An As-Short-As-Possible Introduction to the Least Squares, Weighted Least Squares and Moving Least Squares Methods for Scattered Data Approximation and Interpolation

Andrew Nealen
Discrete Geometric Modeling Group
TU Darmstadt

Abstract

In this introduction to the Least Squares (LS), Weighted Least Squares (WLS) and Moving Least Squares (MLS) methods, we briefly describe and derive the linear systems of equations for the global least squares, and the weighted, local least squares approximation of function values from scattered data. By *scattered data* we mean an arbitrary set of points in $\mathbb{R}^d$ which carry scalar quantities (i.e. a scalar field in d dimensional parameter space). In contrast to the global nature of the least-squares fit, the weighted, local approximation is computed either at discrete points, or continuously over the parameter domain, resulting in the global WLS or MLS approximation respectively.

Keywords: Data Approximation, Least Squares (LS), Weighted Least Squares (WLS), Moving Least Squares (MLS), Linear System of Equations, Polynomial Basis

1 LS Approximation

Problem Formulation. Given N points located at positions $\mathbf{x}_i$ in $\mathbb{R}^d$ where $i \in [1 \dots N]$. We wish to obtain a globally defined function $f(\mathbf{x})$ that approximates the given scalar values f_i at points $\mathbf{x}_i$ in the least-squares sense with the error functional $J_{LS} = \sum_i \|f(\mathbf{x}_i) - f_i\|^2$. Thus, we pose the following minimization problem

$$\min_{f \in \prod_m^d} \sum_i \|f(\mathbf{x}_i) - f_i\|^2, \tag{1}$$

where f is taken from $\prod_m^d$, the space of polynomials of total degree m in d spatial dimensions, and can be written as

$$f(\mathbf{x}) = \mathbf{b}(\mathbf{x})^T \mathbf{c} = \mathbf{b}(\mathbf{x}) \cdot \mathbf{c}, \tag{2}$$

where $\mathbf{b}(\mathbf{x}) = [b_1(\mathbf{x}), \dots, b_k(\mathbf{x})]^T$ is the polynomial basis vector and $\mathbf{c} = [c_1, \dots, c_k]^T$ is the vector of unknown coefficients, which we wish to minimize in (1). Here some examples for polynomial bases: (a) for $m = 2$ and $d = 2$, $\mathbf{b}(\mathbf{x}) = [1, x, y, x^2, xy, y^2]^T$, (b) for a linear fit in $\mathbb{R}^3$ ($m = 1$, $d = 3$), $\mathbf{b}(\mathbf{x}) = [1, x, y, z]^T$, and (c) for fitting a constant in arbitrary dimensions, $\mathbf{b}(\mathbf{x}) = [1]$. In general, the number k of elements in $\mathbf{b}(\mathbf{x})$ (and therefore in $\mathbf{c}$) is given by $k = \frac{(d+m)!}{m!d!}$, see [Levin 1998; Fries and Matthies 2003].

Solution. We can minimize (1) by setting the partial derivatives of the error functional J_{LS} to zero, i.e. $\nabla J_{LS} = 0$ where $\nabla = [\partial/\partial c_1, \dots, \partial/\partial c_k]^T$, which is a necessary condition for a minimum. By taking partial derivatives with respect to the unknown coefficients $c_1, \dots, c_k$, we obtain a linear system of equations (LSE) from which we can compute $\mathbf{c}$

$$\begin{aligned}
\partial J_{LS}/\partial c_1 = 0: &\quad \sum_i 2b_1(\mathbf{x}_i)[\mathbf{b}(\mathbf{x}_i)^T \mathbf{c} - f_i] = 0\\
\partial J_{LS}/\partial c_2 = 0: &\quad \sum_i 2b_2(\mathbf{x}_i)[\mathbf{b}(\mathbf{x}_i)^T \mathbf{c} - f_i] = 0\\
&\quad \vdots\\
\partial J_{LS}/\partial c_k = 0: &\quad \sum_i 2b_k(\mathbf{x}_i)[\mathbf{b}(\mathbf{x}_i)^T \mathbf{c} - f_i] = 0.
\end{aligned}$$

