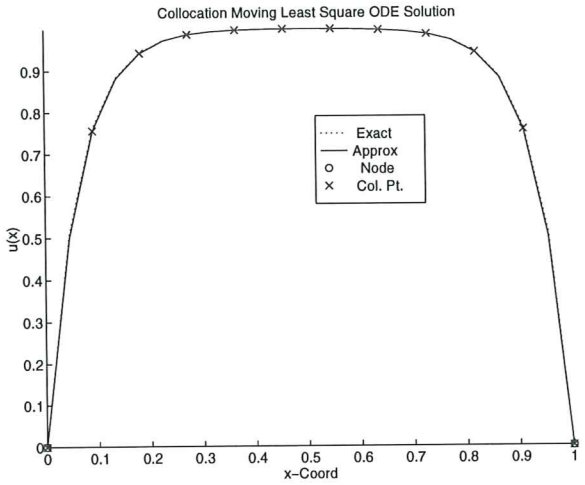


Figure 5.8. String on Elastic Foundation Solution: 2 Points, $k = 0$ $p = 5$

## 6. Solution of ODE by a Galerkin Method

The moving least square approximations described in Sections 2 and 3 may also be used in a Galerkin method to solve second order ordinary differential equations. For an arbitrary function $W(x)$, a weak form for the differential equation may be written as:

$$\int_0^L \left[ \frac{dW}{dx}\, a\, \frac{du}{dx} + W\left( b\, \frac{du}{dx} + c\, u - f(x) \right) \right] dx = 0$$

subject to the boundary conditions

$$u(0) = g_1$$

and

$$u(L) = g_2$$

The boundary conditions are included using a Lagrange multiplier approach where the weak form is given by:

$$\int_0^L \left[ \frac{dW}{dx}\, a\, \frac{du}{dx} + W\left( b\, \frac{du}{dx} + c\, u - f(x) \right) \right] dx$$

$$+ W(0)\,\lambda_1 \;+\; W(L)\,\lambda_2 \;+\; W_1\,[u(0) - g_1] \;+\; W_2\,[u(L) - g_2] \;=\; 0$$

A $p$-order polynomial approximation to the dependent variable may be taken as the hierarchical moving least square functions

$$\hat{u}(x) = \sum_{j=1}^{n} N_j^0(x) \bar{\boldsymbol{q}}_{jp} \boldsymbol{u}_j^p$$

where

$$\bar{\boldsymbol{q}}_{jp} = [\,1 \quad x - x_j \quad (x - x_j)^2 \quad \ldots \quad (x - x_j)^p\,]$$

The weight functions are taken as the same functions, thus,

$$\hat{W}(x) = \sum_{j=1}^{n} N_j^0(x) \bar{\boldsymbol{q}}_{jp} \boldsymbol{W}_j^p$$

in which $\boldsymbol{W}_j^p$ are arbitrary parameters. Substituting the approximations into the weak form yields the discrete problem

$$\sum_{i=1}^{n} (\boldsymbol{W}_i^p)^T \sum_{j=1}^{n} \left\{ \int_0^L \left[ \frac{d(\bar{\boldsymbol{q}}_{ip}^T N_i^0)}{dx} \, a \, \frac{d(\bar{\boldsymbol{q}}_{jp} N_j^0)}{dx} + \bar{\boldsymbol{q}}_{ip}^T N_i^0 \left( b \, \frac{d(\bar{\boldsymbol{q}}_{jp} N_j^0)}{dx} + c \, \bar{\boldsymbol{q}}_{jp} N_j^0 \right) \right] dx \right\} \boldsymbol{u}_j^p$$

$$\sum_{i=1}^{n} (\boldsymbol{W}_i^p)^T \left[ N_i^0(0) \, \bar{\boldsymbol{q}}_{ip}^T(0) \, \lambda_1 + N_i^0(L) \, \bar{\boldsymbol{q}}_{ip}^T(L) \, \lambda_2 \right] = \sum_{i=1}^{n} (\boldsymbol{W}_i^p)^T \int_0^L \bar{\boldsymbol{q}}_{ip}^T N_i^0 \, f(x) dx$$

