Introduction to Moving Least Squares(MLS) Shape Functions
by Louie L. Yaw, PhD, PE, SE
Walla Walla University
April 19, 2009

1 Some relevant terminology.

Some background terminology is helpful to understand the following sections. The terminology here is consistent with terminology often used in the literature.

m – The order of the polynomial base.

Polynomial base – A column vector, $\mathbf{p}$, of a *complete* polynomial of order m. For example, for a 1D linear (order m=1) polynomial base $\mathbf{p}^T = [1\ x]$, for a 1D quadratic (order m=2) polynomial base $\mathbf{p}^T = [1\ x\ x^2]$, and similarly for higher order polynomials. In each case these examples give a complete polynomial base since no terms of the polynomial, from order 0 to m, are missing. For a 2D polynomial (order m=1) linear base $\mathbf{p}^T = [1\ x\ y]$, for quadratic (order m=2) $\mathbf{p}^T = [1\ x\ y\ x^2\ xy\ y^2]$.

k – The number of terms in the polynomial base. When programming the moving least squares shape functions it is useful to know how many terms are in the mth order polynomial base vector. The following table gives the relationship between m and k for 1, 2 or 3 dimensions.

Dimensions	k
1D	$m+1$
2D	$(m+1)(m+2)/2$
3D	$(m+1)(m+2)(m+3)/6$

Table 1: Relationship between m and k based on the number of dimensions.

ϕ_a – A shape (or basis) function associated with a particular point a in the domain of the problem.

$\boldsymbol{x}$ – A point at which a shape function is evaluated. It is just a number in 1D, and a vector in 2D or 3D.

n – The number of points that have nonzero weight function values at evaluation point $\boldsymbol{x}$. These points are a subset of the points used to discretize the entire domain. The points used to discretize the entire domain are the alternative to covering the domain with finite elements.

$\boldsymbol{x}_a$ – A particular point a has its own vector of coordinates $\boldsymbol{x}_a$ each of which are associated with shape function ϕ_a. The collection of points $\boldsymbol{x}_a$, there are n of them, are used to construct the MLS shape functions.

Kronecker-delta property – If the shape functions have the Kronecker-delta property then $\phi_a(x_b) = \delta_{ab}$. In FEM the shape functions have the Kronecker-delta property. However, MLS shape functions generally do not have the Kronecker-delta property. This is the reason that imposition of essential boundary conditions has been difficult in meshfree methods.

Compact Support – In FEM the shape functions are said to have compact support. For instance they have the Kronecker-delta property. So their *support* is limited to a small region

around a particular node. MLS shape functions can have compact support also. Generally, even though the Kronecker-delta property is absent for MLS shape functions, the *support* of each MLS shape function is kept compact. This is done by the use of a support radius, ρ_a, for each node a, and the weight function.

Support Radius - Each shape function has a support or zone of influence. Usually this is a circular region in 2D or a sphere in 3D centered on the point associated with the shape function. The radius of this zone of influence is called the support radius, ρ. The radius is usually taken to be the distance to the 2nd nearest neighboring point in 1D and the 3rd or 4th nearest neighboring point in 2D, times a user input parameter, α.

Zone of influence – The domain within which a given shape function is nonzero.

Weight Function – Generally, weight functions are chosen to be the same for all nodes. However, this is not required. A variety of weight functions are available (see Belytschko et al or Fries and Matthies). Each weight function a is constructed so that it equals 1 at the point a and is zero at a distance equal to the support radius away from the point. The 4th order quartic spline weight function is effective and easy to use. With $q = \|\boldsymbol{x} - \boldsymbol{x}_a\|/\rho_a$ the 4th order quartic spline is defined as follows:

$$w(q) \in C^2 = \begin{cases} 1 - 6q^2 + 8q^3 - 3q^4 & q \leq 1 \\ 0 & q > 1 \end{cases} \tag{1}$$

Bubnov-Galerkin Method– The common method of using the same shape functions for both the trial functions and the test functions in the weak form of a BVP.

Element Free Galerkin – The meshfree method which makes use of MLS shape functions and solves the system of governing pde's by a Galerkin method (usually the Bubnov-Galerkin Method).

Weighted Least Squares – Each squared term in a traditional least squares summation is multiplied by its own weight function. Hence the influence of each squared term is affected or limited by its associated weight. For instance, a weighted least squares functional (which is explained in more detail later in this appendix) differs from a traditional least squares functional only by the addition of the w_a term in the summation as follows:

$$J(\mathbf{g}) = \frac{1}{2} \sum_{a=1}^{n} w_a \left\{ \mathbf{p}^T(\boldsymbol{x}_a)\mathbf{g} - u_a \right\}^2 \tag{2}$$

Moving Least Squares (MLS) – For a particular fixed point $\boldsymbol{x}$ each w_a is calculated for the weighted least squares functional. The functional J is then minimized with respect to each g_i (i ranges from 1 to k) and solved for the g_i values (contained in $\mathbf{g}$). These g_i values are then dependent on the fixed point used to evaluate the w_a's. Later, it is shown how this minimization procedure is used to determine the value of the MLS shape functions at a particular fixed point $\boldsymbol{x}$. Hence every time there is a move to a new point the minimization procedure is repeated. This is why this is a moving least squares procedure, because it is dependent on the current evaluation point $\boldsymbol{x}$.

