AN EFFICIENT ALGORITHM FOR GENERATING B–SPLINE INTERPOLATION CURVES AND SURFACES FROM B–SPLINE APPROXIMATIONS

by

HUI PING WANG, DENTON E. HEWGILL
AND
GEOFFREY W. VICKERS

DMS–525–IR October 1989

An efficient algorithm for generating B-spline interpolation curves and surfaces from B-spline approximations

Hui Ping Wang[1], **Denton E. Hewgill**[2], **Geoffrey W. Vickers**[3]

Abstract

A useful and simple algorithm is presented for interactively generating B-spline interpolation curves and surfaces from B-spline approximation solutions. The difference between the data points and the B-spline approximation is used to modify the control vertices in order to generate a succession of B-spline approximations which converge rapidly to the interpolation solution. The intermediate B-spline approximations can be viewed interactively and the most appropriate solution selected for a particular application.

Keywords

B-spline approximation and interpolation, curves and surfaces, efficient iterative algorithm and control polygon.

[1]Faculty of Engineering, University of Victoria, Victoria, B.C., Canada V8W 2Y2.

[2]Dept. of Mathematics

[3]Faculty of Engineering, Director of CAD/CAM

1. Introduction

B-splines (de Boor [3]; Gordon and Reisenfeld [4]) have found many applications for defining curves and surfaces through sets of data points. If the curve or surface fits among the data, thereby smoothing the shape, the curve is called an approximation. If the curve or surface passes through the data, then it is called an interpolation. Typical B-spline approximation and interpolation curves are shown in Fig 1.

The parametric B-splines for curves can be defined by

$$s(u) = \sum_{i=0}^{n} p_i B_i(u)$$

where the indexed variables p_i are the control polygon vertices, and $B_i(u)$ are the normalized B-spline basis functions of order K (polynomial degree $K-1$) defined on the knot sequence for the curve. The variable u is a parameter and $s(u)$ represents the position of the curve in the plane or space. Likewise,

$$s(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} p_{i,j} N_i(u) M_j(v)$$

defines a B-spline surface where p_{ij} is the control polyhedron, $N_i(u)$ and $M_j(v)$ are the B-spline basis functions in the u and v direction on the surface.

When a B-spline approximation is required, a numerically efficient algorithm for generating the B-splines from the control polygon has been given by Cox [2] and de Boor[3]. The B-spline solution fits within the convex hull defined by the control points and has the effect of smoothing these points. The spline does not in general go through all control points. If the B-spline is required to go through all data points, then a set of suitable of control points must be found.

In order to obtain a solution which passes through a data set given by d_j for $0 \leq j \leq n$, the set of linear equations defined by

$$d_j = s(u_j) = \sum_{i=0}^{n} p_i B_i(u_j)$$

for each knot value u_j, $0 \leq j \leq n$, must be solved for the coefficients of the spline. Since B-splines have local support, the case of fourth order B-spline curves has at each knot u_j

$$\sum_{i=0}^{n} p_i B_i(u_j) = 0.16666 p_{j-1} + 0.66666 p_j + 0.16666 p_{j+1}$$

for $0 < j < n$. This gives a tridiagonal form for the coefficient matrix of the set of linear equations

$$d = Bp,$$

where d is the vector of data points and p is the vector of control vertices to be found. Since the matrix B is diagonally dominant, it can be inverted efficiently with LU decomposition using the Thomas algorithm.

In the case of surfaces, the bandwidth of the coefficient matrix is larger so the LU decomposition may no longer be the preferred method for the inversion. An alternate method is to use an iterative method to solve this problem. An advantage of the iterative method is that the solution can be observed interactively and stopped when sufficient accuracy has been obtained. This partial interpolation will not exactly pass through the data and so has the additional advantage that it may not emphasize any small irregularities that may be present in the original data.

In the present work, a sequence of solutions is developed which progresses efficiently from a B-spline approximation to a B-spline interpolation. The approach uses the B-spline algorithm to generate each approximate solution. The difference, or error, between the B-spline approximation solution and the data vertices is used to modify the control vertices and thereby develop a sequence of solutions. The validity of the iterative scheme is based on standard iterative methods to solve linear equations. The method is discussed for fourth order B-splines curves, although the method is suitable for B-splines of any order.