In matrix-vector notation, this can be written as

$$\begin{aligned}
\sum_i 2\mathbf{b}(\mathbf{x}_i)[\mathbf{b}(\mathbf{x}_i)^T \mathbf{c} - f_i] &= \\
2\sum_i [\mathbf{b}(\mathbf{x}_i)\mathbf{b}(\mathbf{x}_i)^T \mathbf{c} - \mathbf{b}(\mathbf{x}_i) f_i] &= \mathbf{0}.
\end{aligned}$$

Dividing by the constant and rearranging yields the following LSE

$$\sum_i \mathbf{b}(\mathbf{x}_i)\mathbf{b}(\mathbf{x}_i)^T \mathbf{c} = \sum_i \mathbf{b}(\mathbf{x}_i) f_i, \tag{3}$$

which is solved as

$$\mathbf{c} = [\sum_i \mathbf{b}(\mathbf{x}_i)\mathbf{b}(\mathbf{x}_i)^T]^{-1} \sum_i \mathbf{b}(\mathbf{x}_i) f_i. \tag{4}$$

If the square matrix $\mathbf{A}_{LS} = \sum_i \mathbf{b}(\mathbf{x}_i)\mathbf{b}(\mathbf{x}_i)^T$ is nonsingular (i.e. $det(\mathbf{A}_{LS}) \neq 0$), substituting Eqn. (4) into Eqn. (2) provides the fit function $f(\mathbf{x})$. For small k ($k < 5$), the matrix inversion in Eqn. (4) can be carried out explicitly, otherwise numerical methods are the preferred tool, see [Press et al. 1992] [1]. In our applications, we often use the Template Numerical Toolkit (TNT) [2].

Example. Say our data points live in $\mathbb{R}^2$ and we wish to fit a quadratic, bivariate polynomial, i.e. $d = 2$, $m = 2$ and therefore $\mathbf{b}(\mathbf{x}) = [1, x, y, x^2, xy, y^2]^T$ (see above), then the resulting LSE looks like this

$$\sum_i \begin{bmatrix}
1 & x_i & y_i & x_i^2 & x_i y_i & y_i^2\\
x_i & x_i^2 & x_i y_i & x_i^3 & x_i^2 y_i & x_i y_i^2\\
y_i & x_i y_i & y_i^2 & x_i^2 y_i & x_i y_i^2 & y_i^3\\
x_i^2 & x_i^3 & x_i^2 y_i & x_i^4 & x_i^3 y_i & x_i^2 y_i^2\\
x_i y_i & x_i^2 y_i & x_i y_i^2 & x_i^3 y_i & x_i^2 y_i^2 & x_i y_i^3\\
y_i^2 & x_i y_i^2 & y_i^3 & x_i^2 y_i^2 & x_i y_i^3 & y_i^4
\end{bmatrix}
\begin{bmatrix} c_1\\ c_2\\ c_3\\ c_4\\ c_5\\ c_6 \end{bmatrix}
= \sum_i \begin{bmatrix} 1\\ x_i\\ y_i\\ x_i^2\\ x_i y_i\\ y_i^2 \end{bmatrix} f_i.$$

Consider the set of nine 2D points P_i ={(1,1), (1,-1), (-1,1), (-1,-1), (0,0), (1,0), (-1,0), (0,1), (0,-1)} with two sets of associated function values f_i^1 ={1.0, -0.5, 1.0, 1.0, -1.0, 0.0, 0.0, 0.0, 0.0} and f_i^2 ={1.0, -1.0, 0.0, 0.0, 1.0, 0.0, -1.0, -1.0, 1.0}. Figure 1 shows the fit functions for the scalar fields f_i^1 and f_i^2.