and the boundary equations

$$W_1 \sum_{j=1}^{n} N_j^0(0) \, \bar{\boldsymbol{q}}_{jp}(0) \, \boldsymbol{u}_j^p = W_1 \, g_1$$

$$W_2 \sum_{j=1}^{n} N_j^0(L) \, \bar{\boldsymbol{q}}_{jp}(L) \, \boldsymbol{u}_j^p = W_2 \, g_2$$

Since $\boldsymbol{W}_i^p$, $W_1$ and $W_2$ are arbitrary, the solution to the approximate weak form yields the set of equations

$$\sum_{j=1}^{n} \left\{ \int_0^L \left[ \frac{d(\bar{\boldsymbol{q}}_{ip}^T N_i^0)}{dx} a \, \frac{d(\bar{\boldsymbol{q}}_{jp} N_j^0)}{dx} + \bar{\boldsymbol{q}}_{ip}^T N_i^0 \left( b \, \frac{d(\bar{\boldsymbol{q}}_{jp} N_j^0)}{dx} + c \, \bar{\boldsymbol{q}}_{jp} N_j^0 \right) \right] dx \right\} \boldsymbol{u}_j^p$$

$$N_i^0(0) \, \bar{\boldsymbol{q}}_{ip}^T(0) \, \lambda_1 + N_i^0(L) \, \bar{\boldsymbol{q}}_{ip}^T(L) \, \lambda_2 = \int_0^L \bar{\boldsymbol{q}}_{ip}^T N_i^0 \, f(x) dx \quad ; \quad i = 1, 2, \ldots, n$$

and the two boundary equations

$$\sum_{j=1}^{n} N_j^0(0) \, \bar{\boldsymbol{q}}_{jp}(0) \, \boldsymbol{u}_j^p = g_1$$

$$\sum_{j=1}^{n} N_j^0(L) \, \bar{\boldsymbol{q}}_{jp}(L) \, \boldsymbol{u}_j^p = g_2$$

which need to be solved for the parameters $\boldsymbol{u}_j^p$ and Lagrange multipliers $\lambda_1$ and $\lambda_2$. In matrix notation the discrete problem now has the form

$$\begin{bmatrix} \boldsymbol{K} & \boldsymbol{C}^T \\ \boldsymbol{C} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{bmatrix}$$

For differential equations where the parameter $c$ is zero the matrix $\boldsymbol{K}$ is singular (it possesses one singular mode of $\boldsymbol{u}$ constant); however, the total matrix equation still has a solution. If it is desired to solve the equations without resorting to pivoting the alternative regularized form

$$\begin{bmatrix} (\boldsymbol{K} + \gamma \boldsymbol{C}^T \boldsymbol{C}) & \boldsymbol{C}^T \\ \boldsymbol{C} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{bmatrix}$$

may be used. The value of $\gamma$ may be chosen as any positive number.

The Galerkin form requires only first derivatives of the approximating functions as opposed to the second derivatives required for the finite point method. This reduction, however, is accompanied by a need to perform integrals over the domain. For the type 4 weighting functions all functions entering the approximation are polynomials and rational polynomial expressions, thus, a closed form evaluation is difficult at best. Accordingly, in the present study we evaluate the integrals using Gauss and Gauss-Lobatto quadrature over each the intervals generated by the basis points in the moving least square representation ($x_j$ for $j = 1, 2, \ldots, n$). As an example of the type of solutions possible we consider the string on elastic foundataion problem given as Example 3 in the previous section. For the property $a = 0.004$, $c =$ with loading $f = 1$ and zero boundary conditions a Galerkin solution using 3 and 4 point Gauss quadrature and 4 and 5 point Gauss-Lobatto quadrature is shown in Figures 6.1 to 6.4. A mesh consisting of 9 equally spaced points is used to define the intervals for the solution and quadrature. The weight function is generated for $k = 0$ $p = 2$ with a span of 4.4 mesh points.

