

A Parallel B-Spline Surface Fitting Algorithm

FUHUA CHENG and ARDESHIR GOSHTASBY
University of Kentucky

A parallel fitting algorithm using uniform bicubic B-spline surfaces is presented. This algorithm is based on the observation that a tensor product spline surface fitting problem can be split into two spline curve fitting problems, and each of these problems can be carried out in parallel by cyclic reduction. Using this approach, the control points of a uniform bicubic B-spline surface that interpolates a grid of $m \times n$ points can be found in $O(\log m + \log n)$ time on mn processors. Furthermore, since smaller systems of equations are solved in the algorithm, the accumulated error resulting from this approach is smaller than that of the traditional algorithms.

Categories and Subject Descriptors: G.1.1 [Numerical Analysis]: Interpolation—*spline and piecewise polynomial interpolation*; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—*geometric algorithms, languages and systems*

General Terms: Algorithms

Additional Key Words and Phrases: B-splines, cyclic reduction, interpolation, parallel, recursive doubling, uniform cubic B-spline

1. INTRODUCTION

The need for surface fitting arises in many problems in graphics, image processing, pattern recognition, and computer-aided design [1, 3, 6, 8–10]. The task is to model an object by constructing a surface that fits a set of given points. A fitting algorithm, to be of practical value, should meet two criteria: First, it should be efficient enough to be incorporated into an interactive program, and second, it should be based on surface representations with local property [6] so that a local shape modification of the surface would affect only its vicinity.

The B-spline surface representation has proved to be a promising candidate for the second requirement because adjusting a control point affects the shape of only a small area of the surface and therefore provides local control over the shape of the surface.

As far as the efficiency of B-spline surface fitting is concerned, Barsky and Greenberg [3] presented a sequential algorithm that requires $O(mn)$ time for a uniform bicubic B-spline surface to fit a grid of $m \times n$ points. Basically, Gaussian elimination is used to solve a block tridiagonal linear system. An iterative algorithm based on the line SOR (SLOR) method has also been devised [4]. In this method, to take advantage of a parallel computing environment, relaxation

Authors' address: Department of Computer Science, University of Kentucky, Lexington, KY 40506-0027.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1989 ACM 0730-0301/89/0100-0041 \$01.50

ACM Transactions on Graphics, Vol. 8, No. 1, January 1989, Pages 41–50.

of each line in the SLOR method is carried out in parallel by using cyclic reduction. This algorithm performs better than Barsky and Greenberg's algorithm [3] when the number of fitting points is large; it finds the control points of a uniform bicubic B-spline surface that interpolates a grid of $m \times n$ points in $O(m \log n)$ time when m and n are large.

In this paper a new parallel algorithm based on a curve interpolation technique and cyclic reduction is presented. Specifically, we show that, for a given grid of points, the process of finding the control points of a uniform bicubic B-spline surface that interpolates the given points can be split into two steps. Each step is a curve fitting problem and can be carried out in parallel by cyclic reduction. If the dimension of the given grid is $m \times n$, then the algorithm requires only $O(\log m + \log n)$ time on mn processors. Moreover, since the systems of equations solved in each step are of order n or m only, the accumulated error is smaller compared to solving a block tridiagonal linear system.

2. BACKGROUND AND PROBLEM FORMULATION

A tensor product uniform bicubic B-spline surface, $S(x, y)$, with integer knots $1, 2, \dots, m$ and $1, \dots, n$, is a piecewise surface composed of $(m - 1) \times (n - 1)$ patches $S_{rs}(u, v)$, $r = 1, 2, \dots, m - 1$, $s = 1, 2, \dots, n - 1$. Each patch is a bicubic polynomial defined by