[1] at the time of writing this report, [Press et al. 1992] was available online in pdf format through `http://www.nr.com/`

[2] `http://math.nist.gov/tnt/`

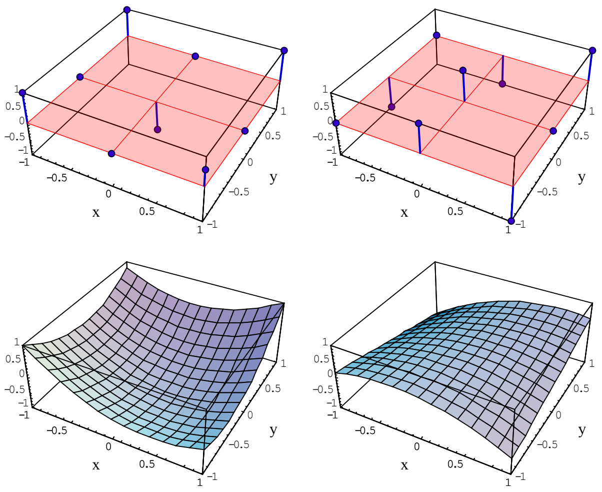

Figure 1: Fitting bivariate, quadratic polynomials to 2D scalar fields: the top row shows the two sets of nine data points (see text), the bottom row shows the least squares fit function. The coefficient vectors $[c_1, \dots, c_6]^T$ are $[-0.834, -0.25, 0.75, 0.25, 0.375, 0.75]^T$ (left column) and $[0.334, 0.167, 0.0, -0.5, 0.5, 0.0]^T$.

Method of Normal Equations. For a different but also very common notation, note that the solution for $\mathbf{c}$ in Eqn. (3) solves the following (generally over-constrained) LSE ($\mathbf{Bc} = \mathbf{f}$) in the least-squares sense

$$\begin{bmatrix} \mathbf{b}^T(\mathbf{x}_1) \\ \vdots \\ \mathbf{b}^T(\mathbf{x}_N) \end{bmatrix} \mathbf{c} = \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix}, \tag{5}$$

using the method of normal equations

$$\begin{aligned} \mathbf{B}^T\mathbf{B}\mathbf{c} &= \mathbf{B}^T\mathbf{f} \\ \mathbf{c} &= (\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T\mathbf{f}. \end{aligned} \tag{6}$$

Please verify that Eqns. (4) and (6) are identical.

2 WLS Approximation

Problem Formulation. In the weighted least squares formulation, we use the error functional $J_{WLS} = \sum_i \theta(\|\bar{\mathbf{x}} - \mathbf{x}_i\|)\ \|f(\mathbf{x}_i) - f_i\|^2$ for a fixed point $\bar{\mathbf{x}} \in \mathbb{R}^d$, which we minimize

$$\min_{f \in \prod_m^d} \sum_i \theta(\|\bar{\mathbf{x}} - \mathbf{x}_i\|)\ \|f(\mathbf{x}_i) - f_i\|^2, \tag{7}$$

similar to (1), only that now the error is weighted by $\theta(d)$ where d_i are the Euclidian distances between $\bar{\mathbf{x}}$ and the positions of data points $\mathbf{x}_i$.

The unknown coefficients we wish to obtain from the solution to (7) are weighted by distance to $\bar{\mathbf{x}}$ and therefore a function of $\bar{\mathbf{x}}$. Thus, the local, weighted least squares approximation in $\bar{\mathbf{x}}$ is written as

$$f_{\bar{\mathbf{x}}}(\mathbf{x}) = \mathbf{b}(\mathbf{x})^T\mathbf{c}(\bar{\mathbf{x}}) = \mathbf{b}(\mathbf{x}) \cdot \mathbf{c}(\bar{\mathbf{x}}), \tag{8}$$

and only defined locally within a distance R around $\bar{\mathbf{x}}$, i.e. $\|\mathbf{x} - \bar{\mathbf{x}}\| < R$.