2. Description of iteration scheme

The B-spline approximation has a variation diminishing property (de Boor[3]), which forces the curve to always lie within the convex hull of the enclosing polygon as shown in Fig. 1. In general, there is always a difference between each data vertex and the corresponding point on the curve. To solve the B-spline interpolation problem $d = Bp$, a sequence of intermediate approximation solutions to the interpolation problem is defined in the form

$$p^{k+1} = p^k + (d - Bp^k) \text{ , for } k \geq 0 \text{ with } p^0 = d \ .$$

The vector $d - Bp^k$ can be thought of as the error in the k th approximating B-spline iteration based on the control coefficients p^k. The sequence of control coefficients p^k generates the sequence of approximating splines.

The convergence of p^k can be easily shown by the applications of standard iterative linear algebra theorems. In fact, the scheme is the well known Gauss-Jacobi iterative method. For example in the case of curves with forth order splines, the value of the spline at a knot point u^j given by

$$(Bp^k)_j = 0.16666p^k_{j-1} + 0.66666p^k_j + 0.16666p^k_{j+1},$$

for $0 < j < n$. Since

$$p^{k+1} = d + (I - B)p^k \ ,$$

where I is the identity matrix, the matrix $I - B$ is diagonally dominant, and so the iteration scheme will converge (see e.g. Atkinson [1]). In practice, the iteration is continued until the vector error term $(d - Bp^k)$ is sufficiently small for the application. Fig 2 shows an iterative control vertex moving in order to force the curve through the data point.

The method of SOR or successive overrelaxation can provide a significant improvement in the method while retaining the general simplicity of the argument. To do this, a parameter ω is introduced into the iteration

$$p^{k+1} = p^k + \omega(d - Bp^k) \text{ , for } k \geq 0 \text{ with } p^0 = d \text{ .}$$

The correction then tends to overshoot the error and give a sequence of solutions that converges faster than the Gauss-Jacobi method. Many numerical experiments have been done in the literature to determine suitable values for the relaxation parameters and values of 1.5 to 1.7 are commonly used. For this problem, $\omega = 1.5$ has proved to be useful.

3. Sample fitting problems

To demonstrate the effectiveness of the technique and the rapid convergence from the approximation to the interpolation, a few examples are presented. In Fig 3, the function $y = \sin(\pi x/2)$ was used and the B-spline interpolation solution after 3 SOR iterations was within a tolerance of 0.0001 of the function.

An analytical surface defined by

$$z = \frac{1.25 + \cos(5.4y)}{6 + 6(3x - 1)^2}$$

was examined where the original control polyhedron formed by (11×9) vertices. After four iterations the maximum error between the interpolating surface data points and the given data vertices was 0.0001 with an average error of .00002.

A B-spline fit to a set of data, obtained by stereo photogrammetry of a human face, is given in Fig. 4. The face surface is defined by (80×80) points and the fitted surface by (160×160) points. The B-spline fit in this case was carried out so that the error above a certain level was not adjusted and hence the surface was not modified at these points. As the error was largest at the periphery of the face, this had the effect of adjusting only the interior surface. This selective smoothing give a close interior fit to the face and avoids the edge oscillations that would normally occur with a regularly interpolated surface.

4. Complexity discussion

For the case of curves, the iterative technique for solving the equation has no particular computational advantage over the LU decomposition using the Thomas algorithm. For example, the LU decomposition uses about $4n$ multiplications in each component to completely invert the problem. On the other hand, the iterative method uses about $2nk$ multiplications where k is the number of iterations used. Experience has shown about two to four iterations are required for curves.

