Least-Squares Fitting of Data with B-Spline Curves

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

Created: December 2, 2005
Last Modified: September 13, 2008

Contents

This document describes how to fit a set of data points with a B-spline curve using a least-squares algorithm. The construction allows for any dimension for the data points. A typical application is to fit keyframes for animation sequences, whether the data is positional (3D) or rotational using quaternions (4D). In the latter case, the B-spline curve is not necessarily on the unit hypersphere in 4D, but the curve evaluations may be normalized to force the results onto that hypersphere.

1 Definition of B-Spline Curves

A *B-spline curve* is defined for a collection of $n+1$ control points $\{\mathbf{Q}_i\}_{i=0}^n$ by

$$\mathbf{X}(t) = \sum_{i=0}^{n} N_{i,d}(t)\mathbf{Q}_i \tag{1}$$

The control points can be any dimension, but all of the same dimension. The *degree* of the curve is d and must satisfy $1 \le d \le n$. The functions $N_{i,d}(t)$ are the *B-spline basis functions*, which 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 curve from an open knot vector. The standard way to construct a closed curve 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 curve, 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 data points are $\{(s_k, \mathbf{P}_k)\}_{k=0}^m$, where s_k are the sample times and $\mathbf{P}_k$ are the sample data. The sample times are assumed to be increasing: $s_0 < s_1 < \ldots < s_m$. A B-spline curve that fits the data is parameterized by $t \in [0,1]$, so the sample times need to be mapped to the parameter domain by $t_k = (s_k - s_0)/(s_m - s_0)$.

The fitted B-spline curve is formally presented in equation (1), but the control points $\mathbf{Q}_i$ are unknown quantities to be determined later. The control points are considered to be column vectors, and the collection of control points may be arranged into a single column vector

$$\hat{Q} = \begin{bmatrix} \mathbf{Q}_0 \\ \mathbf{Q}_1 \\ \vdots \\ \mathbf{Q}_n \end{bmatrix}. \tag{6}$$

Similarly, the samples $\mathbf{P}_k$ are considered to be column vectors, and the collection written as a single column vector

$$\hat{P} = \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \vdots \\ \mathbf{P}_m \end{bmatrix}. \tag{7}$$

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

$$E(\hat{Q}) = \frac{1}{2} \sum_{k=0}^{m} \left| \sum_{j=0}^{n} N_{j,d}(t_k)\mathbf{Q}_j - \mathbf{P}_k \right|^2 \tag{8}$$

The half term is just for convenience in the calculations. The quantity $\sum_{j=0}^n N_{j,d}(t_k)\mathbf{Q}_j$ is the point on the B-spline curve at the scaled sample time t_k. The term within the summation on the right-hand side of equation (8) measures the squared distance between the sample point and its corresponding curve 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 $\mathbf{Q}_i$ rather than in terms of the components of the control points:

$$\begin{aligned} \frac{\partial E}{\partial \mathbf{Q}_i} &= \sum_{k=0}^m \left(\sum_{j=0}^n N_{j,d}(t_k)\mathbf{Q}_j - \mathbf{P}_k \right) N_{i,d}(t_k) \\ &= \sum_{k=0}^m \sum_{j=0}^n N_{i,d}(t_k) N_{j,d}(t_k)\mathbf{Q}_j - \sum_{k=0}^m N_{i,d}(t_k)\mathbf{P}_k \\ &= \sum_{k=0}^m \sum_{j=0}^n a_{ki} a_{kj} \mathbf{Q}_j - \sum_{k=0}^m a_{ki}\mathbf{P}_k \end{aligned} \tag{9}$$

where $a_{rc} = N_{c,d}(t_r)$, and for $0 \le i \le n$. Setting the partial derivatives equal to the zero vector leads to the system of equations

$$\mathbf{0} = \sum_{k=0}^{m} \sum_{j=0}^{n} a_{ik} a_{jk} \mathbf{Q}_j - \sum_{k=0}^{m} a_{ik} \mathbf{P}_k = A^{\mathrm{T}} A \hat{Q} - A^{\mathrm{T}} \hat{P} \tag{10}$$

where $A = [a_{rc}]$ is a matrix with $n+1$ rows and $m+1$ columns. The matrix A^{T} is the transpose of A. This system of equations is in a familiar form of a least-squares problem. Recall that such problems arise when wanting to solve $A\mathbf{x} = \mathbf{b}$. If the system does not have a solution, the next best thing is to construct $\mathbf{x}$ so that $|A\mathbf{x} - \mathbf{b}|$ is as small as possible. The minimization leads to the linear system $A^{\mathrm{T}} A\mathbf{x} = A^{\mathrm{T}}\mathbf{b}$.

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 $d+1$. The bandedness is a consequence of the local control for B-spline curves (the supports of the B-spline basis functions are bounded intervals).

The direct approach to solving the equation (10) is to invert the coefficient matrix. The equation $A^{\mathrm{T}} A\hat{Q} = A^{\mathrm{T}} \hat{P}$ implies

$$\hat{Q} = \left(A^{\mathrm{T}} A\right)^{-1} A^{\mathrm{T}} \hat{P} = \left[\left(A^{\mathrm{T}} A\right)^{-1} A^{\mathrm{T}}\right] \hat{P} = X\hat{P} \tag{11}$$

where the last equality defines the matrix X. The problem, though, is that the matrix inversion can be ill conditioned because the matrix has eigenvalues that are nearly zero. The ill conditioning causes a Gaussian elimination, even with full pivoting, to have problems. For the application of fitting keyframe data, the keyframes tend to be sampled at evenly spaced time intervale. In this case, the ill conditioning is not an issue as long as you choose a B-spline curve with uniform knots. Regardless, an approach different from the direct inversion is called for, both to minimize the effects of ill conditioning *and* to take advantage of the bandedness of the matrix. 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. In our case, the Cholesky decomposition is

$$A^{\mathrm{T}} A = GG^{\mathrm{T}} \tag{12}$$

A numerically stable LU solver may be used first to invert G, then to invert G^{T}. For the application of fitting keyframe 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 curve 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 $A^{\mathrm{T}} A$. The vector of control points, $\hat{Q}$, is then computed from Equation (11).

3 Implementation

A sample application to illustrate the fit is in the folder

```
GeometricTools/WildMagic4/SampleFoundation/BSplineCurveFitter
```

In the application, a spiral curve is generated as a polyline of 1000 sample points. A B-spline curve is created using the algorithm of this document. The initial degree is 3 and the initial number of control points is 500. The display shows the current degree and the current number of control points. It also shows the average distance error between the original samples and the resample B-spline curve points as well as showing the root-mean-square error for the distances.

You can increase or decrease the degree and the number of control points.

key	action
'd'	decrease the degree by 1
'D'	increase the degree by 1
's'	decrease the number of control points by 1 (s for small)
'S'	increase the number of control points by 1 (S for small)
'm'	decrease the number of control points by 10 (m for medium)
'M'	increase the number of control points by 10 (M for medium)
'l'	decrease the number of control points by 100 (l for large)
'L'	increase the number of control points by 100 (L for large)
ESC	terminate the application

You may rotate the scene using the mouse and the built-in virtual trackball. Of interest is when the number of control points is between 20 and 30. This is where you will see big differences between the curves. Note that you get a very good fit with an extremely small number of control points.