Moment Matrix – In the literature this is often denoted as the $\mathbf{A}$ matrix. This matrix is defined during the derivation of the MLS shape functions given below.

Interpolation property – In FEM the approximation is of the form $u^h = \Sigma\phi_a(\boldsymbol{x}_b)u_a$, so that when b is equal to a, $u^h = u_b$ due to the Kronecker-delta property. This is the

interpolating property of FEM shape functions. However, MLS shape functions generally do not have the Kronecker-delta property and hence are not interpolating. Hence more than one shape function may be nonzero at each node. Another way of saying this is that the values u_a are not nodal values like they are in FEM, hence they are sometimes called nodal coefficients instead.

Partition of unity – If the MLS shape functions are constructed correctly then the shape functions evaluated at any given point $\boldsymbol{x}$ sum to 1. This is a good way to numerically check if the shape functions are constructed correctly. If shape functions are constructed correctly this criteria is satisfied to machine precision. The partition of unity property is stated mathematically as follows:

$$\sum_{a=1}^{n} \phi_a(\boldsymbol{x}) = 1 \tag{3}$$

Partition of nullity – If the derivative of the MLS shape functions are constructed correctly then the shape function derivatives evaluated at any given point $\boldsymbol{x}$ sum to 0. This is a good way to numerically check if the derivatives are constructed correctly. If the derivatives are constructed correctly this criteria is satisfied very close to machine precision:

$$\sum_{a=1}^{n} \nabla\phi_a(\boldsymbol{x}) = \mathbf{0} \tag{4}$$

Consistency – The highest polynomial order which is represented exactly. For example, MLS shape functions constructed with a linear polynomial base have 1st order consistency. That is, if a differential equation has a linear solution then the MLS shape functions can represent the solution exactly. Another example, consider an mth order polynomial base $\mathbf{p}(\boldsymbol{x})$ and n points at which MLS shape functions are constructed. So to each point $\boldsymbol{x}_a$ (a vector in 2 and 3 dimensions) there is a corresponding shape function ϕ_a. Then, with mth order consistency, the following is true:

$$\sum_{i=a}^{n} \phi_a(\boldsymbol{x})\mathbf{p}(\boldsymbol{x}_a) = \mathbf{p}(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \Omega \tag{5}$$

2 Moving least squares (MLS) shape functions derivation

The derivation given is for 1D problems (hence the x variable is non bold in the following, the derivation is easily extended to 2 or 3 dimensions merely by making x bold as necessary to reflect its vector character and by appropriate modification of the polynomial base).

Consider the task of finding an approximate solution $u^h(x)$, while knowing the true solution, u_a at selected points x_a. Then in a least squares sense minimization of the expression $[u^h(x_a) - u_a]^2$ for each a is the objective. Suppose a polynomial approximation is chosen so that

$$u^h(x) = g_1 + g_2 x + g_3 x^2 + \ldots g_{m+1} x^m. \tag{6}$$

The approximation is then written in matrix form

$$u^h(x) = \mathbf{p}^T(x)\mathbf{g} = \{1 \; x \; x^2 \; \ldots x^m\} \begin{Bmatrix} g_1 \\ g_2 \\ g_3 \\ \vdots \\ g_{m+1} \end{Bmatrix}. \tag{7}$$

Using the above the least squares functional is written with the approximation substituted in for $u^h(x)$ (the 1/2 in front is added for mathematical convenience):

$$J = \frac{1}{2}\sum_{a=1}^{n} \left\{u^h(x_a) - u_a\right\}^2 = \frac{1}{2}\sum_{a=1}^{n} \left\{\mathbf{p}^T(x_a)\mathbf{g} - u_a\right\}^2. \tag{8}$$

Now, recall that compact support for each node a is intended. Therefore, the local solution is influenced by the local nodes. Whereas, nodes far away have no influence. Hence, each summation term, indexed by a, in the least squares functional is weighted by a weight function w_a, which limits the term's influence to point a and usually several surrounding nodes. Based on this intuition the functional J is modified and becomes a weighted least squares functional as follows:

$$J = \frac{1}{2}\sum_{a=1}^{n} w_a \left\{\mathbf{p}^T(x_a)\mathbf{g} - u_a\right\}^2. \tag{9}$$

Next, it is necessary to minimize J with respect to each g_i. However, before this operation, it is helpful to first write the functional in matrix form. To this end, the functional J is written as follows:

$$J = \frac{1}{2}(\mathbf{Pg} - \mathbf{u})^T\mathbf{W}(\mathbf{Pg} - \mathbf{u}), \tag{10}$$

where,

$$\mathbf{P} = \begin{bmatrix} p_1(x_1) & p_2(x_1) & \ldots & p_k(x_1) \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ p_1(x_n) & p_2(x_n) & \ldots & p_k(x_n) \end{bmatrix}. \tag{11}$$

Notice that each row of the $\mathbf{P}$ matrix is just $\mathbf{p}^T(x_a)$ for each row a. And, this matrix is n by k in size.