In the case of surfaces, the increased bandwith of the coefficient matrix makes the iterative method more attractive. For the case of fourth order B-splines, the coefficient matrix is determined from the equation

$$d_{i,j} = 0.66666^2 p_{i,j} + 0.66666 \times 0.16666(p_{i-1,j} + p_{i+1,j} +$$

$$p_{i,j-1} + p_{i,j+1}) + 0.16666^2(p_{i-1,j-1} + p_{i-1,j+1}$$

$$p_{i+1,j-1} + p_{i+1,j+1}) \ ,$$

for $0 < i < n$ and $0 < j < m$. This expression generates a coefficient matrix with a bandwidth of at least 9 depending on how the points are ordered. Then, if $n = m$ and $N = n^2$, the LU decomposition will need about $2((9-1)/2)^2 N = 32N$ multiplications, whereas the iterative solution will require $3Nk$ multiplications in each component and some extra additions. This means that the iterative solution can take about 10 iterations to complete the solution in about the same time. Experience has shown that 4 to 5 iterations are usually sufficient to obtain a suitable accuracy for the solution with SOR iteration.

There are three main reasons that method is successful:

- usually the full accuracy for an inversion is not required when defining and machining curved surfaces.
- the initial data provides a good approximation to the control vertices and so it is a good starting value for the SOR iteration.

- the method is simple to program for various types of B-splines

5. Conclusions

A algorithm for iteratively generating B-spline interpolation curves and surfaces form the B-spline approximation solution is presented. The algorithm is numerically stable, computationally efficient and can readily handle large amounts of sampled data. The converging sequence of graphical B-spline solutions can be interactively viewed and the most appropriate solution selected for a particular application. Alternatively, the number of iterations can be adjusted to produce a curve or surface that is within a specified tolerance of B-spline interpolation.

References

1. Atkinson K.E.,(1989) *An introduction to numerical analysis*, New York, John Wiley.

2. Cox, M.G.(1972), The numerical evaluation of B-splines, J. Inst. Maths. Applics. 10, 134-149.

3. de Boor, C. (1978), *A practical Guide to Splines*, New York: Springer-Verlag.

4. Gordon, W.J. and Reisenfeld, R.F. (1974), B-spline curves and surfaces, in: Barnhill, R.E. and Reienfeld, R.F., eds., *Computer Aided Geometric Design*, New York, Academic Press.

Acknowledgements

The authors would like to thank Professor James P. Duncan for providing the data for the face. Also, the financial support from the Natural Sciences and Engineering Research Council of Canada is gratefully acknowledged.

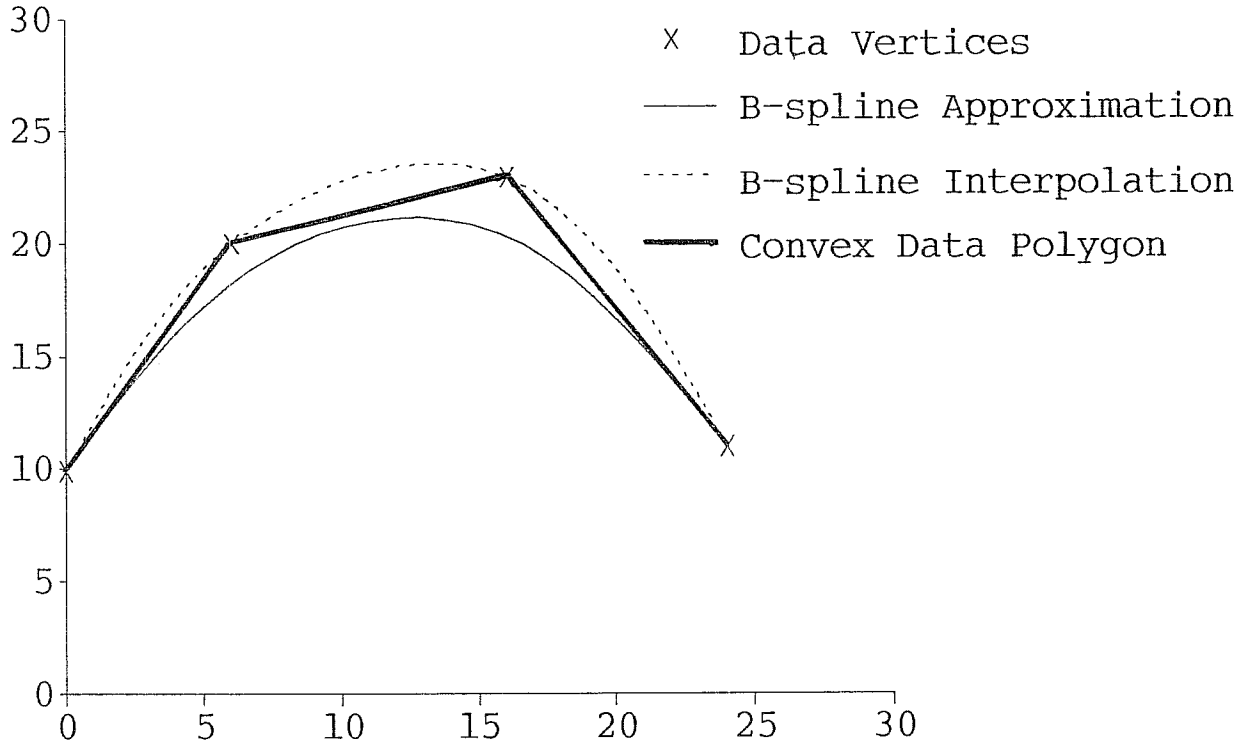
30
25
20
15
10
5
0
0
5
10
15
20
25
30
Data Vertices
B-spline Approximation
B-spline Interpolation
Convex Data Polygon

