MESHFREE SHAPE FUNCTION AND DERIVATIVES IN ONE DIMENSION WITH ARBITRARY NODAL DISTRIBUTION: MATLAB PROGRAMMING

Jeetender Singh Kushawaha[1] and Bal Krishna Misra[2]

Received: August 24, 2012; Revised: July 7, 2013; Accepted: July 8, 2013

Abstract

The paper is intended for the beginners on meshfree methods, to present the detailed matlab programming aspects for the construction of the moving least square approximation shape function and their derivatives in onedimension, with arbitrary nodal distribution, using the MATLAB code provided at the appendix. The condition of not a number (NaN) during the execution of MATLAB program has been given the vital attention and elaborated by presenting the related plots of shape function and its derivatives. The program has been further extended to solve and compare the results of an elastostatics problem, using the exact and element free galerkin method so-lutions.

Keywords: Meshfree, weight function, shape function, MLS approximation, basis function, matlab

Introduction

The development of the approximate methods for the numerical solution of practical problems, representable by partial differential equations has helped engineers, physicists and mathematicians in analyzing the complex phenomena at reduced costs. The finite element method (FEM) is one of the most popular, well-developed and possessing much versatility in analyzing complicated phenomena, whose behavior is governed by increasingly complex partial differential equations.

In recent years, meshfree methods have been developed as an alternative numerical tool in effort to eliminate known drawbacks of the finite element methods. The nature of the various approximation functions used by meshfree methods allows the representation of the problem domains by simply adding or de-leting nodes where-ever desired. The prior knowledge of nodal connectivity to form a discrete element as in finite element methods is not necessary, only nodal coordinates and their domain of influence are sufficient to represent the problem domain.

There are several meshfree methods under current development, including the most versatile element free Galerkin (EFG)

[1] *Research Scholar, Teerthankar Mahaveer University, Working in the Department of Mechanical Engineering, Indus Institute of Technology and Management, Kanpur, Pin-209202, India.*

[2] *Director, Rama Institute of Engineering and Technology, Kanpur, Pin-209217, India.*

* *Corresponding author*

method proposed by Belytschko *et al.* (1994), the Reproducing Kernel Particle Method (RKPM) proposed by Liu *et al.* (1995), Smooth Particle Hydrodynamics (SPH) method proposed by Gingold and Monaghan, Meshless Local Petrov-Galerkin (MLPG) method proposed by Atluri *et al.* (1999) and many other methods (Liu 2003). The well-established EFG and MLPG method use the shape functions which are derived from moving least-square approximation. The main purpose of this paper is to elaborate the construction of meshfree shape function using the MLS approximation using the arbitrary nodal distribution and implementation in matlab.

Meshfree Shape Function

The meshfree shape function is the central and most important issue and main differentiating point for the meshfree methods from the finite element methods. There are a number of ways proposed to construct the meshfree shape functions by Belytschko *et al.* (1994). In this paper the finite seriesrepresentation, moving least square approximation method is studied and elaborated considering the programming and implementation aspects, for arbitrary distribution of nodes, for the uniform distribution of nodes one may refer to Kushawaha (2012), and the different plots have been included to indicate the different steps and effects of various parameters exclusively.

A good meshfree shape functions needs to satisfy the following conditions:

1. A compulsory condition for the shape function is the satisfaction of partition of unity.

$$\sum_{i=1}^{n} \phi_i(x) = 1 \tag{1}$$

2. It should be able to manage the reasonable randomness of distribution of nodes.

3. The algorithm should be numerically stable.

4. The shape function constructed should have the consistency to enable the convergence of numerical results with increase of nodes.

5. The domain of influence should be compact.

6. The Kronecker-delta property should be satisfied.

7. The computational efficiency should be at par with FEM.

Ideally, the field approximation should be compatible throughout the problem domain.

Meshfree Shape Function Construction

All Moving least square (MLS) was originated by mathematicians for data fitting and surface construction, the procedure for constructing the meshfree shape function using the MLS approximation starts with the assumption that x_1, x_2, x_3, x_4, and x_n are the nodes distributed in the domain Ω and the associated field variable or nodal parameter with these node are u_1, u_2, u_3, u_4, and u_n. The different sampling points are represented by x which is the locations of different points and x_I represents the nodal points distributed in the domain of the problem Ω and the number of these sampling points will dictate the smoothness of the curve plotted for the weight and shape functions.

