JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING **35,** 205–210 (1996)
ARTICLE NO. 0082

Parallel B-Spline Surface Fitting on Mesh-Connected Computers

KUO-LIANG CHUNG*,[1] AND WEN-MING YAN†,[2]

*Department of Information Management, National Taiwan Institute of Technology, No. 43, Section 4, Keelung Road, Taipei, Taiwan 10672, Republic of China; and †Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan 10764, Republic of China

The solution of uniform bicubic B-spline curve/surface fitting problem is considered. Based on the matrix perturbation method, this paper first presents a novel approximate $O(n/p)$-time parallel B-spline curve fitting algorithm for finding the corresponding n control points that interpolate those n data points on a linear array processor with p processors, where $p \le n$. Given $m \times n$ data points, we then present an $O(mn/(p_1p_2))$-time parallel algorithm for solving the uniform bicubic B-spline surface fitting problem on a $p_1 \times p_2$ mesh-connected computer, where $p_1 \le m$ and $p_2 \le n$. The relative error analyses of our two stable and cost-optimal parallel solvers are also given. When setting $p_1 = m$ and $p_2 = n$, a constant-time parallel solver for B-spline surface fitting can be derived; this time- and cost-optimal result is a direct method, in contrast to the parallel iterative method of Cheng *et al.* (Parallel B-spline surface interpolation on a mesh-connected processor array, *J. Parallel Distrib. Comput.* 24, 2 (1995), 224–229). © 1996 Academic Press, Inc.

1. INTRODUCTION

Surface fitting is very important in the fields of CAD, CAM, robotic path planning, graphics, pattern recognition, and image processing. Due to the local change property, B-spline surface interpolation is a good fitting tool to construct a smooth surface that fits those given points in the space [15, 2]. From the viewpoint of filters, finding the control points can be solved by using an inverse filtering operation [17].

Suppose we are given a set of $m \times n$ data points to be interpolated. Based on the Gauss–Seidel iteration method, Ajjanagadde and Patnaik [1] presented the first parallel algorithm to perform uniform bicubic B-spline surface fitting in $O(n + p)$ time using $O(pn)$ processors, where p is the number of iterations specified by the user. Then Cheng and Goshtasby [4] presented another parallel iterative solver based on the SLOR method to find the control points in $O(m \log n)$ time using $O(n)$ processors. Based on the cyclic reduction method [18], Cheng and Goshtasby [5] presented the first logarithmic-time parallel algorithm, which takes $O(\log m + \log n)$ time using $O(mn)$ processors. Later, some cost-optimal parallel solvers were presented, where cost is defined to be the product of time and the number of processors used [19]. Using a newly proposed computational model called the mesh-of-unshuffle network, Chung and Lin [7] presented an $O(\log m \log n)$-time parallel algorithm using $O(mn/(\log m \log n))$ processors. On a hypercube network, a similar result was presented by Lin and Chung [20]. Using the pipelining strategy [11], an $O(\log n')$-time parallel algorithm was presented on the same network with $O(mn/\log n')$ processors [13], where $n' = \max(m, n)$. Cheng *et al.* [6] presented the first constant-time parallel algorithm on a mesh-connected computer with $O(mn)$ processors based on the Chebyshev iterative method; they also gave the corresponding relative error analysis. Note that their algorithm [6] is time- and cost-optimal.

Based on the matrix perturbation method, this paper first presents a novel approximate $O(n/p)$-time parallel B-spline curve fitting algorithm for finding the corresponding n control points that interpolate those n data points on a linear array processor with p processors, where $p \le n$. Given $m \times n$ data points, we then present an $O(mn/(p_1p_2))$-time parallel algorithm for B-spline surface fitting on a $p_1 \times p_2$ mesh-connected computer, where $p_1 \le m$ and $p_2 \le n$. The relative error analyses of our two stable and cost-optimal parallel solvers are also given. When setting $p_1 = m$ and $p_2 = n$, a time- and cost-optimal parallel solver can be derived; our result is a direct method vs. the result of Cheng *et al.* [6].

2. PARALLEL B-SPLINE CURVE INTERPOLATOR

Suppose we are given a set of 3-dimensional data points, $B_i = (b_i^{(1)}, b_i^{(2)}, b_i^{(3)})$ for $1 \le i \le n$. According to [2, 15], for uniform B-spline curve, each data point can be expressed by a weighted average of three control points: $B_i = \frac{1}{6}(C_{i-1} + 4C_i + C_{i+1})$, $1 \le i \le n$, where $C_i = (c_i^{(1)}, c_i^{(2)}, c_i^{(3)})$. They form a system of n equations in $n + 2$ unknowns for all given points. In order to completely solve the system, we need the following two additional equations to specify how the boundary control points are interpolated: $C_0 = C_1$; $C_{n+1} = C_n$. For simplicity, we only consider $\mathbf{b} = (b_1, b_2, ..., b_n)^T = (b_1^{(1)}, b_2^{(1)}, ..., b_n^{(1)})^T$ and $\mathbf{c} = (c_1, c_2, ..., c_n)^T = (c_1^{(1)}, c_2^{(1)}, ..., c_n^{(1)})^T$ throughout this

[1] E-mail: klchung@cs.ntit.edu.tw.
[2] E-mail: ganboon@csie.ntu.edu.tw.