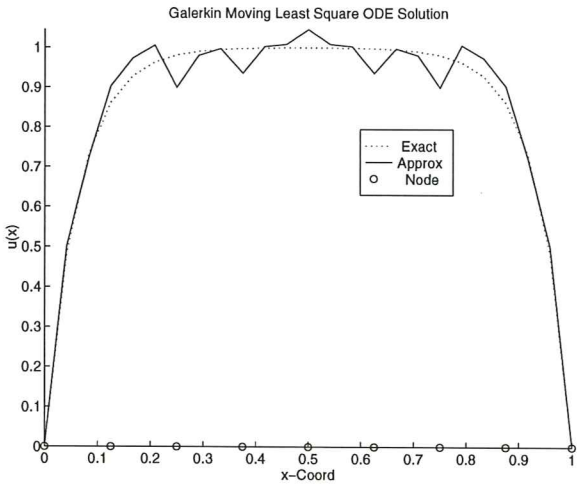


Figure 6.1. String on Elastic Foundation Solution: 3-Point Gauss Quadrature.

The above sequence of problems was repeated with a weight having a span of 2.1 mesh spaces, which produces a set of $N_j^0(x)$ that just overlap each other. The results are shown in Figures 6.5 to 6.8. In general the smaller span produces answers which are somewhat less sensitive to the quadrature. However, it is evident that the answers for a nine point mesh depend on accurate evaluation of integrals to produce acceptable answers.

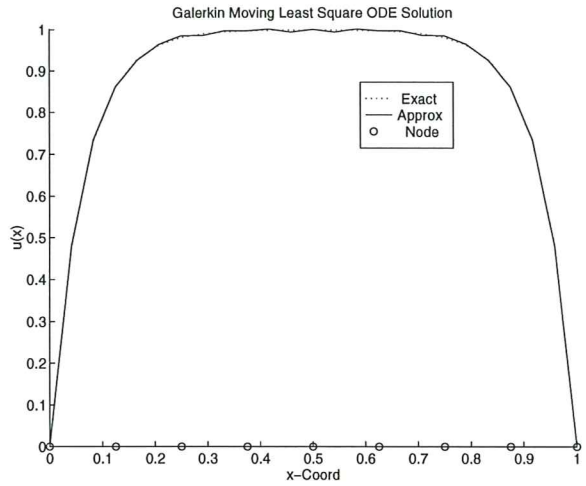


Figure 6.2. String on Elastic Foundation Solution: 4-Point Gauss Quadrature.

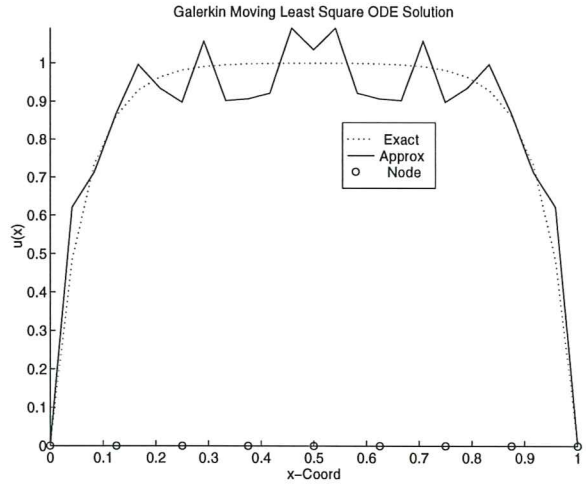


Figure 6.3. String on Elastic Foundation Solution: 4-Point Gauss-Lobatto Quadrature.