$$S_{rs}(u, v) = \frac{1}{36}UMQ_{rs}M^tV^t, \quad 0 \leq u, v \leq 1,$$

where $U = [u^3, u^2, u, 1]$, $V = [v^3, v^2, v, 1]$,

$$M = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix},$$

and

$$Q_{rs} = \begin{bmatrix} C_{r-1,s-1} & C_{r-1,s} & C_{r-1,s+1} & C_{r-1,s+2} \\ C_{r,s-1} & C_{r,s} & C_{r,s+1} & C_{r,s+2} \\ C_{r+1,s-1} & C_{r+1,s} & C_{r+1,s+1} & C_{r+1,s+2} \\ C_{r+2,s-1} & C_{r+2,s} & C_{r+2,s+1} & C_{r+2,s+2} \end{bmatrix},$$

and $C_{i,j}$, $i = 0, 1, \dots, m + 1$, $j = 0, 1, \dots, n + 1$, are the control points of the surface (see Figure 1). The value of $S(x, y)$ satisfies the relation

$$S(x, y) = S_{rs}(x - r, y - s), \quad (x, y) \in [1, m] \times [1, n],$$

where $r = \lfloor x \rfloor$ and $s = \lfloor y \rfloor$.

Given a grid of $m \times n$ points P_{rs} , $r = 1, \dots, m$, $s = 1, \dots, n$, the easiest way to construct a uniform bicubic B-spline surface that interpolates all P_{rs} is to let the domain of the desired uniform bicubic B-spline surface be $[1, m] \times [1, n]$ and

find $C_{i,j}$, $i = 0, 1, \dots, m+1$, $j = 0, 1, \dots, n+1$, so that

$$\begin{aligned} S(r, s) &= \frac{1}{36}(C_{r-1,s-1} + 4C_{r,s-1} + C_{r+1,s-1} + 4C_{r-1,s} + 16C_{r,s} + 4C_{r+1,s} \\ &\quad + C_{r-1,s+1} + 4C_{r,s+1} + C_{r+1,s+1}) \\ &= P_{rs}, \end{aligned} \quad (2.1)$$

for $r = 1, 2, \dots, m$ and $s = 1, 2, \dots, n$.

$2(m+n+2)$ additional conditions are needed to solve (2.1) uniquely. These additional conditions can be given in several different ways [2], for instance, using double-boundary control vertices, triple-boundary control vertices, or phantom boundary control vertices. They are all based on the assumption that the curvature of the spline surface at each point of the boundary is zero—an extension of the fact that the physical spline is straight outside the interval where curve design is performed. By adopting the concept of double-boundary control vertices,¹ namely,

$$\begin{aligned} C_{0,j} &= C_{1,j}; & C_{m+1,j} &= C_{m,j}, & j &= 1, 2, \dots, n \\ C_{i,0} &= C_{i,1}; & C_{i,n+1} &= C_{i,n}, & i &= 0, 1, \dots, m+1, \end{aligned} \quad (2.2)$$

(2.1) combined with (2.2) can be expressed as

$$AC^t = 36P^t, \quad (2.3)$$

where $C = [C_{i,j}]_{1 \times mn}$, $P = [P_{i,j}]_{1 \times mn}$, and A is a block tridiagonal matrix of the following form:

$$A = \begin{bmatrix} 5B & B & 0 & \cdot & \cdot & \cdot & \cdot & 0 \\ B & 4B & B & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & B & 4B & B & 0 & \cdot & \cdot & 0 \\ \cdot & 0 & \cdot & \cdot & \cdot & 0 & \cdot & 0 \\ \cdot & \cdot & 0 & \cdot & \cdot & \cdot & 0 & 0 \\ \cdot & \cdot & \cdot & 0 & B & 4B & B & 0 \\ \cdot & \cdot & \cdot & \cdot & 0 & B & 4B & B \\ 0 & 0 & 0 & 0 & 0 & 0 & B & 5B \end{bmatrix}_{mn \times mn} \quad (2.4)$$

$$B = \begin{bmatrix} 5 & 1 & 0 & \cdot & \cdot & \cdot & \cdot & 0 \\ 1 & 4 & 1 & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & 1 & 4 & 1 & 0 & \cdot & \cdot & 0 \\ \cdot & 0 & \cdot & \cdot & \cdot & 0 & \cdot & 0 \\ \cdot & \cdot & 0 & \cdot & \cdot & \cdot & 0 & 0 \\ \cdot & \cdot & \cdot & 0 & 1 & 4 & 1 & 0 \\ \cdot & \cdot & \cdot & \cdot & 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 5 \end{bmatrix}_{n \times n} \quad (2.5)$$

¹ The technique developed hereafter works for the other two boundary conditions as well.