Weighting Function. Many choices for the weighting function θ have been proposed in the literature, such as a Gaussian

$$\theta(d) = e^{-\frac{d^2}{h^2}}, \tag{9}$$

where h is a spacing parameter which can be used to smooth out small features in the data, see [Levin 2003; Alexa et al. 2003]. Another popular weighting function with compact support is the Wendland function [Wendland 1995]

$$\theta(d) = (1 - d/h)^4(4d/h + 1). \tag{10}$$

This function is well defined on the interval $d \in [0, h]$ and furthermore, $\theta(0) = 1$, $\theta(h) = 0$, $\theta'(h) = 0$ and $\theta''(h) = 0$ (C^2 continuity). Several authors suggest using weighting functions of the form

$$\theta(d) = \frac{1}{d^2 + \varepsilon^2}. \tag{11}$$

Note that setting the parameter ε to zero results in a singularity at $d = 0$, which forces the MLS fit function to interpolate the data, as we will see later.

Solution. Analogous to Section 1, we take partial derivatives of the error functional J_{WLS} with respect to the unknown coefficients $\mathbf{c}(\bar{\mathbf{x}})$

$$\begin{aligned} \sum_i \theta(d_i)\ 2\mathbf{b}(\mathbf{x}_i)[\mathbf{b}(\mathbf{x}_i)^T\mathbf{c}(\bar{\mathbf{x}}) - f_i] &= \\ 2\sum_i [\theta(d_i)\mathbf{b}(\mathbf{x}_i)\mathbf{b}(\mathbf{x}_i)^T\mathbf{c}(\bar{\mathbf{x}}) - \theta(d_i)\mathbf{b}(\mathbf{x}_i)f_i] &= \mathbf{0}, \end{aligned}$$

where $d_i = \|\bar{\mathbf{x}} - \mathbf{x}_i\|$. We divide by the constant and rearrange to obtain

$$\sum_i \theta(d_i)\mathbf{b}(\mathbf{x}_i)\mathbf{b}(\mathbf{x}_i)^T\mathbf{c}(\bar{\mathbf{x}}) = \sum_i \theta(d_i)\mathbf{b}(\mathbf{x}_i)f_i, \tag{12}$$

and solve for the coefficients

$$c(\bar{\mathbf{x}}) = [\sum_i \theta(d_i)\mathbf{b}(\mathbf{x}_i)\mathbf{b}(\mathbf{x}_i)^T]^{-1} \sum_i \theta(d_i)\mathbf{b}(\mathbf{x}_i)f_i. \tag{13}$$

Obviously, the only difference between Eqns. (4) and (13) are the weighting terms. Note again though, that whereas the coefficients $\mathbf{c}$ in Eqn. (4) are global, the coefficients $c(\bar{\mathbf{x}})$ are local and need to be recomputed for every $\bar{\mathbf{x}}$. If the square matrix $\mathbf{A}_{WLS} = \sum_i \theta(d_i)\mathbf{b}(\mathbf{x}_i)\mathbf{b}(\mathbf{x}_i)^T$ (often termed the *Moment Matrix*) is nonsingular (i.e. $det(\mathbf{A}_{WLS}) \neq 0$), substituting Eqn. (13) into Eqn. (8) provides the fit function $f_{\bar{\mathbf{x}}}(\mathbf{x})$.

Global Approximation using a Partition of Unity (PU). By fitting polynomials at $j \in [1 \dots n]$ discrete, fixed points $\bar{\mathbf{x}}_j$ in the parameter domain Ω, we can assemble a global approximation to our data by ensuring that every point in Ω is covered by at least one approximating polynomial, i.e. the support of the weight functions θ_j centered at the points $\bar{\mathbf{x}}_j$ covers Ω

$$\Omega = \bigcup_j \mathrm{supp}(\theta_j).$$

Proper weighting of these approximations can be achieved by constructing a Partition of Unity (PU) from the θ_j [Shepard 1968]

$$\varphi_j(\mathbf{x}) = \frac{\theta_j(\mathbf{x})}{\sum_{k=1}^n \theta_k(\mathbf{x})}, \tag{14}$$

where $\sum_j \varphi_j(\mathbf{x}) \equiv 1$ everywhere in Ω. The global approximation then becomes

$$f(\mathbf{x}) = \sum_j \varphi_j(\mathbf{x})\ \mathbf{b}(\mathbf{x})^T\mathbf{c}(\bar{\mathbf{x}}_j). \tag{15}$$