In meshfree methods approximation of the field variable function *u(x)* without any connectivity information between the nodes is done by considering the approximation as the product of a vector of polynomial basis function and a set of coefficients varying with *x*. This can be also put as the approximation of the function *u(x)* is obtained by assuming the approximate solution as a polynomial function represented as:

$$u(x)_{appx} = a_0(x) + a_1(x)x + a_2(x)x^2 \tag{2}$$

Linear basis in 1D

$$u(x)_{appx} = a_0(x) + a_1(x)x + a_2(x)x^2 \tag{3}$$

Quadratic Basis in 1D

Considering the linear polynomial function in one-dimension, Equation (2), can be written in matrix form:

$$u(x)_{appx} = \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} a_0(x) \\ a_1(x) \end{bmatrix} \tag{3}$$

$$u(x)_{appx} = \sum_{j=0}^{m} p_j(x) a_j(x) = p^T(x) a(x) \tag{4}$$

where,

$$p^T(x) = \begin{bmatrix} 1 & x \end{bmatrix} \tag{5}$$

and

$$a^T(x) = \begin{bmatrix} a_0(x) & a1(x) \end{bmatrix} \tag{6}$$

The value of approximate solution or approximation $u(x)_{appx}$ can be evaluated by determining the unknown coefficients of x. The unknown coefficients are evaluated by minimizing the difference between the local approximation at that point or the considered node in the *support domain* and the nodal parameter for the node I, i.e.

$$u_I = u(x_I) \tag{7}$$

The approximated value of field variable at the local nodes is given by:

$$u(x, x_I)_{appx} = p^T(x_I) a(x) \tag{8}$$

Again it is emphasized that $a(x)$ is arbitrary function of the sampling point's x and x_I represents the nodal points. The minimization process starts with the construction of a weight residual functional with respect to unknown coefficients, considering the Equations (4) and (8), and given by:

$$J = \sum_{I=1}^{n} w(x - x_I)(u(x, x_I)_{appx} - u_I)^2 \tag{9}$$

The minimization of functional J produces a set of linear Equations [1, 4]:

$$\mathbf{A}(x)\mathbf{a}(x) = \mathbf{B}(x)\mathbf{u} \tag{10}$$

where,

$$a(x) = A^{(-1)}(x) B(x) u \tag{11}$$

where $\mathbf{A}(x)$ is known as weight moment matrix and given by:

$$\mathbf{A}(x) = \sum_{I=1}^{n} w(x - x_I)\, p(x_I)\, p^T(x_I) \tag{12}$$

$$\mathbf{A}(x) = w(x - x_1) \begin{bmatrix} 1 \\ x_1 \end{bmatrix} \begin{bmatrix} 1 & x_1 \end{bmatrix} + w(x - x_2) \begin{bmatrix} 1 \\ x_2 \end{bmatrix} \begin{bmatrix} 1 & x_2 \end{bmatrix} + w(x - x_n) \begin{bmatrix} 1 \\ x_n \end{bmatrix} \begin{bmatrix} 1 & x_n \end{bmatrix} \tag{13}$$

$$\mathbf{B}(x) = \begin{bmatrix} B(x_1) & B(x_2) \ldots B(x_n) \end{bmatrix} \tag{14}$$

and,

$$B(x_I) = w(x - x_I) \begin{bmatrix} 1 \\ x_I \end{bmatrix} \tag{15}$$

The nodal parameters of the field variables are represented by the vector $\mathbf{u}$:

$$\mathbf{u} = \begin{bmatrix} u_1 & u_2 \ldots u_n \end{bmatrix}^{\mathrm{T}} \tag{16}$$

The new variable $w(x - x_I)$, introduced in the Equation (10) is known as the weight function. The weight function, Belytscho and Dolbow (1998), considered in this study, is cubic spline and given by:

$$w(x - x_1) = \begin{cases} \frac{2}{3} - 4r^2 + 4r^3 & if \quad r \le 1/2 \\ \frac{4}{3} - 4r + 4r^2 - 4r^3 & if \quad 0.5 < r \le 1 \\ 0 & if \quad r > 1 \end{cases} \tag{17}$$

Another weight function known as quartic spline, is also used to present the effect of weight function on the shape function, is given as:

$$w(x - x_1) = \begin{cases} 1 - 6r^2 + 8r^3 - 3r^4 & if \quad r \le 1 \\ 0 & if \quad r > 1 \end{cases} \tag{18}$$

where $r = |x - x_I|/d_I$ and, dI is the radius of

influence domain or radius of support domain of the node.

Substitution of Equation (12) into the approximate solution Equation (5), leads to:

$$u(x)_{appx} = \sum_{(j=0)}^{m} p_j(x) \sum_{(I=1)}^{n} A^{(-1)}(x) B(x) u_I \qquad (19)$$

$$u(x)_{appx} = \sum_{I=1}^{n} \phi(x_1) u_1 = \mathbf{\Phi}^{\mathbf{T}}(x)\mathbf{u} \qquad (20)$$

where,

$$\mathbf{\Phi}(x) = [\phi_1(x) \quad \phi_2(x) \quad \ldots \quad \phi_n(x)] \qquad (21)$$

The MLS shape function for i^{th} node is defined by:

$$\phi_I(x) = p^T(x)(A^{(-1)}(x)B(x))_I \qquad (22)$$

The MLS shape function for the middle node i.e. I = 3 is expanded to clear the programming aspects and distinguishing between the nodes and sampling points. The domain Ω = (0, 1) is represented by five nodes located at points (0, 0.25, 0.5, 0.75 and 1.0). $\phi_3(x)$, indicates the shape function associated with the *support node three*, it is a vector having the values corresponding to the number of *support nodes* in the support or influence domain of middle node located at x = 0.5 and is given by:

$$\phi_3(x) = [1 \quad x_3]\mathbf{A}^{-1}(x)\, w(x - x_3)\begin{bmatrix} 1 \\ x_3 \end{bmatrix} \qquad (23)$$