Figure 1

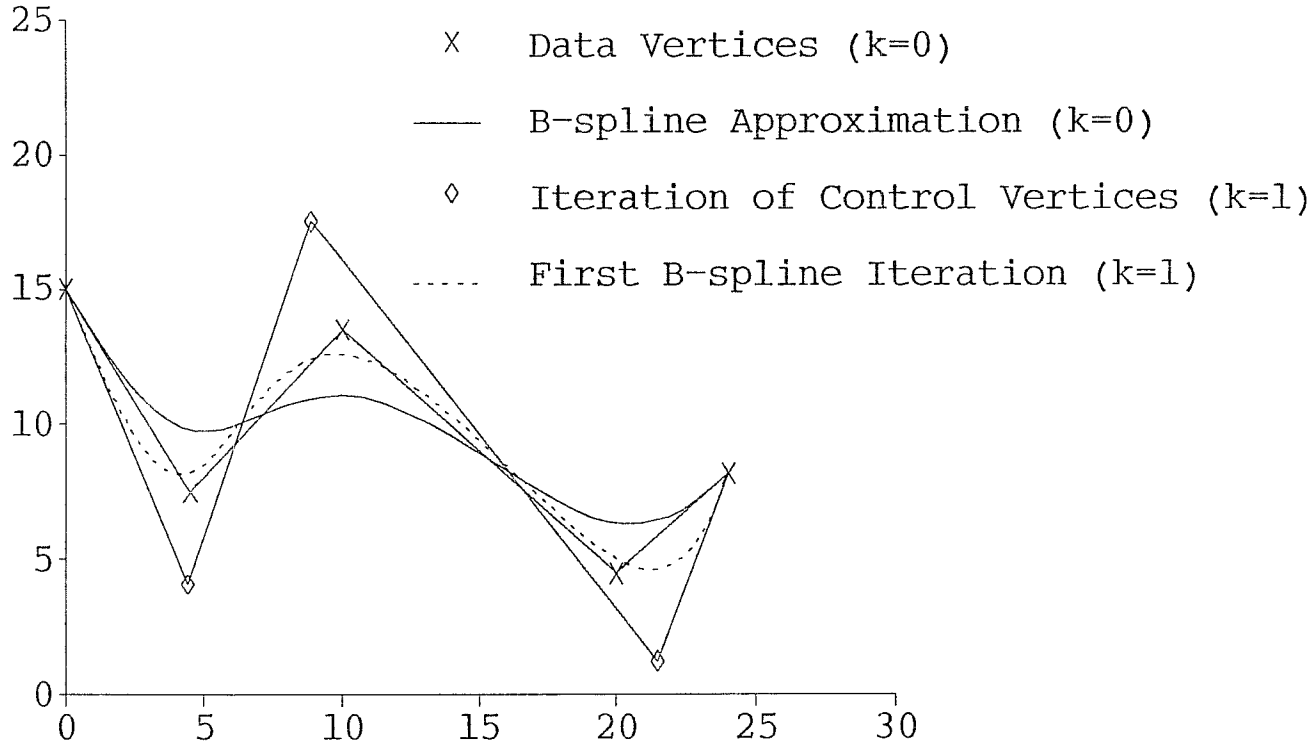
25
20
15
10
5
0
0
5
10
15
20
25
30
Data Vertices (k=0)
B-spline Approximation (k=0)
Iteration of Control Vertices (k=1)
First B-spline Iteration (k=1)