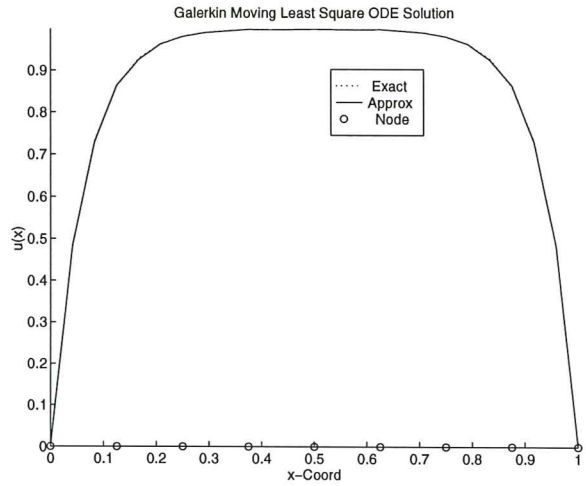


Figure 6.4. String on Elastic Foundation Solution: 5-Point Gauss-Lobatto Quadrature.

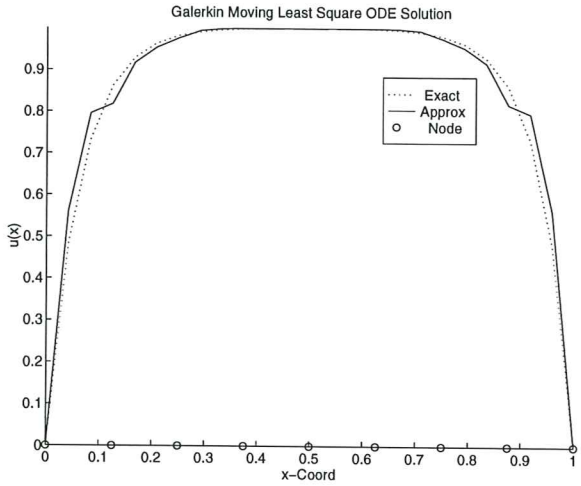


Figure 6.5. String on Elastic Foundation Solution: 3-Point Gauss Quadrature.

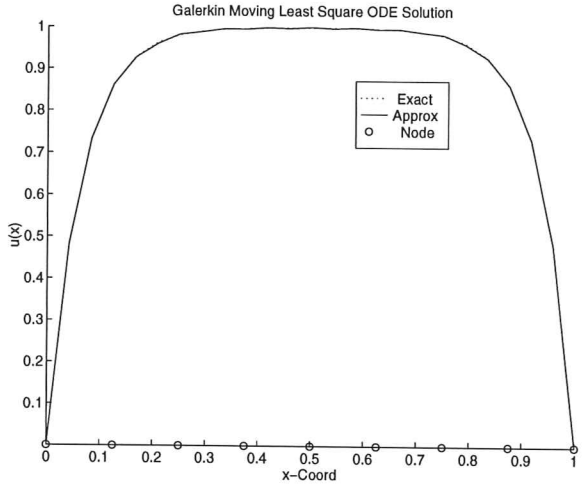


Figure 6.6. String on Elastic Foundation Solution: 4-Point Gauss Quadrature.

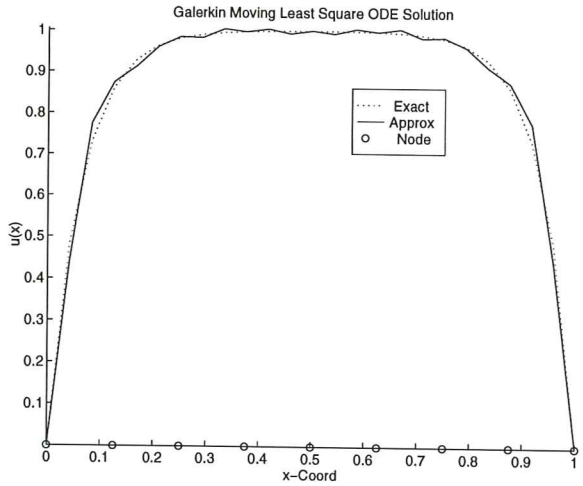


Figure 6.7. String on Elastic Foundation Solution: 4-Point Gauss-Lobatto Quadrature.