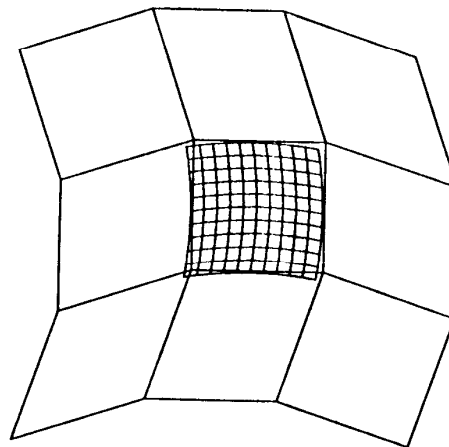


Fig. 1. A uniform bicubic B-spline surface patch and its control points.

Barsky and Greenberg [3] solved (2.3) directly using a modified form of Gaussian elimination. First, the coefficient matrix A in (2.3) is reduced to a unit upper triangular form in two passes, and then back substitution is used to compute the solution. Pass 1 is performed at the block level, and pass 2 at the scalar level. The complexity of this approach is $O(mn)$ for a grid of $m \times n$ given points. Note that pass 1 and pass 2 could actually be implemented in parallel. In that case, the numbers of operations required for pass 1 and pass 2 are $O(m)$ and $O(n)$, respectively. However, since back substitution is performed in strictly sequential order, the complexity of the entire process is again $O(mn)$.

Cheng and Goshtasby [4] solved (2.3) using an iterative algorithm based on the SLOR method, with each line relaxed in parallel by cyclic reduction. When m and n are small (≤ 20), Barsky and Greenberg's algorithm performs better than this method because of the overhead this algorithm incurs and the relatively large number of iterations required compared with the size of m and n . When m and n are large (≥ 20), this algorithm outperforms Barsky and Greenberg's algorithm in both accuracy and computational requirements, since the number of iterations required is bounded above for a fixed sampling space.

3. A PARALLEL ALGORITHM

In this section we show how a parallel algorithm could be designed to make the computation of the control points more efficient.

First, note that, according to a general theorem (Theorem XVII.1) given by de Boor [5], a tensor product spline surface fitting problem on an $m \times n$ grid can actually be split into two spline curve fitting problems involving m and n components, respectively, and each of these problems can be solved in parallel. This can be seen as follows: Observe that, from (2.1) and (2.2), the system of equations shown in (2.3) can be expressed as

$$B_m C_{m \times n} B_n = 36 P_{m \times n}, \quad (3.1)$$

where

$$C_{m \times n} = [C_{i,j}]_{m \times n}, \quad P_{m \times n} = [P_{i,j}]_{m \times n},$$

and B_j is a $j \times j$ matrix defined in (2.5). If we apply B_n^{-1} to both sides of (3.1), we get

$$B_m C_{m \times n} = 6Q_{m \times n}, \quad (3.2)$$

where

$$Q_{m \times n} B_n = 6P_{m \times n}.$$

Note that, since B defined in (2.5) is symmetric and positive definite, its inverse always exists. By taking the transpose of the above equation, we get

$$B_n^t Q_{m \times n}^t = 6P_{m \times n}^t$$

or, equivalently,

$$B_n Q_{m \times n}^t = 6P_{m \times n}^t. \quad (3.3)$$

Therefore, solving eq. (3.1) for $C_{m \times n}$ then becomes a two-step process, namely, solving (3.3) for $Q_{m \times n}$ first, and then solving (3.2) for $C_{m \times n}$.