Figure 2

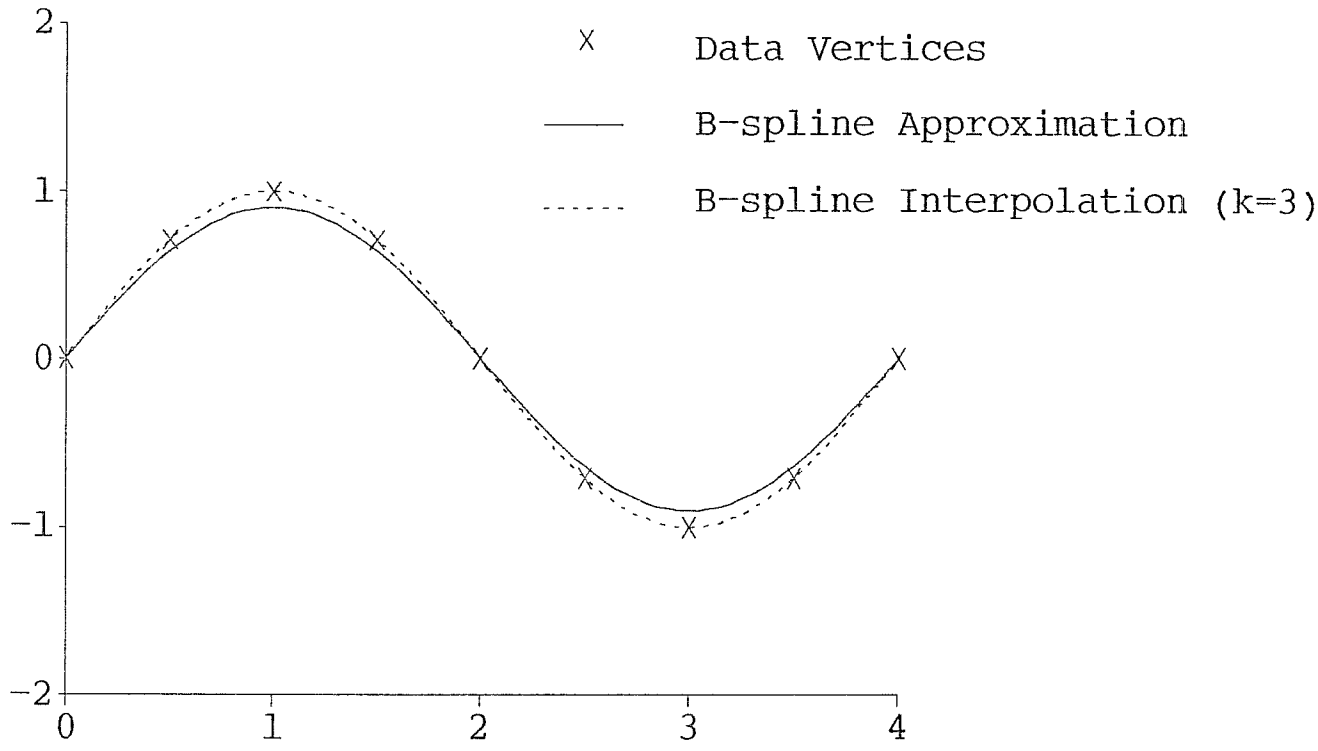
Data Vertices
B-spline Approximation
B-spline Interpolation (k=3)
2
1
0
-1
-2
0
1
2
3
4

Figure 3

Fig 6