The influence domain is considered to be equal to d_I = 0.4375, this is the radius of circle; in the area of this circle all the nodes are influencing or contributing to the approximation. The number of nodes in the influence domain of the middle node is three and their location are at *x = 0.25, x = 0.5* and *x = 0.75*. The shape function associated with this middle or star node at *x = 0.5*, is elaborated to bring out the clarity. The evaluation of weight moment matrix-A; associated with middle node:

$$\mathbf{A}_3(0.25) = w(0.25 - 0.5)\begin{bmatrix} 1 \\ 0.5 \end{bmatrix}[1 \quad 0.5] \qquad (24)$$

$$\mathbf{A}_3(0.5) = w(0.5 - 0.5)\begin{bmatrix} 1 \\ 0.5 \end{bmatrix}[1 \quad 0.5] \qquad (25)$$

$$\mathbf{A}_3(0.75) = w(0.75 - 0.5)\begin{bmatrix} 1 \\ 0.5 \end{bmatrix}[1 \quad 0.5] \qquad (26)$$

The calculations for the shape function are:

$$\phi_3(0.25) = [1 \quad 0.5]\mathbf{A}^{-1}(0.25)\, w(0.25 - 0.5)\begin{bmatrix} 1 \\ 0.5 \end{bmatrix} \qquad (27)$$

$$\phi_3(0.5) = [1 \quad 0.5]\mathbf{A}^{-1}(0.5)\, w(0.5 - 0.5)\begin{bmatrix} 1 \\ 0.5 \end{bmatrix} \qquad (28)$$

$$\phi_3(0.75) = [1 \quad 0.5]\mathbf{A}^{-1}(0.75)\, w(0.75 - 0.5)\begin{bmatrix} 1 \\ 0.5 \end{bmatrix} \qquad (29)$$

After performing the calculation we receive the value of shape function corresponding to $\phi_{3(0.25)}$ = 0.1197, $\phi_{3(0.50)}$ = 0.7605 and $\phi_{3(0.75)}$ = 0.1197. This can be better assimilated by Figure 1, representing the shape function, using the linear basis function, cubic spline weight function and d_I = 0.4375 with two different number of sampling points (SP). The shape functions associated with six nodes are represented in Figure 2 using the arbitrary nodal distribution.

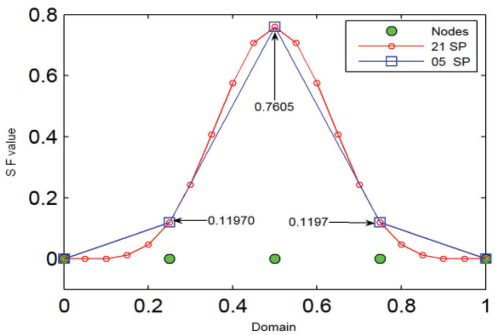


Figure 1. Shape function for middle node with uniform nodal distribution

Shape Function Derivative

The derivatives of shape function can be calculated by applying the product rule to Equation (23):

$$\phi_I(x) = p^T(x)(A^{(-1)}(x)B(x))_I \tag{30}$$

The first derivative is obtained as

$$\phi_{I,x} = (p^T(x)(A^{(-1)}(x)B(x))_I),x \tag{31}$$

$$\phi_{I,x} = p^T{}_{,x}A^{(-1)}(x)B(x)_I + p^T A^{(-1)}{}_{,x}B(x)_I + p^T A^{(-1)}B(x)_{I,x} \tag{32}$$

The further expansion for the equation is similar to as described for the shape function.

Nan Error

The critical and erroneous situation occurs whenever the derivative of shape function is to be evaluated at the node which is also a sampling point or point of interest. The derivative of weighting function and shape function cease to exist, this condition if occurs during the meshfree program execution the results will not be logical and accurate. Figure 3 presents the plot of shape function derivatives, in this case the *NaN* error has been solved by shifting each node by a distance of 0.0001 from the locations of uniform distribution, within the boundary of domain. The next combined Figure 4 presents the plots of weight function, shape function and their derivatives with (above) and without (below) implementing the node shifting. The derivatives of the weighting function are discontinuous and the shape function derivatives cease to exist. The attached matlab program provides two underlined entries the % symbol may be placed alternately on each line and the plots can be visualized.

Matlab Program Implementation

The programming flow chart for the moving least square shape function and their derivatives, given by Equation (22 and 32) is presented in Figure 5. The Matlab program for getting the shape function plots is provided at appendix. The program steps are elaborated, here under:

1. Enter the domain of the problem and represent the geometry into nodal and sampling points.
2. Initialize the support domain of influence, matrices for the weight function and shape function
3. Initialize the first "for" loop for number of nodes as the shape function needs to be calculated at each node, within this "for" loop other loops are initialized to:

3.1 Find the values of weight function and derivatives at the nodal points,

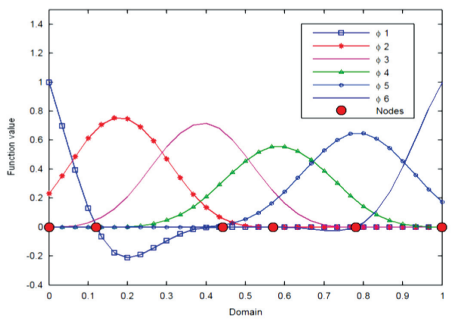


Figure 2. Shape function for six nodes with arbitrary nodal distribution

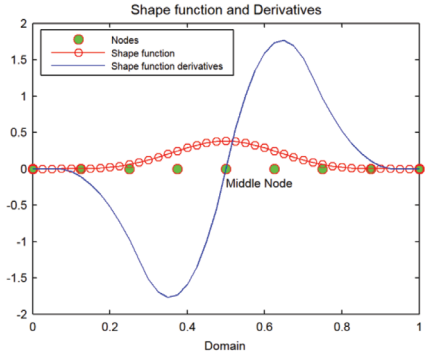


Figure 3. Shape function and derivative for middle node

3.2 Find the values of weight function and derivatives at the sampling points

3.3 Find A-matrices and their inverse at nodes

3.4 Find B-matrices and derivatives at Sampling points

3.5 Find the value of shape function and derivatives at the nodal and sampling points

3.6 End of first "for" loop

4. Plot the nodal points

5. Plot the weight function and derivatives

6. Plot the shape function and derivatives using nodal points and sam-pling points

7. Add legend, x and y label to the plots.

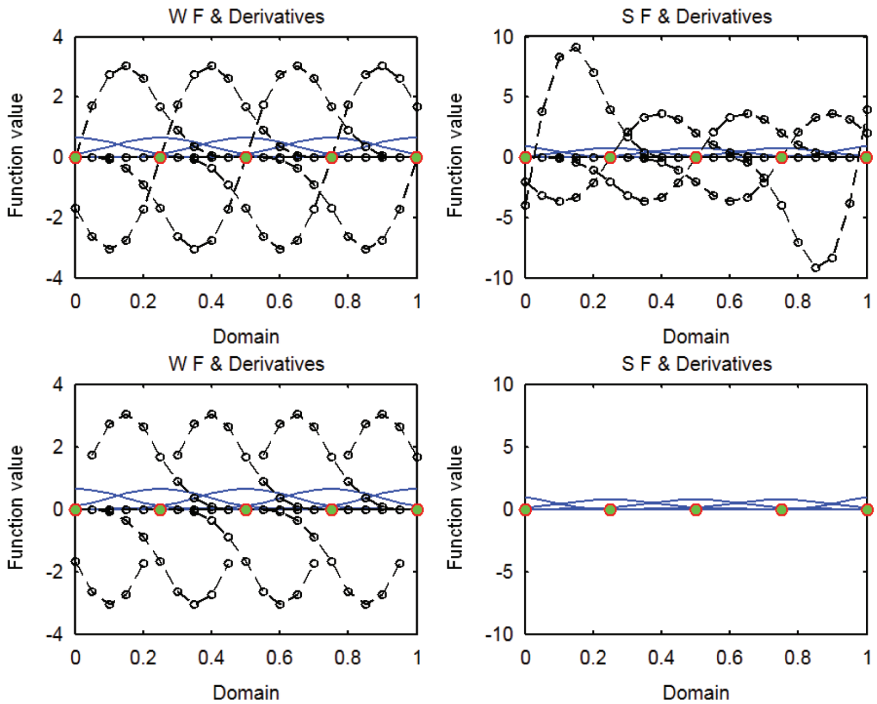


Figure 4. Effect of node shifting for uniform nodal distribution

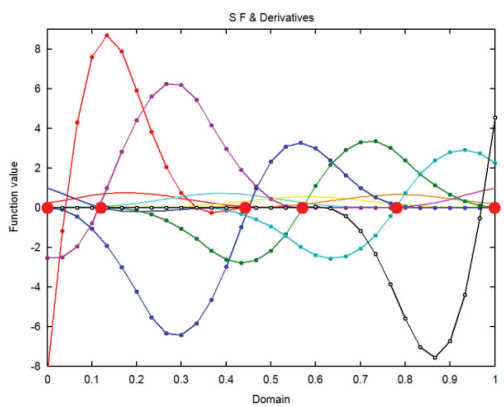


Figure 5. Shape function and derivatives with arbitrary nodal distribution

Observations

The various findings related to the properties and effects of various factors on the behavior of the shape function are included along with the plots obtained using the matlab program.

Partition of Unity

The Figure 7 presents the shape function for uniformly distributed 9 nodes and the values of the shape function for the middle node, corresponding to other node locations, marked by circles; are presented in the Table 1, to show that the shape function approximates and satisfies partition of unity conditions subject to use of constant terms. Similarly Table 2 tabulates the shape function values con-sidering the arbitrary distribution of the nodes and it is confirmed that the nodal arbitrary distribution also satisfies the partition of unity.

Lack of Kronecker Delta

The Table 1 presents the values of shape function associated with node number five $x = 0.5$, the value of shape function at this node is $\phi_{(5)}(x) = 0.3807 \neq 1.0$. Thus the moving least square shape functions do not satisfy the Kronecker delta condition.