For $\mathbf{W}$, an n by n matrix results (see Eq. (1) for the definition of diagonal terms w_a)

$$\mathbf{W} = \begin{bmatrix} w_1(x - x_1) & & & 0 \\ & w_2(x - x_2) & & \\ & & \ddots & \\ 0 & & & w_n(x - x_n) \end{bmatrix}. \tag{12}$$

The $\mathbf{g}$ vector is k by 1 and the $\mathbf{u}$ vector is n by 1.

Now, set $\frac{\partial J}{\partial \mathbf{g}} = \mathbf{0}$. This yields the following:

$$(\mathbf{Pg} - \mathbf{u})^T \mathbf{WP} = \mathbf{0}. \tag{13}$$

Transposing the whole equation yields

$$(\mathbf{WP})^T (\mathbf{Pg} - \mathbf{u}) = \mathbf{0}. \tag{14}$$

Multiplying through gives

$$\mathbf{P}^T \mathbf{WPg} - \mathbf{P}^T \mathbf{Wu} = \mathbf{0} \tag{15}$$

and finally

$$\mathbf{P}^T \mathbf{WPg} = \mathbf{P}^T \mathbf{Wu}. \tag{16}$$

Now define the moment matrix $\mathbf{A} = \mathbf{P}^T \mathbf{WP}$ and $\mathbf{B} = \mathbf{P}^T \mathbf{W}$. Note that $\mathbf{A}$ is k by k and $\mathbf{B}$ is k by n. Using these definitions Eq. (16) becomes

$$\mathbf{Ag} = \mathbf{Bu}. \tag{17}$$

Solve now for the unknown coefficients $\mathbf{g}$

$$\mathbf{g} = \mathbf{A}^{-1} \mathbf{Bu}. \tag{18}$$

Substitute this into the first part of Eq. (7)

$$u^h = \mathbf{p}^T(x)\mathbf{g} = \mathbf{p}^T(x)\mathbf{A}^{-1}\mathbf{Bu}. \tag{19}$$

The approximations u^h are usually written as

$$u^h = \boldsymbol{\phi}^T \mathbf{u} = \sum_{a=1}^{n} \phi_a u_a. \tag{20}$$

Comparison of Eq. (20) with Eq. (19) reveals that the vector of MLS shape functions is

$$\boldsymbol{\phi}^T = \mathbf{p}^T(x)\mathbf{A}^{-1}\mathbf{B}. \tag{21}$$

Notice that the $\mathbf{A}$ and $\mathbf{B}$ matrices depend on $\mathbf{W}$. The $\mathbf{W}$ matrix in turn is a function of the x_a and the evaluation point x. Hence every time a new evaluation point x is chosen the matrices $\mathbf{A}^{-1}$ and $\mathbf{B}$ are recomputed to calculate the MLS shape functions based on the equation $\boldsymbol{\phi}^T = \mathbf{p}^T(x)\mathbf{A}^{-1}\mathbf{B}$.

3 MLS shape function characteristics

The properties of MLS shape functions are different then FEM shape functions in the following ways:

- MLS shape functions do not have the Kronecker-delta property
- MLS shape functions are not known in closed form

- MLS shape functions are non-interpolating
- The size of support can be controlled by the radius of support parameter, ρ. It is common to set ρ equal to the distance to the 3rd or 4th nearest neighboring point (in 2D for example) and adjust this as necessary by a parameter called α. This parameter is a user specified input when constructing MLS shape functions. It is sometimes necessary to adjust the radius of support by use of α to eliminate a singular moment matrix $\mathbf{A}$ when insufficient nodes are included within the support radius.
- More computational effort is required to calculate the MLS shape functions.
- Because the MLS shape functions do not have the Kronecker-delta property special attention must be paid to enforcing essential boundary conditions during the solution of a boundary value problem.