This is an important observation because (3.3) and (3.2) can be expressed as m and n systems of equations,

$$B_n \begin{bmatrix} Q_{i,1} \\ Q_{i,2} \\ \vdots \\ Q_{i,n} \end{bmatrix} = 6 \begin{bmatrix} P_{i,1} \\ P_{i,2} \\ \vdots \\ P_{i,n} \end{bmatrix}, \quad i = 1, 2, \dots, m \quad (3.4)$$

$$B_m \begin{bmatrix} C_{1,i} \\ C_{2,i} \\ \vdots \\ C_{m,i} \end{bmatrix} = 6 \begin{bmatrix} Q_{1,i} \\ Q_{2,i} \\ \vdots \\ Q_{m,i} \end{bmatrix}, \quad i = 1, 2, \dots, n, \quad (3.5)$$

respectively, and each of these equations can be solved independently; solving (3.3) for Q and solving (3.2) for C can be carried out in parallel by solving the m systems in (3.4) and solving the n systems in (3.5) simultaneously. Furthermore, since each of the equations in (3.4) and (3.5) is of order either m or n , the accumulated error for this approach is expected to be smaller than that of solving (2.3) directly.

For a given grid of $m \times n$ points $P_{i,j}$, $i = 1, 2, \dots, m$, $j = 1, 2, \dots, n$ (Figure 2), solving (3.4) corresponds to the process of finding the control points $Q_{i,j}$, $j = 0, 1, \dots, n + 1$ of the uniform cubic B-spline curve that interpolates the n points $P_{i,j}$, $j = 1, 2, \dots, n$ for each $i = 1, 2, \dots, m$ (Figure 3).

Fig. 2. A grid of 6×5 given points $P_{i,j}$, $1 \leq i \leq 6, 1 \leq j \leq 5$.

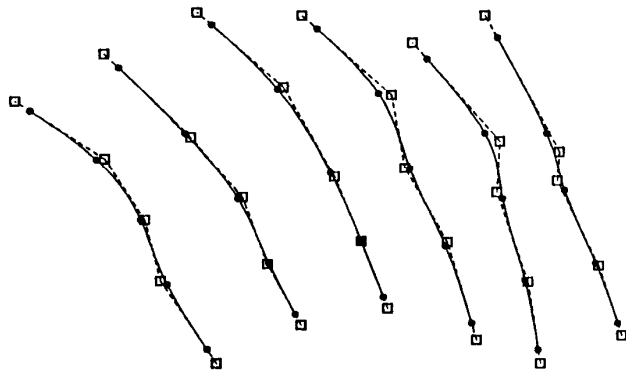
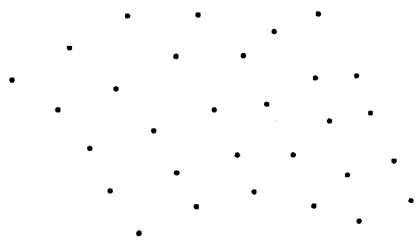


Fig. 3. Control points $Q_{i,j}$, $0 \leq j \leq 6$, of the cubic B-spline curve that interpolates $P_{i,j}$, $1 \leq j \leq 5$, for each $1 \leq i \leq 6$.