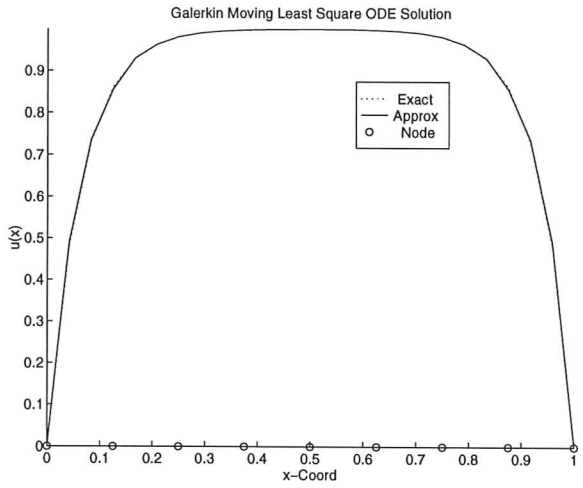


Figure 6.8. String on Elastic Foundation Solution: 5-Point Gauss-Lobatto Quadrature.

## 7. Closure

In this report we have summarized some of the properties of moving least squares approximations to functions in one variable. The use of MLS to generate shape functions based on polynomial basis functions has been presented for the case where the moving weighting function is approximated by different fixed weighting functions associated with each point of the interpolation. The use of hierarchical enhancement of moving least squares based on a partition of unity concept as described by Babuška and Melenk [1] and Duarte and Oden [6] in conjunction with four types of weighting functions has also been considered. Based on our evaluations using the four weighting function forms, we can conclude, that to attain continuous functions and derivatives, functions with $C_0^q$ continuity on a ball of radius $r_k$ should have a $q$ value at least one greater than the number of derivatives to be computed in collocating a differential equation or in a Galerkin weak form of the differential equation.

Using the Type 4 polynomial based weighting functions described in Section 4 with the parameter $m$ selected as four solutions for simple ordinary differential equation are constructed using a point collocation method (Section 5) and a Galerkin method (Section 6). Based on this study the results suggest that the point collocation method, also called finite point method, has less sensitivity in selection of the approximation points than the Galerkin scheme. Furthermore, the need for high quadrature order in the Galerkin scheme renders it far more costly in generating the discrete equations than that for the finite point method.

Hierarchical interpolation greatly reduces the cost for computing the weighted least square approximations. Indeed, using Shepard interpolations it is only necessary to invert a $1 \times 1$ matrix. Hierarchical interpolation also has the ability to eliminate the *flat spots* often observed in Shepard interpolations (e.g., see Figure 4.5a). On the negative side, however, is a need to generate additional points (than the $x_k$ points needed for the least square approximation) for use with the finite point method, whereas a Galerkin method does not present such difficulty. The generation of the additional points must ensure that no linear dependence (or near linear dependence) results in the set of equations to be

solved for the unknown parameters defining the solution to the differential equation and its boundary conditions. Work is in progress to investigate various alternatives for generating optimally located additional points and to apply the method to problems in two and three dimensions.

## References


1. Babuška, I and J.M. Melenk. The Partition of Unity Finite Element Method, Technical Note BN-1185, Institute for Physical Science and Technology, University of Maryland, April 1995.

2. Batina, J.T. A Gridless Euler/navier Stokes Solution Algorithm for Complex-Aircraft Applications, *AIAA 93-0333*, 31st Aerospace Science Meeting, Reno, Nevada, Jan 1993.