A Numerical Issue. To avoid numerical instabilities due to possibly large numbers in $\mathbf{A}_{WLS}$ it can be beneficial to perform the fitting procedure in a local coordinate system relative to $\bar{\mathbf{x}}$, i.e. to shift $\bar{\mathbf{x}}$ into the origin. We therefore rewrite the local fit function in $\bar{\mathbf{x}}$ as

$$f_{\bar{\mathbf{x}}}(\mathbf{x}) = \mathbf{b}(\mathbf{x}-\bar{\mathbf{x}})^T \mathbf{c}(\bar{\mathbf{x}}) = \mathbf{b}(\mathbf{x}-\bar{\mathbf{x}}) \cdot \mathbf{c}(\bar{\mathbf{x}}), \quad (16)$$

the associated coefficients as

$$c(\bar{\mathbf{x}}) = \Big[\sum_i \theta(d_i)\mathbf{b}(\mathbf{x}_i-\bar{\mathbf{x}})\mathbf{b}(\mathbf{x}_i-\bar{\mathbf{x}})^T\Big]^{-1} \sum_i \theta(d_i)\mathbf{b}(\mathbf{x}_i-\bar{\mathbf{x}}) f_i, \quad (17)$$

and the global approximation as

$$f(\mathbf{x}) = \sum_j \varphi_j(\mathbf{x})\, \mathbf{b}(\mathbf{x}-\bar{\mathbf{x}}_j)^T \mathbf{c}(\bar{\mathbf{x}}_j). \quad (18)$$

3 MLS Approximation and Interpolation

Method. The MLS method was proposed by Lancaster and Salkauskas [Lancaster and Salkauskas 1981] for smoothing and interpolating data. The idea is to start with a weighted least squares formulation for an arbitrary fixed point in $\mathbb{R}^d$, see Section 2, and then *move* this point over the entire parameter domain, where a weighted least squares fit is computed and evaluated for each point individually. It can be shown that the global function $f(\mathbf{x})$, obtained from a set of local functions

$$f(\mathbf{x}) = f_{\mathbf{x}}(\mathbf{x}), \quad \min_{f_{\mathbf{x}} \in \prod_m^d} \sum_i \theta(\|\mathbf{x}-\mathbf{x}_i\|)\, \|f_{\mathbf{x}}(\mathbf{x}_i) - f_i\|^2 \quad (19)$$

is continuously differentiable if and only if the weighting function is continuously differentiable, see Levins work [Levin 1998; Levin 2003].

So instead of constructing the global approximation using Eqn. (15), we use Eqns. (8) and (13) (or (16) and (17)) and construct and evaluate a local polynomial fit continuously over the *entire* domain Ω, resulting in the MLS fit function. As previously hinted at, using (11) as the weighting function with a very small ε assigns weights close to infinity near the input data points, forcing the MLS fit function to interpolate the prescribed function values in these points. Therefore, by varying ε we can directly influence the approximatimg/interpolating nature of the MLS fit function.

4 Applications

Least Squares, Weighted Least Squares and Moving Least Squares, have become widespread and very powerful tools in Computer Graphics. They have been successfully applied to surface reconstruction from points [Alexa et al. 2003] and other point set surface definitions [Amenta and Kil 2004], interpolating and approximating implicit surfaces [Shen et al. 2004], simulating [Belytschko et al. 1996] and animating [Müller et al. 2004] elastoplastic materials, Partition of Unity implicits [Ohtake et al. 2003], and many other research areas.