Double-boundary control points are used here; that is,

$$\begin{cases} Q_{i,0} = Q_{i,1} \\ Q_{i,n+1} = Q_{i,n} \end{cases} \quad i = 1, 2, \dots, m.$$

Solving (3.5) corresponds to the process of computing the control points $C_{i,j}$, $i = 0, 1, \dots, m + 1$ of the uniform cubic B-spline curve that interpolates the m points $Q_{i,j}$, $i = 1, 2, \dots, m$ for each $j = 0, 1, \dots, n + 1$ (Figure 4). Double-boundary control points are again used here; that is,

$$\begin{cases} C_{0,j} = C_{1,j} \\ C_{m+1,j} = C_{m,j} \end{cases} \quad j = 0, 1, \dots, n + 1.$$

The control points of the uniform bicubic B-spline surface that interpolates the $m \times n$ points $P_{i,j}$, $i = 1, 2, \dots, m, j = 1, 2, \dots, n$ with double-boundary control points are simply $C_{i,j}$, $i = 0, 1, \dots, m + 1, j = 0, 1, \dots, n + 1$ (Figure 5).

Moreover, each of the equations in (3.4) and (3.5) can also be solved in parallel. Observe that for a system of the form

$$BX^t = K^t, \tag{3.6}$$

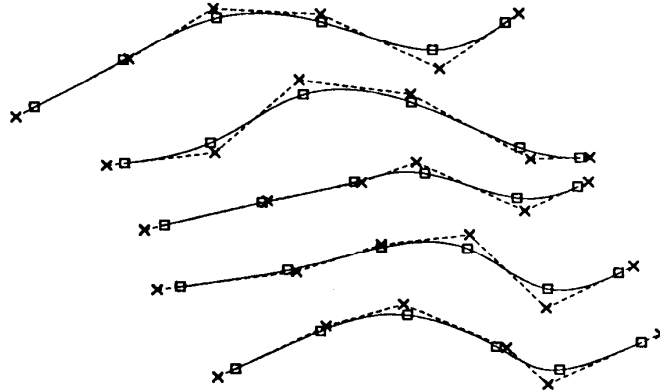


Fig. 4. Control points $C_{i,j}$, $0 \leq i \leq 7$, of the cubic B-spline curve that interpolates $Q_{i,j}$, $1 \leq i \leq 6$, for each $1 \leq j \leq 5$.

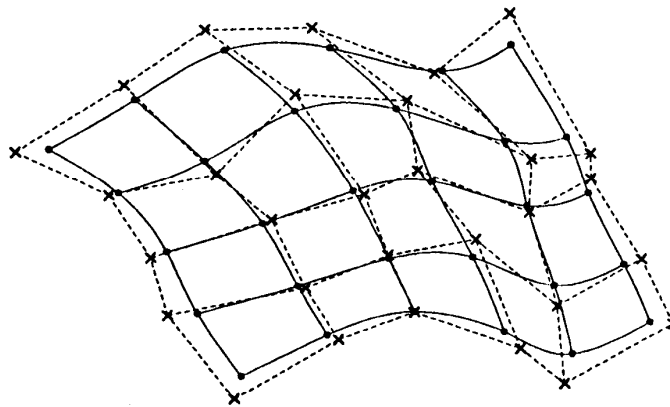


Fig. 5. $C_{i,j}$, $0 \leq i \leq 7$, $0 \leq j \leq 6$, are the control points of the uniform bicubic B-spline surface that interpolates the given points.

where $X = [x_1, x_2, \dots, x_n]$, $K = [k_1, k_2, \dots, k_n]$, and B is an $n \times n$ matrix defined in (2.5), a special form of Gaussian elimination can be given as follows:

(1) *Forward elimination*

$$\begin{cases} w_1 = \frac{1}{5} \\ w_j = \frac{1}{(4 - w_{j-1})}, & j = 2, 3, \dots, n-1, \\ w_n = \frac{1}{(5 - w_{n-1})} \end{cases} \quad (3.7)$$

and

$$\begin{cases} g_1 = k_1/5 = k_1 w_1 \\ g_j = (k_j - g_{j-1}) * w_j, \quad j = 2, 3, \dots, n. \end{cases} \quad (3.8)$$

(2) *Back substitution*

$$\begin{cases} x_n = g_n \\ x_j = g_j - w_j x_{j+1}, \quad j = n-1, n-2, \dots, 2, 1. \end{cases} \quad (3.9)$$

Each of these can be carried out in linear time. Therefore, by solving (3.4) and then (3.5) in parallel, using this approach, we could find the control points of the interpolating spline surface in $O(m+n)$ time. Note that, since the auxiliary vector $W = [w_1, w_2, \dots, w_k]$ ($k = m$ for (3.4) and $k = n$ for (3.5)) is the same for each equation; there is no need to calculate this vector $m+n$ times. We may precalculate and store W to reduce the number of operations. Only (3.8) and (3.9) are needed to obtain the solution for each equation.

