Least-Squares Fitting of Data with B-Spline Surfaces

David Eberly
Geometric Tools, LLC
http://www.geometrictools.com/

Created: September 13, 2008

Contents

This document describes how to fit a set of data points with a B-spline tensor product surface using a least-squares algorithm. A typical application is to fit height-field data on a rectangular grid with a control-point surface as a way of reducing the amount of data needed to represent the height field.

1 Definition of B-Spline Tensor Product Surfaces

A *B-spline tensor product surface* is defined for a 2-dimensional array of $(n_0+1)\times(n_1+1)$ control points $\mathbf{Q}_{i_0 i_1}$ with $0 \le i_0 \le n_0$ and $0 \le i_1 \le n_1$,

$$\mathbf{X}(u,v) = \sum_{i_0=0}^{n_0} \sum_{i_1=0}^{n_1} N_{i_0,d_0}(u) N_{i_1,d_1}(v) \mathbf{Q}_{i_0 i_1} \tag{1}$$

The numbers d_0 and d_1 are the *degrees* for the surface. They must satisfy $1 \le d_0 \le n_0$ and $1 \le d_1 \le n_1$. The functions $N_{i_0,d_0}(u)$ and $N_{i_1,d_1}(v)$ are the *B-spline basis functions*. In the following discussion, I drop the subscripts for simplicity and use t to denote the independent variable of the function. The basis functions $N_{i,d}(t)$ are defined recursively and require selection of a sequence of scalars t_i for $0 \le i \le n+d+1$. The sequence is nondecreasing; that is, $t_i \le t_{i+1}$. Each t_i is referred to as a *knot*, the total sequence a *knot vector*. The basis function that starts the recursive definition is

$$N_{i,0}(t) = \begin{cases} 1, & t_i \le t < t_{i+1} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

for $0 \le i \le n+d$. The recursion itself is

$$N_{i,j}(t) = \frac{t - t_i}{t_{i+j} - t_i} N_{i,j-1}(t) + \frac{t_{i+j+1} - t}{t_{i+j+1} - t_{i+1}} N_{i+1,j-1}(t) \tag{3}$$

for $1 \le j \le d$ and $0 \le i \le n+d-j$. The *support* of a function is the smallest closed interval on which the function has at least one nonzero value. The support of $N_{i,0}(t)$ is clearly $[t_i, t_{i+1}]$. In general, the support of $N_{i,j}(t)$ is $[t_i, t_{i+j+1}]$. This fact means that locally the curve is influenced by only a small number of control points, a property called *local control*.

The main classification of the knot vector is that it is either *open* or *periodic*. If open, the knots are either *uniform* or *nonuniform*. Periodic knot vectors have uniformly spaced knots. The use of the term *open* is perhaps a misnomer since you can construct a closed B-spline surface from open knot vectors. The standard way to construct a closed surface uses periodic knot vectors. Uniform knots are

$$t_i = \begin{cases} 0 & , \quad 0 \le i \le d \\ \frac{i-d}{n+1-d} & , \quad d+1 \le i \le n \\ 1 & , \quad n+1 \le i \le n+d+1 \end{cases} \tag{4}$$

Periodic knots are

$$t_i = \frac{i-d}{n+1-d}, \quad 0 \le i \le n+d+1 \tag{5}$$

Equations (2) and (3) allow you to recursively evaluate the B-spline surface, but there are faster ways based on the local control. This document does not cover the various evaluation schemes. You may find this topic in any book on B-splines and most likely at online sites.

2 Least-Squares Fitting

The sample data points are $(r_{j_0}, s_{j_1}, \mathbf{P}_{j_0 j_1})$ with $0 \le j_0 \le m_0$ and $0 \le j_1 \le m_1$. It is assumed that $r_0 < r_1 < \cdots < r_{m_0}$ and $s_0 < s_1 < \cdots < s_{m_1}$. A B-spline surface that fits the data is parameterized by $(u, v) \in [0, 1]^2$, so the sample r-values and s-values need to be mapped to the parameter domain by $u_{j_0} = (r_{j_0} - r_0)/(r_{m_0} - r_0)$ and $v_{j_1} = (s_{j_1} - s_0)/(s_{m_1} - s_0)$.