4 MLS shape function derivatives

Shape function derivatives are calculated by application of the product rule on Eq. (21). For example, if the derivative of $\boldsymbol{\phi}$ is taken with respect to the kth dimension (could be x, y or z)

$$\boldsymbol{\phi}^T_{,k}(\boldsymbol{x}) = \mathbf{p}^T_{,k}\mathbf{A}^{-1}\mathbf{B} + \mathbf{p}^T\mathbf{A}^{-1}_{,k}\mathbf{B} + \mathbf{p}^T\mathbf{A}^{-1}\mathbf{B}_{,k} \tag{22}$$

with $\mathbf{A}^{-1}_{,k} = -\mathbf{A}^{-1}\mathbf{A}_{,k}\mathbf{A}^{-1}$. This last expression is found as follows. Take the derivative of Eq. (17) to get

$$\mathbf{A}_{,k}\mathbf{g} + \mathbf{A}\mathbf{g}_{,k} = \mathbf{B}_{,k}\mathbf{u}. \tag{23}$$

Solving Eq. (23) for $\mathbf{g}_{,k}$ and substituting in Eq. (18) for $\mathbf{a}$ yields

$$\mathbf{g}_{,k} = \mathbf{A^{-1}}\mathbf{B}_{,k}\mathbf{u} - \mathbf{A^{-1}}\mathbf{A}_{,k}\mathbf{A^{-1}}\mathbf{B}\mathbf{u}. \tag{24}$$

Now take the derivative of Eq. (18), which is

$$\mathbf{g}_{,k} = \mathbf{A^{-1}}\mathbf{B}_{,k}\mathbf{u} + \mathbf{A^{-1}_{,k}}\mathbf{B}\mathbf{u}. \tag{25}$$

Finally compare Eq. (24) and Eq. (25) and observe that they are the same except for the coefficient of the $\mathbf{Bu}$ term. Hence the coefficients of this term must be equivalent. Therefore $\mathbf{A}^{-1}_{,k} = -\mathbf{A}^{-1}\mathbf{A}_{,k}\mathbf{A}^{-1}$, which completes the derivation. Second derivatives are found similarly, see Fries and Matthies page 21.

4.1 Second derivatives

The second derivatives are found as follows. Take the derivative of (22) with respect to the lth dimension (x, y, or z) by repeated application of the product rule to obtain

$$\begin{aligned}\boldsymbol{\phi}^T_{,kl}(\boldsymbol{x}) =& \mathbf{p}^T_{,kl}\mathbf{A}^{-1}\mathbf{B} + \mathbf{p}^T_{,k}\mathbf{A}^{-1}_{,l}\mathbf{B} + \mathbf{p}^T_{,k}\mathbf{A}^{-1}\mathbf{B}_{,l} \\ &+\mathbf{p}^T_{,l}\mathbf{A}^{-1}_{,k}\mathbf{B} + \mathbf{p}^T\mathbf{A}^{-1}_{,kl}\mathbf{B} + \mathbf{p}^T\mathbf{A}^{-1}_{,k}\mathbf{B}_{,l} \\ &+\mathbf{p}^T_{,l}\mathbf{A}^{-1}\mathbf{B}_{,k} + \mathbf{p}^T\mathbf{A}^{-1}_{,l}\mathbf{B}_{,k} + \mathbf{p}^T\mathbf{A}^{-1}\mathbf{B}_{,kl}.\end{aligned} \tag{26}$$

In (26) everything is straightforward except, $\mathbf{A}^{-1}_{,kl}$, which is found in the following way. Previously, it is shown that

$$\mathbf{A}^{-1}_{,k} = -\mathbf{A}^{-1}\mathbf{A}_{,k}\mathbf{A}^{-1}. \tag{27}$$

Taking the derivative of (27) with respect to the lth dimension gives

$$\mathbf{A}^{-1}_{,kl} = -\mathbf{A}^{-1}_{,l}\mathbf{A}_{,k}\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{A}_{,kl}\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{A}_{,k}\mathbf{A}^{-1}_{,l}. \tag{28}$$

Last, using (27), substitute $\mathbf{A}^{-1}_{,l}$ into (28) to get

$$\mathbf{A}^{-1}_{,kl} = \mathbf{A}^{-1}\mathbf{A}_{,l}\mathbf{A}^{-1}\mathbf{A}_{,k}\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{A}_{,kl}\mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{A}_{,k}\mathbf{A}^{-1}\mathbf{A}_{,l}\mathbf{A}^{-1}. \tag{29}$$

The above results match those found in the article by Fries and Matthies.

5 References

T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl, Meshless methods: An overview and recent developments. *Computer Methods in Applied Mechanics and Engineering*, 139:3-47, 1996.

T. P. Fries and H. G. Matthies, Classification and overview of meshfree methods. Technical Report Informatikbericht-Nr. 2003-03, *Institute of Scientific Computing, Technical University Braunschweig, Braunschweig, Germany*, 2004. (This document may be found on the internet as a pdf file by Google search.)

L. L. Yaw, Class notes, Meshfree methods, ECI 289F, UC Davis, Spring 2006, lectures by Professor N. Sukumar.