In [Alexa et al. 2003] a point-set, possibly acquired from a 3D scanning device and therefore noisy, is replaced by a representation point set derived from the MLS surface defined by the input point-set. This is achieved by down-sampling (i.e. iteratively removing points which have little contribution to the shape of the surface) or up-sampling (i.e. adding points and projecting them to the MLS surface where point-density is low). The projection procedure has recently been augmented and further analyzed in the work of Amenta and Kil [Amenta and Kil 2004]. Shen et. al [Shen et al. 2004] use an MLS formulation to derive implicit functions from polygon soup. Instead of solely using value constraints at points (as shown in this report) they also add value constraints integrated over polygons and normal constraints.

References

ALEXA, M., BEHR, J., COHEN-OR, D., FLEISHMAN, S., LEVIN, D., AND T. SILVA, C. 2003. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics 9*, 1, 3–15.

AMENTA, N., AND KIL, Y. 2004. Defining point-set surfaces. In *Proceedgins of ACM SIGGRAPH 2004*.

BELYTSCHKO, T., KRONGAUZ, Y., ORGAN, D., FLEMING, M., AND KRYSL, P. 1996. Meshless methods: An overview and recent developments. *Computer Methods in Applied Mechanics and Engineering 139*, 3, 3–47.

FRIES, T.-P., AND MATTHIES, H. G. 2003. Classification and overview of meshfree methods. Tech. rep., TU Brunswick, Germany Nr. 2003-03.

LANCASTER, P., AND SALKAUSKAS, K. 1981. Surfaces generated by moving least squares methods. *Mathematics of Computation 87*, 141–158.

LEVIN, D. 1998. The approximation power of moving least-squares. *Math. Comp. 67*, 224, 1517–1531.

LEVIN, D., 2003. Mesh-independent surface interpolation, to appear in 'geometric modeling for scientific visualization' edited by brunnett, hamann and mueller, springer-verlag.

MÜLLER, M., KEISER, R., NEALEN, A., PAULY, M., GROSS, M., AND ALEXA, M. 2004. Point based animation of elastic, plastic and melting objects. In *Proceedings of 2004 ACM SIGGRAPH Symposium on Computer Animation*.

OHTAKE, Y., BELYAEV, A., ALEXA, M., TURK, G., AND SEIDEL, H.-P. 2003. Multi-level partition of unity implicits. *ACM Trans. Graph. 22*, 3, 463–470.

PRESS, W., TEUKOLSKY, S., VETTERLING, W., AND FLANNERY, B. 1992. *Numerical Recipes in C - The Art of Scientific Computing*, 2nd ed. Cambridge University Press.

SHEN, C., O'BRIEN, J. F., AND SHEWCHUK, J. R. 2004. Interpolating and approximating implicit surfaces from polygon soup. In *Proceedings of ACM SIGGRAPH 2004*, ACM Press.

SHEPARD, D. 1968. A two-dimensional function for irregularly spaced data. In *Proc. ACM Nat. Conf.*, 517–524.

WENDLAND, H. 1995. Piecewise polynomial, positive definite and compactly supported radial basis functions of minimal degree. *Advances in Computational Mathematics 4*, 389–396.

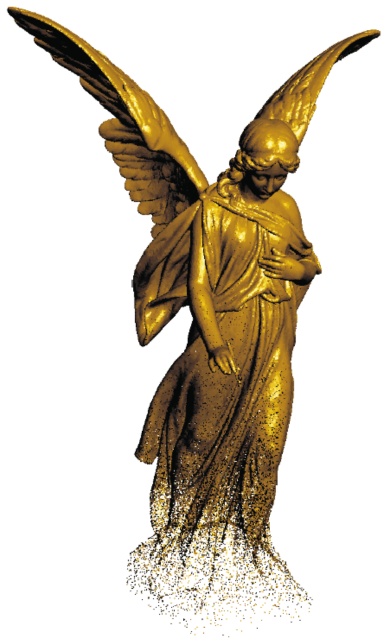

Figure 2: The MLS surface of a point-set with varying density (the density is reduced along the vertical axis from top to bottom). The surface is obtained by applying the projection operation described by Alexa et. al. [2003]. Image courtesy of Marc Alexa.