Continuity

The shape functions have high order of continuity, although only linear basis has been used, because of the fact that shape functions inherit the continuity of the weighting function. The weighting function used is cubic spline. Figure 9 indicates the effect of the weighting functions, as the two weighting functions have different shapes and order of

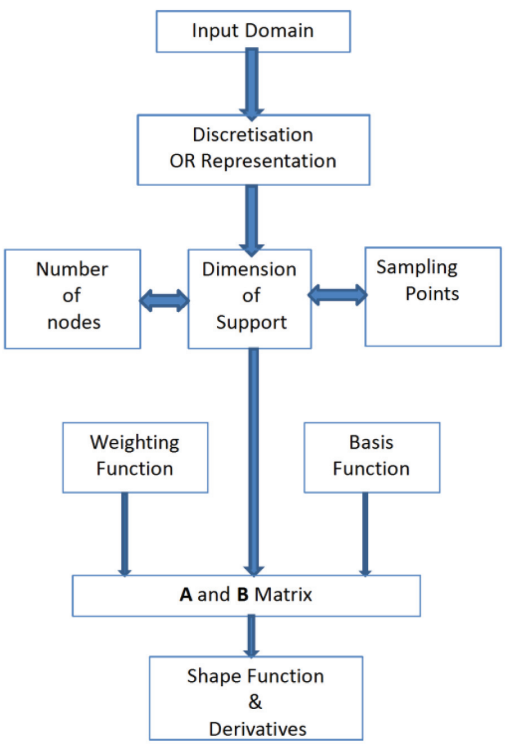


Figure 6. Flow chart for shape function calculation

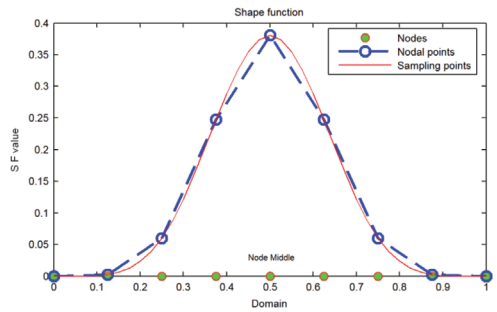


Figure 7. Shape function to confirm partition of unity: uniform nodal distribution

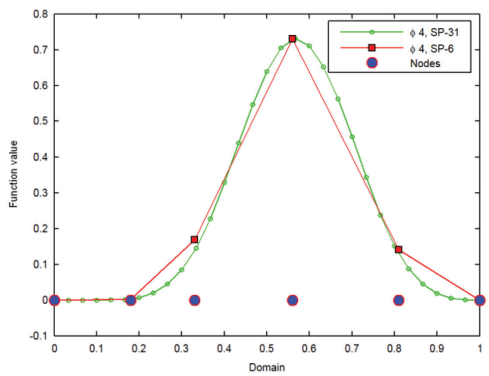


Figure 8. Shape function for arbitrary nodal distribution

continuities the shape function inherits these features of the weighting functions.

Weighting Function

The selection of the weighting function plays a very vital role in the for-mulation and solution of meshfree methods. The shape functions generated with cubic spline and quartic spline are represented in Figure 9. The conclusion from this can be made that the cubic spline weighting function gives the shape function which posses more local character.

Bell Shape

The shape functions possess the *bell* shape, presented in Figures 1-6, as the number of nodes in the support domain is increased the height of the bell gets lo-wered and spreads gets lengthened increasing the global influence.

Table 1. Partition of unity: Uniform nodal distribution

Node number	$\phi_{(5)}$
1	0
2	0.0022
3	0.0599
4	0.2475
5	0.3807
6	0.2475
7	0.0599
8	0.0022
9	0
Total	**0.9999**

Reduction of Peaks

The peak values of the shape functions fall down as the number of the nodes in the support domain is increased as a result the smoothness increases and the local character starts decreasing and the behavior tends to be global. Comparing Figures 1 and 7, give a better visualization, note the values on the y-axis.

Sampling Points

The difference between the nodal points and sampling points is very vital and made very clear and unambiguous by Figure 10, representing the plots for the first and the second nodes, as the number of sampling points is increased the smoothness of the curve is increased without affecting the peak values of the shape function corresponding to the nodes.

Mirror Image

The shape functions are the mirror image of each other from the central node, for the uniform nodal distribution. These approach or/and reach unit values near the boundaries of the domain so that the imposition of the boundary condition is sim-plified. The approach to unit value is reasonably gradual and emphasized

Table 2. Partition of unity: Arbitrary nodal distribution

Node no.	$\phi_{(1)}$	$\phi_{(2)}$	$\phi_{(3)}$	$\phi_{(4)}$	$\phi_{(5)}$	$\phi_{(6)}$
1	0.983	0.221	0.008	0	0	0
2	0.035	0.515	0.240	0.002	0	0
3	0.019	0.260	0.582	0.137	0	0
4	0	0.002	0.168	0.730	0.140	0
5	0	0	0	0.130	0.675	0
6	0	0	0	0	0.184	1
Total	**1**	**1**	**1**	**0.999**	**1**	**1**

by drawing a horizontal line in Figure 11(a) and 11(b).