3. Belytschko, T., Y.Y. Lu and L. Gu. Element-Free Galerkin Methods, *Int. J. Num. Meth. Engrg.*, **37**, 229-256, 1994.

4. Belytschko, T., Y.Y. Lu and L. Gu. Crack Propagation by Element-Free Galerkin Methods, *Engrg. Fracture Mech.*, **51**, 295-315, 1995.

5. Duarte, C.A. A Review of Some Meshless Methods to Solve Partial Differential Equations, TICAM Report 95-06, The University of Texas, May 1995.

6. Duarte, C.A. and J.T. Oden. *Hp* Clouds - A Meshless Method to Solve Boundary Value Problems, TICAM Report 95-05, The University of Texas, May 1995.

7. Lancaster, P. and K. Salkauskas. Surfaces Generated by Moving Least Squares Methods, *Mathematics of Computation*, **37**, 141-158, 1981.

8. Liszka, T. An Interpolation Method for an Irregular Net of Nodes, *Int. J. Num. Meth. Engrg.*, **20**, 1599-1612, 1984.

9. Liszka, T. and J. Orkisz. The Finite Difference Method at Arbitrary Irregular Grids and its Application in Applied Mechanics, *Comp. and Struct.*, **11**, 83-95, 1980.

10. Liu, W.K., S. Jun and Y.F. Zhang. Reproducing Kernel Particle Methods, *Int. J. Num. Meth. in Fluids*, **20**, 1081-1106, 1995.

11. Liu, W.K., S.J. Jun, S. Li, J. Adee and T. Belytschko. Reproducing Kernel Particle Methods for Structural Dynamics, *Int. J. Num. Meth. Engrg.*, **38**, 1655-1679, 1995.

12. Liu, W.K., S. Li and T. Belytschko. Moving Least Square Kernel Galerkin Method (I) Methodology and Convergence, Report Tech-ME-95-3-XX, Northwestern University, March 1995.

13. Liu, W.K., Y. Chen, S. Jun, J.S. Chen, T. Belytschko, C. Pan, R.A. Uras and C.T. Chang. Overview and Applications of the Reproducing Kernel Particle Methods, (manuscript), August 1995.

14. Lu, Y.Y., T. Belytschko and L. Gu. A New Implementation fo the Element-Free Galerkin Method, *Comp. Meth. in Appl. Mech. Engrg.*, **113**, 397-414, 1994.

15. Oñate, E., S. Idelsohn, and O.C. Zienkiewicz. Finite Point Methods in Computational Mechanics, CIMNE Report 67, International Center for Numerical Methods in Engineering, Barcelona, Spain, July 1995.

16. Orkisz, J. and M. Pazanowski. Analysis of Residual Stresses by the Generalized Finite Difference Method, *Proc. 4th. Int. Conf. Comp. Plast.*, Barcelona, Spain, 189-200, 1995.

17. Nayroles, B., G. Touzot, and P. Villon. La méthode des éléments diffus, *C.R. Acad. Sci. Paris*, **313**, Série II, 133-138, 1991.

18. Nayroles, B., G. Touzot, and P. Villon. L'approximation diffuse, *C.R. Acad. Sci. Paris*, **313**, Série II, 293-296, 1991.

19. Nayroles, B., G. Touzot, and P. Villon. Generalizing the Finite Element Method: Diffuse Approximations and Diffuse Elements, *Computer Mechanics*, **10**, 307-318, 1992.

20. Rudin, W. *Principles of Mathematical Analysis*, 3rd. ed., McGraw Hill, 1976.

21. Shepard, D. A Two-Dimensional Interpolation Function for Irregularly Spaced Points, *Procedings ACM National Conference*, 517-524, 1968.

22. Tabbara, M., T. Blacker and T. Belytschko. Finite Element Derivative Recovery by Moving Least Square Interpolates, *Comp. Meth. in Appl. Mech. Engrg.*, **117**, 211-223, 1994.