The fitted B-spline surface is formally presented in Equation (1), but the control points $\mathbf{Q}_{i_0 i_1}$ are unknown quantities to be determined later. The control points may be arranged formally as an $(n_0 + 1) \times (n_1 + 1)$ matrix,

$$\hat{Q} = \begin{bmatrix} \mathbf{Q}_{00} & \cdots & \mathbf{Q}_{0n_1} \\ \vdots & \ddots & \vdots \\ \mathbf{Q}_{n_0 0} & \cdots & \mathbf{Q}_{n_0 n_1} \end{bmatrix} \tag{6}$$

Similarly, the samples $\mathbf{P}_{j_0 j_1}$ may be arranged formally as an $(m_0 + 1) \times (m_1 + 1)$ matrix,

$$\hat{P} = \begin{bmatrix} \mathbf{P}_{00} & \cdots & \mathbf{P}_{0m_1} \\ \vdots & \ddots & \vdots \\ \mathbf{P}_{m_0 0} & \cdots & \mathbf{P}_{m_0 m_1} \end{bmatrix} \tag{7}$$

For a specifed set of control points, the *least-squares error function* between the B-spline surface and sample points is the scalar-valued function

$$E(\hat{Q}) = \frac{1}{2} \sum_{j_0=0}^{m_0} \sum_{j_1=0}^{m_1} \left| \sum_{i_0=0}^{n_0} \sum_{i_1=0}^{n_1} N_{i_0,d_0}(u_{j_0}) N_{i_1,d_1}(v_{j_1}) \mathbf{Q}_{i_0 i_1} - \mathbf{P}_{j_0 j_1} \right|^2 \tag{8}$$

The half term is just for convenience in the calculations. The quantity

$$\sum_{i_0=0}^{n_0} \sum_{i_1=0}^{n_1} N_{i_0,d_0}(u_{j_0}) N_{i_1,d_1}(v_{j_1}) \mathbf{Q}_{i_0 i_1} \tag{9}$$

is the point on the B-spline surface at the scaled sample parameters (u_{j_0}, v_{j_1}). The term within the double summation (with indices j_0 and j_1) on the right-hand side of equation (8) measures the squared distance between the sample point and its corresponding surface point. The error function measures the total accumulation of squared distances. The hope is that we may choose the control points to make this error as small as possible.

The minimization is a calculus problem. The function E is quadratic in the components of $\hat{Q}$, its graph a paraboloid (in high dimensional space), so it must have a global minimum that occurs when all its first-order partial derivatives are zero. That is, the vertex of the parabola occurs where the first derivatives are zero. The first-order partial derivatives are written in terms of the control points $\mathbf{Q}_{i_0 i_1}$ rather than in terms of the components of the control points:

$$\begin{aligned} \frac{\partial E}{\partial \mathbf{Q}_{k_0 k_1}} &= \textstyle\sum_{j_0=0}^{m_0} \sum_{j_1=0}^{m_1} \left(\sum_{i_0=0}^{n_0} \sum_{i_1=0}^{n_1} N_{i_0,d_0}(u_{j_0}) N_{i_1,d_1}(v_{j_1}) \mathbf{Q}_{i_0 i_1} - \mathbf{P}_{j_0 j_1} \right) N_{k_0,d_0}(u_{j_0}) N_{k_1,d_1}(v_{j_1}) \\ &= \textstyle\sum_{j_0=0}^{m_0} \sum_{j_1=0}^{m_1} \sum_{i_0=0}^{n_0} \sum_{i_1=0}^{n_1} N_{i_0,d_0}(u_{j_0}) N_{i_1,d_1}(v_{j_1}) N_{k_0,d_0}(u_{j_0}) N_{k_1,d_1}(v_{j_1}) \mathbf{Q}_{i_0 i_1} \\ &\quad \textstyle - \sum_{j_0=0}^{m_0} \sum_{j_1=0}^{m_1} N_{k_0,d_0}(u_{j_0}) N_{k_1,d_1}(v_{j_1}) \mathbf{P}_{j_0 j_1} \\ &= \textstyle\sum_{j_0=0}^{m_0} \sum_{j_1=0}^{m_1} \sum_{i_0=0}^{n_0} \sum_{i_1=0}^{n_1} a_{j_0 i_0} b_{j_1 i_1} a_{j_0 k_0} b_{j_1 k_1} \mathbf{Q}_{i_0 i_1} - \sum_{j_0=0}^{m_0} \sum_{j_1=0}^{m_1} a_{j_0 k_0} b_{j_1 k_1} \mathbf{P}_{j_0 j_1} \end{aligned} \tag{10}$$

where $a_{j_0 i_0} = N_{i_0,d_0}(u_{j_0})$ and $b_{j_1 i_1} = N_{i_1,d_1}(v_{j_1})$. Equation (10) in matrix form is

$$\frac{\partial E}{\partial \hat{Q}} = A^{\mathrm{T}} A \hat{Q} B^{\mathrm{T}} B - A^{\mathrm{T}} \hat{P} B \tag{11}$$

where $\partial E/\partial \hat{Q}$ is an $(n_0+1)\times(n_1+1)$ matrix, $A = [a_{rc}]$ is a $(m_0+1)\times(n_0+1)$ matrix, and $B = [b_{rc}]$ is a $(m_1+1)\times(n_1+1)$ matrix.