0743-7315/96 $18.00

section. Thus, the above system of linear equations can be equivalently transformed into

$$A\mathbf{c} = 6\mathbf{b}, \tag{1}$$

where

$$A = \begin{pmatrix} 5 & 1 & & & & & \\ 1 & 4 & 1 & & & & \\ & \cdot & \cdot & \cdot & & & \\ & & & 1 & 4 & 1 & \\ & & & & 1 & 5 & \end{pmatrix}.$$

Our parallel solver consists of three phases called the Toeplitz factorization, the substitution procedure, and the update procedure, respectively. Suppose we have a linear array processor with p processors. Figure 1 shows a linear array processor with four processors, where the processor identity inside the square box denotes the address of the processor. For convenience, suppose n is a multiple of the number of processors used, i.e., $n = pq$. Throughout the remainder of this section, matrices are represented by uppercase letters, vectors by bold lowercase letters, and scalars by plain lowercase letters. The superscript T corresponds to the transpose operation. We now show how each phase can be accomplished on the linear array network in a highly parallel way.

2.1. Phase 1: Toeplitz Factorization

Let

$$A'_q = \begin{pmatrix} a & 1 & & & & & \\ 1 & 4 & 1 & & & & \\ & \cdot & \cdot & \cdot & & & \\ & & & 1 & 4 & 1 & \\ & & & & 1 & 4 & \end{pmatrix}_{q\times q} = L'_q U'_q, \tag{2}$$

$$L'_q = \begin{pmatrix} 1 & & & & & & \\ -b & 1 & & & & & \\ & \cdot & \cdot & & & & \\ & & & -b & 1 & & \\ & & & & -b & 1 & \end{pmatrix}_{q\times q}, \quad \text{and}$$

$$U'_q = \begin{pmatrix} a & 1 & & & & & \\ & a & 1 & & & & \\ & & \cdot & \cdot & & & \\ & & & & a & 1 & \\ & & & & & a & \end{pmatrix}_{q\times q}, \tag{3}$$

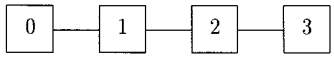

FIG. 1. A linear array processor with four processors.

which implies that $a - b = 4$ and $-ab = 1$; this then implies that $b = -1/a$ and $a = 2 \pm \sqrt{3}$. Since we wish the matrices L' and U' to be diagonally dominant, we will select the sign so that the absolute value of a is greater than 1. Therefore, let $a = 2 + \sqrt{3}$ and $b = -1/a = \sqrt{3} - 2$.

The computation of a and b provides the Toeplitz factorization of matrix A'_q (phase 1 in our algorithm), which can be computed in parallel by each processor in $O(1)$ time on the array processor.

2.2. Phase 2: Substitution Procedure

Initially, we partition $\mathbf{b}$ into p parts and $\mathbf{b} = (\mathbf{b}^{0^{\mathrm{T}}}, \mathbf{b}^{1^{\mathrm{T}}}, \ldots, \mathbf{b}^{p-1^{\mathrm{T}}})^{\mathrm{T}}$, where $\mathbf{b}^i = (6b_{iq+1}, 6b_{iq+2}, \ldots, 6b_{iq+q})^{\mathrm{T}}$ for $0 \le i \le p-1$. The ith processor first solves $A'_q\mathbf{z}^i = \mathbf{b}^i$, $0 \le i \le p-1$, where $\mathbf{z}^i = (z_{iq+1}, z_{iq+2}, \ldots, z_{iq+q})^{\mathrm{T}}$, that is, the ith processor solves $L'_q U'_q \mathbf{z}^i = \mathbf{b}^i$. It can be verified that each processor in the linear array processor needs about $5q$ time to perform the forward and backward substitution procedure for solving $L'_q U'_q \mathbf{z}^i = \mathbf{b}^i$ (second phase in our algorithm) sequentially [16].