Influence of Support Domain

The effect of influence of support domain is represented by Figure 12, it can be concluded that with the increase of support domain the local behavior of the shape function diminishes and as the nodes in the support domain are decreased the shape function value approaches to unity. In this condition the shape function will interpolate through the nodal values, if the A-matrix is invertible, however if the number of node in the support domain becomes less than the num-ber of monomials in the basis function, inverse of A-matrix will not exist.

Basis Function

The effect of the basis function on the shape function is presented by Figure 13. Moving least square shape functions using linear, quadratic and cubic basis function and cubic spline weighting function are computed and plotted to vi-sualize the effect of basis function on the shape functions. The study concludes that as the order of basis function is increased, the value of $\phi_{(5)}(x)$ increases to maximum and becomes constant as the order of basis and weighting function be-come equal.

Verification

The program for the shape function is validated by solving the problem from the

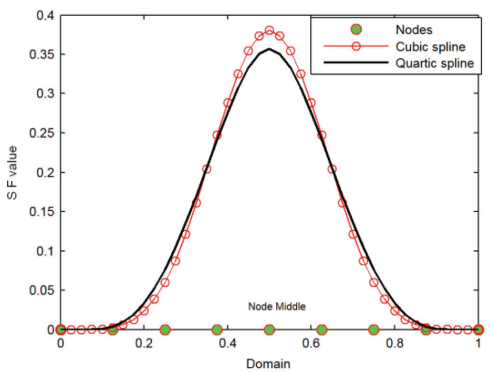


Figure 9. Shape function with cubic and quarti spline using linear basis

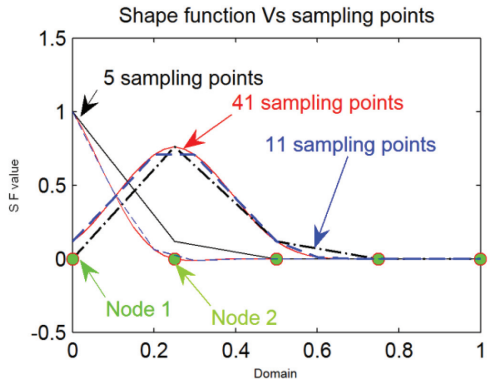
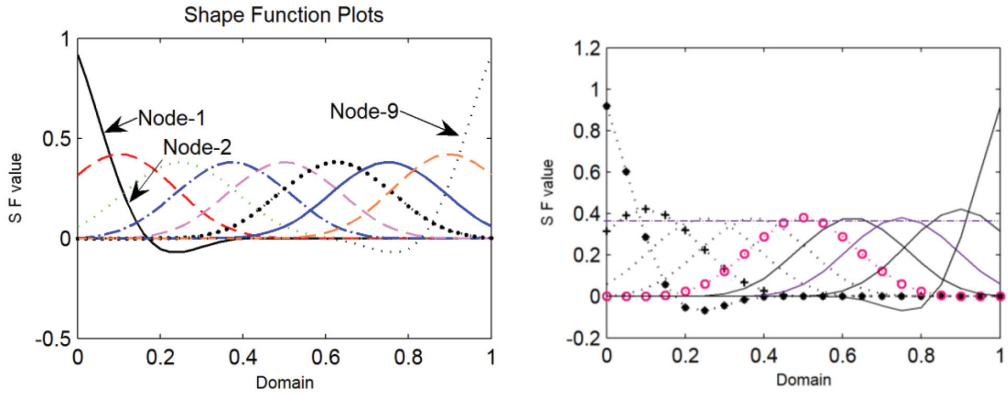


Figure 10. Shape function for the first and second node



(a) Sampling points-51 **(b) Sampling points-21**

Figure 11. Shape function for 9 nodes with Linear basis, Cubic spline, (d_I = 0.4375)

elastostatics using uniform nodal distribution. The problem is considered from Hutton (2004) and Kushawaha (2012). A tapered bar of un-iformly varying cross sectional area, $A_1 = 1\ m^2$, $A_2 = 0.5\ m^2$ on each end, subjected to tensile point load, P = 1000 kN, and young's modulus, E = 200 GPa; was mod-eled and solved usingthe element free galerkin method and the results are positive and in good agreement to the exact solution, validating that the developed of for-mulation and matlab program. The Figure 14 represents the comparative plots ob-tained by exact solution and the element free galerkin (EFG) method with the variation of support domain of influence. The increase in the number of nodes brings the convergence of the meshfree solution to the exact solution.

Conclusions

The moving least square shape function and its derivative is studied after generating a Matlab program and the various characteristics of the shape functions are elabo-rated in context to the parameters affecting the features of shape function. The significance of *NaN* in contribution to meshfree solution errors is highlighted and prevention methodology has been implemented to minimize the error generation. The plots are presented to support the discussion. The program is extensively explained and provided as appendix to help the beginners on the meshfree metho-dology.