Nevertheless, since the recurrences for g_j in (3.8) and x_j in (3.9) are both linear and first order, cyclic reduction [7, 11] can be applied to further reduce the computational time. Note that by defining

$$d_j = \begin{cases} w_j k_j, & j = 1, 2, \dots, n \\ 0, & j \leq 0 \end{cases}$$

and

$$a_j = \begin{cases} -w_j, & j = 2, 3, \dots, n \\ 0, & j \leq 1, \end{cases}$$

equations in (3.8) can be written in a linear, first-order recurrence relation as follows:

$$g_j = a_j g_{j-1} + d_j, \quad j = 1, \dots, n. \quad (3.10)$$

For each $l = 1, 2, \dots, \lceil \log n \rceil$, if we compute

$$\begin{cases} a_j^{(l)} = a_j^{(l-1)} a_{j-2^{l-1}}^{(l-1)}, \\ d_j^{(l)} = a_j^{(l-1)} d_{j-2^{l-1}}^{(l-1)} + d_j^{(l-1)}, \end{cases} \quad j = 1, 2, \dots, n \quad (3.11)$$

with

$$a_j^{(0)} = a_j, \quad d_j^{(0)} = d_j, \quad j = 1, 2, \dots, n$$

(a_j and d_j are set to zero if $j < 1$ or $j > n$), then the value of g_j , $j = 1, 2, \dots, n$, is simply $d_j^{(t)}$, where $t = \lceil \log n \rceil$. The computation of each $a_j^{(l)}$ and $d_j^{(l)}$ is independent of the other $a_i^{(l)}$ and $d_i^{(l)}$ ($i \neq j$), respectively, and (3.11) can be carried out in parallel with respect to j for each $l = 1, 2, \dots, \lceil \log n \rceil$. The evaluation of x_j by cyclic reduction is similar to that of g_j . In this case, however, the evaluation is done forward instead of backward.

By solving each of the systems of equations in (3.4) and (3.5) using the approach shown above, we only have to use $O(\log n)$ and $O(\log m)$ time to evaluate $Q_{i,j}$ s and $C_{i,j}$ s, respectively, on mn processors. It should be pointed out that w_j s can