Consequently, it follows that

$$A_n\mathbf{z} - \mathbf{b} = g_p\mathbf{e}_n + h_p\mathbf{e}_1 + \sum_{i=1}^{p-1}(g_i\mathbf{e}_{iq} + h_i\mathbf{e}_{iq+1}), \tag{4}$$

where

$$g_i = z_{iq+1}, h_i = z_{iq} + (2-\sqrt{3})z_{iq+1}, 1 \le i \le p-1, \tag{5}$$

$$g_p = z_n, \quad h_p = (3-\sqrt{3})z_1. \tag{6}$$

By (4), the solution of $\mathbf{c}$ in (1) will be determined approximately, say, $\mathbf{z} - \mathbf{p}$, in Section 2.3 later, where $A\mathbf{p} \approx g_p\mathbf{e}_n + h_p\mathbf{e}_1 + \sum_{i=1}^{p-1}(g_i\mathbf{e}_{iq} + h_i\mathbf{e}_{iq+1})$. To estimate the bound of the relative error, we need the following two auxiliary results. Here $\| \bullet \|$ denotes the infinity norm of a vector.

LEMMA 1 [13]. *If $A\mathbf{x} = \mathbf{b}$, where $A = [a_{ij}]_{n\times n}$ is a diagonally dominant matrix, then $\|\mathbf{x}\| \le (1/\min_{1\le i\le n}(|a_{ii}| - \sum_{j=1,j\ne i}^{n}|a_{ij}|))\,\|\mathbf{b}\|$.*

For A' in (2), because $\min_{1\le i\le n}(|a_{ii}| - \sum_{j=1,j\ne i}^{n}|a_{ij}|) = 2$, we have the immediate result.

COROLLARY 2 [13]. *If $A'\mathbf{x}' = \mathbf{b}'$, then $\|\mathbf{x}'\| \le \frac{1}{2}\|\mathbf{b}'\|$.*

From $A'_q\mathbf{z}^i = \mathbf{b}^i$ for $0 \le i \le p-1$ and by Corollary 2,

$$\|\mathbf{z}^i\| \le \tfrac{1}{2}\|\mathbf{b}^i\| \le \tfrac{1}{2}\|\mathbf{b}\|. \tag{7}$$

2.3. Phase 3: Update Procedure

From (4), the solution of $\mathbf{c}$ in (1) will be determined approximately in this section and only local communica-

tions between adjacent processors on the linear array processor will be needed. Then, the bound of the relative error will be analyzed.

From $a - b = 4$ and $-ab = 1$, we have $1 + 4b + b^2 = 0$. Let $\mathbf{p}_k = (\underbrace{0, ..., 0, 1}_{k}, \underbrace{b, ..., b^t, 0, ..., 0}_{n-k})^{\mathrm{T}}$ and $\mathbf{q}_k = (\underbrace{0, ..., 0, b^t, ..., b, 1}_{k}, \underbrace{0, ..., 0}_{n-k})^{\mathrm{T}}$ for $q \le k \le n - q + 1$. Here we assume that $q > t$. In addition, we let $\mathbf{p}_1 = (1, b, ..., b^{t-1}, b^t, 0, ..., 0)^{\mathrm{T}}$ and $\mathbf{q}_n = (0, ..., 0, b^t, b^{t-1}, ..., b, 1)^{\mathrm{T}}$. Accordingly, we have

$$A\mathbf{p}_k = \mathbf{e}_{k-1} + (2 + \sqrt{3})\mathbf{e}_k + b^t(-b\mathbf{e}_{k+t} + \mathbf{e}_{k+t+1}), \quad (8)$$

$$A\mathbf{q}_k = b^t(\mathbf{e}_{k-t-1} - b\mathbf{e}_{k-t}) + (2 + \sqrt{3})\mathbf{e}_k + \mathbf{e}_{k+1}, \quad (9)$$

$$A\mathbf{p}_1 = (3 + \sqrt{3})\mathbf{e}_1 + b^t(-b\mathbf{e}_{t+1} + \mathbf{e}_{t+2}), \quad (10)$$

$$A\mathbf{q}_n = (3 + \sqrt{3})\mathbf{e}_n + b^t(\mathbf{e}_{n-t-1} - b\mathbf{e}_{n-t}). \quad (11)$$

By (8), (9), (10), and (11), for $1 \le i \le p - 1$, we have

$$\begin{aligned} A(u_i\mathbf{p}_{iq+1} + v_i\mathbf{q}_{iq}) &= (u_i + (2 + \sqrt{3})v_i)\mathbf{e}_{iq} \\ &\quad + ((2 + \sqrt{3})u_i + v_i)\mathbf{e}_{iq+1} \\ &\quad + u_ib^t(-b\mathbf{e}_{iq+t+1} + \mathbf{e}_{iq+s+2}) \\ &\quad + v_ib^t(\mathbf{e}_{iq-t-1} - b\mathbf{e}_{iq-t}), \quad (12) \\ A(u_p\mathbf{p}_1 + v_p\mathbf{q}_n) &= (3 + \sqrt{3})v_p\mathbf{e}_n + (3 + \sqrt{3})u_p\mathbf{e}_1 \\ &\quad + u_pb^s(-b\mathbf{e}_{t+1} + \mathbf{e}_{t+2}) \\ &\quad + v_pb^t(\mathbf{e}_{n-t-1} - b\mathbf{e}_{n-t}). \quad (13) \end{aligned}$$