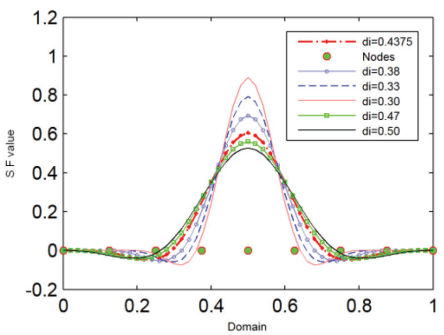


Figure 12. Influence of support domain, d_I

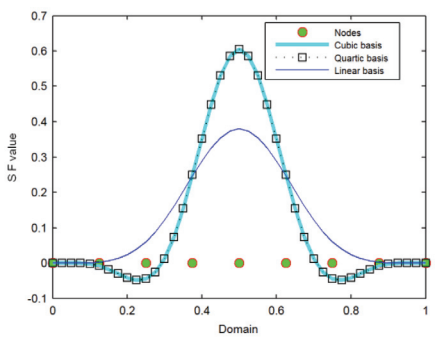


Figure 13. Shape function with different basis functions

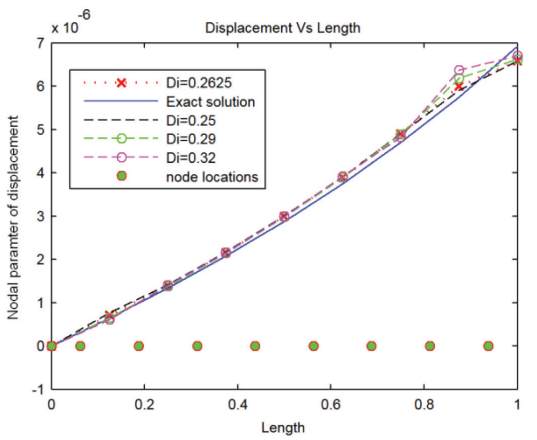


Figure 14. Comparison of EFG and exact solution

Appendix

This program is written to explain the implementation and plotting of the moving least square shape function used in the meshfree methods. The program will display the plots of shape function & its derivatives for the arbitrarily distributed Seven Nodes with influence domain= 0.4375. No effort has been made to optimize the program to keep it easy to grasp.

```
% SETTING UP DOMAIN AND NODAL POINTS OR LOCATION OF NODES
xnode=[0 0.12 0.42 0.57 0.78 0.87 1];% ARBITRARILY DISTRIBUTED SEVEN NODES
ynode=zeros(1,length(xnode)); % TO PLOT NODES
% SETTING UP SAMPLING POINTS
X=linspace(0,1,31);
% SETTING UP INFLUENCE DOMAIN
di=0.4375;
% TO PREVENT THE NaN ERROR DURING THE PROGRAMM EXECUTION THE NODES ARE
% SHIFTED BY NEGLIGIBLE DISTANCE WITHIN THE DOMAIN, TRY RUNNING THE
% PROGRAM WITH THE FOLLOWING VALUES EQUAL TO THAT OF DOMAIN = xnode TO VISUALISE
% THE NaN EFFECT AND YOU SHOULD GET THE PLOT SHOWN IN FIGURE-16, WITH MISSING
% CURVES FOR DERIVATIVES
Shift_xnode(1)=0.0001;
Shift_xnode(2)=0.1201;
Shift_xnode(3)=0.4201;
Shift_xnode(4)=0.5699;
Shift_xnode(5)=0.7801;
Shift_xnode(6)=0.8701;
Shift_xnode(7)=0.9999;
% INITIALISE MATRICES FOR WEIGHT FUNCTION & DERIVATIVES AT NODAL POINTS
w=zeros(length(xnode),length(xnode));
dw=zeros(length(xnode),length(xnode));
% INITIALISE MATRICES FOR WEIGHT FUNCTION & DERIVATIVES AT SAMPLING POINTS
W=zeros(length(xnode),length(X));
dW=zeros(length(xnode),length(X));
% INITIALISE MATRICES FOR SHAPE FUNCTION & DERIVATIVES AT SAMPLING POINTS
Phi=zeros(length(xnode),length(X));
dPhi=zeros(length(xnode),length(X));
% SET UP 'for' LOOPS FOR CALCULATING SHAPE FUNCTION & DERIVATIVES
for i =1:length(xnode)
% CALCULATE WEIGHT FUNCTION & DERIVATIVES FOR NODAL POINT
    for j = 1:length(xnode)
        r = (abs(Shift_xnode(i)-xnode(j)))/di;
    if r< = 0.5
          w(i,j) = (2/3) - 4*r*r + 4*r^3;
          dw(i,j)= (-8*r + 12*r^2)*(Shift_xnode(i)-xnode(j))/(di*(abs(Shift_xnode(i)-xnode(j))));
          elseif (r>0.5)&(r<=1.0)
          w(i,j)= (4/3)-4*r+4*r*r -(4/3)*r^3;
          dw(i,j)= (-4 + 8*r-4*r^2)*(Shift_xnode(i)-xnode(j))/(di*(abs(Shift_xnode(i)-xnode(j))));
          elseif r>1.0
          w(i,j) = 0.0;
          dw(i,j) = 0.0;
        end
    end
    w;  dw;
% CALCULATE WEIGHT FUNCTION & DERIVATIVES AT SAMPLING POINTS TO GET SMOOTH PLOTS
    for l = 1:length(X)
        r = (abs(xnode(i)-X(l)))/di;
        if r<=0.5
          W(i,l) = (2/3) - 4*r*r + 4*r^3;
            dW(i,l)= (-8*r + 12*r^2)*(Shift_xnode(i)-X(l))/(di*(abs(Shift_xnode(i)-X(l))));
        elseif (r>0.5)&(r<=1.0)
          W(i,l)= (4/3)-4*r+4*r*r -(4/3)*r^3;
           dW(i,l)= (-4 + 8*r-4*r^2)*(Shift_xnode(i)-X(l))/(di*(abs(Shift_xnode(i)-X(l))));
        elseif r>1.0
          W(i,l) = 0.0;
          dW(i,l) = 0.0;
       end
    end
     W;  dW;
% NOW WE START TO DERIVE THE SHAPE FUNCTIONS AND DERIVATIVES USING
% EQUATION 1.23 AND 1.33
   won=ones(1, length(xnode)); % Initialisation of ones vector
   Won=ones(1, length(X)); % Initialisation of ones vector
     p=[won; xnode];
  P=[Won; X];
  A=zeros(2, 2);          % Initialization of A-Matrix
  dA=zeros(2, 2);         % Initialization of dA-Matrix
       for j=1:length(xnode)% CALCULATE A-MATRIX AT NODAL POINT
         A= A + (w(i,j)*p(1:2,j)*p(1:2,j)');
        dA= dA + (dw(i,j)*p(1:2,j)*p(1:2,j)');
    end
     A ;   dA;
     invA=inv(A);
```