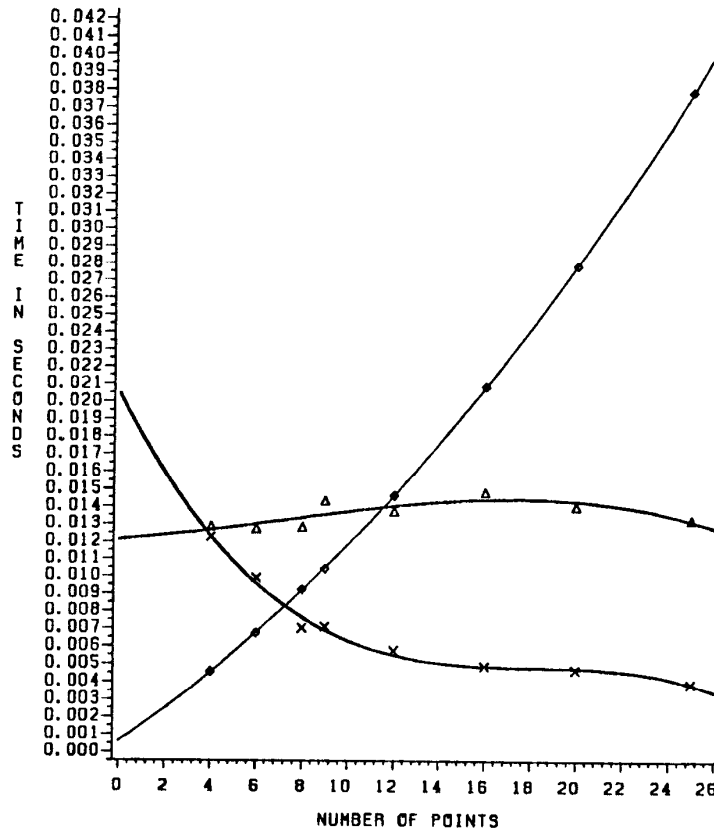


Fig. 6. Performance comparison of algorithms PBSF, BG, and SS.

also be computed in parallel using either recursive doubling or cyclic reduction [7]. However, since w_j s have to be computed only once, it would not be worth doing so.

4. CONCLUSIONS

Our algorithm (PBSF) has been implemented in Pascal on a Sequent Balance 21000 computer with 26 processors. Barsky and Greenberg's algorithm (BG) and the algorithm (SS) that solves eq. (3.1) by solving (3.4) and (3.5) in parallel, but using (3.8) and (3.9) only, have been implemented as well. The time complexity of algorithm SS is $O(m + n)$. The performance of these algorithms is shown in Figure 6. The data collected for algorithms BG, SS, and PBSF are shown in squares, triangles, and crosses, respectively. The curves are generated using least-squares approximation. For a grid of $m \times n$ points, algorithm BG is implemented using one processor only; algorithm SS is implemented using m processors to solve (3.4) in parallel first, and then using n processors to solve (3.5) in parallel; algorithm PBSF is implemented using mn processors. According to the data we

have collected, our algorithm appears well suited to the fitting process when the number of fitting points is greater than or equal to eight.

REFERENCES

1. BARNHILL, R. E., AND RIESENFELD, R. F. *Computer Aided Geometric Design*, R. E. Barnhill and R. F. Riesenfeld, Eds. Academic Press, New York, 1974.
2. BARSKY, B. A. End conditions and boundary conditions for uniform B-spline curve and surface representations. *Comput. Ind.* 3, 1-2 (March & June 1982), 17-29.
3. BARSKY, B. A., AND GREENBERG, D. P. Determining a set of B-spline control vertices to generate an interpolating surface. *Comput. Graph. Image Process.* 14, 3 (November 1980), 203-226.
4. CHENG, F., AND GOSHTASBY, A. B-spline surface interpolation using SLOR method with parallel relaxation. Tech. Rep. 96-87, Dept. of Computer Science, Univ. of Kentucky, Lexington, 1987.
5. DE BOOR, C. In *A Practical Guide to Splines*. Springer-Verlag, New York, 1978.
6. FAUX, I. D., AND PRATT, M. J. *Computational Geometry for Design and Manufacture*. Wiley, New York, 1979.
7. HOCKNEY, R. W., AND JOSSHOPE, C. R. *Parallel Computers: Architecture, Programming and Algorithms*. Adam Hilger, Bristol, U.K., 1981.
8. KNUTH, D. E. In *TEX and METAFONT: New Directions in Typesetting*. Digital Press and American Mathematical Society, Bedford, Mass., 1979.
9. MORTENSON, M. E. *Geometric Modeling*. Wiley, New York, 1985.
10. PAVLIDIS, T. Curve fitting as a pattern recognition problem. In *Proceedings of the 6th International Conference on Pattern Recognition* (Munich, Ger., Oct). IEEE Computer Society Press, Silver Spring, Md., 1982, pp. 853-859.
11. SWEET, R. A. A generalized cyclic reduction algorithm. *SIAM J. Numer. Anal.* 11 (1974), 506-520.

Received July 1987; revised March 1988; accepted March 1988