Setting the partial derivatives equal to the zero matrix $\hat{0}$ leads to the matrix system of equations,

$$A^{\mathrm{T}} A \hat{Q} B^{\mathrm{T}} B - A^{\mathrm{T}} \hat{P} B = \hat{0} \tag{12}$$

The matrix $A^{\mathrm{T}}A$ is symmetric, a property that is desirable in the numerical solution of systems. Moreover, the matrix A is *banded.* This is a generalization of *tridiagonal.* A banded matrix has a diagonal with (potentially) nonzero entries. It has a contiguous set of *upper bands* and a contiguous set of *lower bands*, each band with (potentially) nonzero entries. All other entries in the matrix are zero. In our case, the number of upper bands and the number of lower bands are the same, namely $d_0 + 1$. The bandedness is a consequence of the local control for B-spline surfaces (the supports of the B-spline basis functions are bounded intervals). A similar analysis shows that $B^{\mathrm{T}}B$ is symmetric and banded with $d_1 + 1$ upper bands and $d_1 + 1$ lower bands.

The direct approach to solving the equation (12) is to invert the matrices $A^{\mathrm{T}}A$ and $B^{\mathrm{T}}B$,

$$\hat{Q} = \left(A^{\mathrm{T}}A\right)^{-1} A^{\mathrm{T}} \hat{P} B \left(B^{\mathrm{T}}B\right)^{-1} = \left[\left(A^{\mathrm{T}}A\right)^{-1} A^{\mathrm{T}}\right] \hat{P} \left[\left(B^{\mathrm{T}}B\right)^{-1} B^{\mathrm{T}}\right]^{\mathrm{T}} = XPY^{\mathrm{T}} \tag{13}$$

where the last equality defines matrices X and Y. The problem, though, is that the matrix inversions can be ill conditioned because the matrices have eigenvalues that are nearly zero. The ill conditioning causes a Gaussian elimination, even with full pivoting, to have problems. For the application of fitting a height field, the heights are sampled on a rectangular lattice. In this case, the ill conditioning is not an issue as long as you choose a B-spline surface with uniform knots. Regardless, an approach different from the direct inversions is called for, both to minimize the effects of ill conditioning *and* to take advantage of the bandedness of the matrices. Recall that Gaussian elimination to solve a linear system with an $n \times n$ matrix is an $O(n^3)$ algorithm. The solution to a linear system with a tridiagonal matrix is $O(n)$. The same is true for a banded matrix with a small number of bands relative to the size of the matrix.

The numerical method of choice for symmetric, banded matrix systems is the *Cholesky decomposition.* The book *Matrix Computations* by G. Golub and C. van Loan has an excellent discussion of the topic. The algorithm starts with a symmetric matrix and factors it into a lower-triangular matrix, G, times the transpose of that lower-triangular matrix, G^{T}, which is necessarily upper triangular. That is, the Cholesky decomposition is

$$A^{\mathrm{T}}A = GG^{\mathrm{T}} \tag{14}$$

A numerically stable LU solver may be used first to invert G, then to invert G^{T}. For the application of fitting height-field data, the choice of uniform knots leads to good stability, but also required is to make certain that the number of control points is smaller than the number of samples by a half. This is essentially a Nyquist-frequency argument. If you have as many control points as samples, the B-spline surface can have large oscillations.

The matrix X is computed as the solution to $A^{\mathrm{T}}AX = A^{\mathrm{T}}$ using the Cholesky decomposition to invert the matrix $A^{\mathrm{T}}A$. Similarly, the matrix Y is computed as the solution to $B^{\mathrm{T}}BY = B^{\mathrm{T}}$. The matrix of control points, $\hat{Q}$, is then computed from Equation (13).

3 Implementation

A sample application to illustrate the fit is in the folder

```
GeometricTools/WildMagic4/SampleFoundation/BSplineSurfaceFitter
```

A 64×64 height field is fitted with a B-spline surface that has 32×32 control points. The fitted surface is evaluated on a 64×64 grid so that the fitted surface and original height field may be compared.