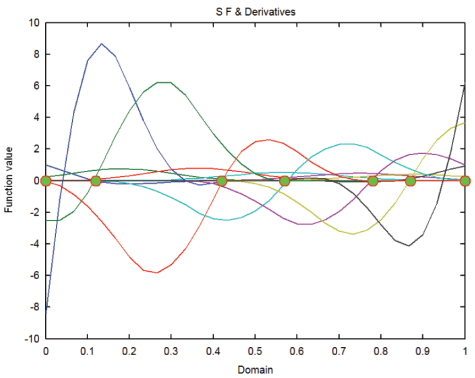


Figure I. Shape function and derivatives plot by above Matlab program

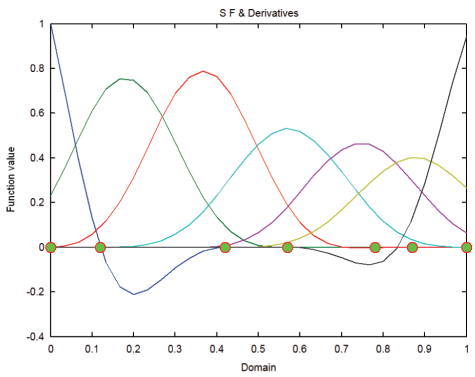


Figure II. Shape function and derivatives plot with NaN error

```
    invdA=-invA*dA*invA;
      for j=1:length(X)% CALCULATE THE B-
MATRIX AT SAMPLING POINTS
      B(1:2,j)=W(i,j).*P(1:2,j);
      dB(1:2,j)=dW(i,j)*P(1:2,j);
    end
    B; dB;
      phi=[1 xnode(i)]*invA*B;% CALCULATE
SHAPE FUNCTION
     dphi=([0 1]*invA*B) + [1 xnode(i)]*(invdA*B +
invA*dB);
    Phi(i,:)=phi;
    dPhi(i,:)=dphi;
end
plot(X,Phi,'LineWidth',0.5)
hold on
plot(X,dPhi,'LineWidth',0.5,'MarkerSize',1.5);
hold on
title('S F & Derivatives','FontSize',5)
xlabel('Domain','FontSize',5)
ylabel('Function value','FontSize',5)
plot(xnode,ynode,'ro','MarkerEdgeColor','r','Marker
FaceColor','g','MarkerSize',5)
hold on
%  We are getting the plot shown below (figure-I):
```

References

Atluri, S.N., Kim, H.G., and Cho, J.Y. (1999). A critical assessment of the truly Meshless Local Petrov-Galerkin (MLPG), and Local Boundary Integral Equation (LBIE) methods. Comput. Mech., 24:348-372.

Belytschko, T. and Dolbow, J. (1998). An introduction to programming the meshless element free galerkin method. Archives of Comput. Methods Eng., 5:207-241.

Belytschko, T., Lu, Y.Y., and Gu, L. (1994). Element-free galerkin methods. Int. J. Numer. Methods Eng., 37:229-256.

Hutton, D.V. (2004). Fundamentals of finite element analysis. Tata Mc-Graw Hill.

Kushawaha, J.S. (2012). Meshfree shape function from moving least square. E J. Sci. Technol., 1(7):29-41.

Liu, G.R. (2003). Mesh Free Methods: Moving beyond the finite element methods. CRC Press, Boca Raton, USA.

Liu, W.K., Jun, S., and Zhang, Y. (1995). Reproducing kernel particle methods, Int. J. Numer. Methods Fluids, 20:1081-1106.