Let $u_i + (2 + \sqrt{3})v_i = g_i$ and $(2 + \sqrt{3})u_i + v_i = h_i$ for $1 \le i \le p - 1$; $(3 + \sqrt{3})v_p = g_p$ and $(3 + \sqrt{3})u_p = h_p$, by (5) and (6),

$$\begin{aligned} u_i &= \frac{g_i - (2 + \sqrt{3})h_i}{-6 - 4\sqrt{3}} = \frac{(2 + \sqrt{3})z_{iq}}{6 + 4\sqrt{3}}, \\ v_i &= \frac{h_i - (2 + \sqrt{3})g_i}{-6 - 4\sqrt{3}} = \frac{2\sqrt{3}z_{iq+1} - z_{iq}}{6 + 4\sqrt{3}}, \\ &\quad 1 \le i \le p - 1, \quad (14) \end{aligned}$$

$$u_p = \frac{h_p}{3 + \sqrt{3}} = \frac{3 - \sqrt{3}}{3 + \sqrt{3}}z_1, v_p = \frac{g_p}{3 + \sqrt{3}} = \frac{z_n}{3 + \sqrt{3}}. \quad (15)$$

Let

$$\mathbf{p} = (u_p\mathbf{p}_1 + v_p\mathbf{q}_n) - \sum_{i=1}^{p-1}(u_i\mathbf{p}_{iq+1} + v_i\mathbf{q}_{iq}), \quad (16)$$

by (12), (13), (14), and (15), we have $A\mathbf{p} \approx g_p\mathbf{e}_n + h_p\mathbf{e}_1 + \sum_{i=1}^{p-1}(g_i\mathbf{e}_{iq} + h_i\mathbf{e}_{iq+1})$. Recall that the solution of $\mathbf{c}$ in (1) is determined approximately by $\mathbf{c} = \mathbf{z} - \mathbf{p}$; then we have the bound of the corresponding relative error.

THEOREM 3 [13]. *Let $\bar{\mathbf{c}} = A^{-1}\mathbf{b}$; i.e., let $\bar{\mathbf{c}}$ be the exact solution of $A\mathbf{x} = \mathbf{b}$; then*

$$\|A_n\mathbf{c} - \mathbf{b}\| \le 0.173|b^t|\,\|\mathbf{b}\| \quad \textit{and} \quad \frac{\|\mathbf{c} - \bar{\mathbf{c}}\|}{\|\bar{\mathbf{c}}\|} \le 0.519|b^t|.$$

The above three-phase parallel algorithm for solving (1) can be written as follows, where the ith, $0 \le i \le p - 1$, processor in the linear array processor returns the partial solution vector $\mathbf{c}^i$.

ALGORITHM 1

FOR $i := 0$ TO $p - 1$ DO IN PARALLEL
(1) Solve $A\mathbf{z}^i = \mathbf{b}^i$;
(2) Evaluate u_i, v_{i+1};
(3) $\mathbf{c}^i \leftarrow \mathbf{z}^i - u_i(1, b, b^2, ..., b^t, 0, ..., 0)^{\mathrm{T}} - v_{i+1}(0, ..., 0, b^t, ..., b^2, b, 1)^{\mathrm{T}}$;
END

Following the above algorithm, the parallel pseudo code performed by processor (node) i, $0 \le i \le p - 1$, is referred to [13]. Since each processor wants to solve a small linear system of size $O(n/p)$ and only local communications between adjacent processors on the linear array processor are needed, we have the following result.

THEOREM 4. *Given n data points, the uniform B-spline curve fitting problem can be solved in $O(n/p)$ time on the linear array processor with $O(p)$ processors; the relative error is $\le 0.519(2 - \sqrt{3})^t$ for $t < n/p$.*

When setting $p = O(n)$, a constant-time parallel algorithm for B-spline curve fitting is obtained. However, it is a trade-off between the time needed and the relative error desired.

We have presented the parallel B-spline curve fitting on the linear array processor. The next section will extend this result to solve the B-spline surface fitting on the mesh-connected computer and derive the corresponding relative error.

3. PARALLEL B-SPLINE SURFACE INTERPOLATOR

Suppose we are given a set of 3-dimensional points, $B_{i,j} = (b_{i,j}^{(1)}, b_{i,j}^{(2)}, b_{i,j}^{(3)})$ for $1 \le i \le m$, $1 \le j \le n$. The uniform bicubic B-spline surface for interpolating these $m \times n$ data points consists of $(m - 1) \times (n - 1)$ patches $S_{i,j}(u, v)$ for $0 \le u, v < 1$, $1 \le i \le m - 1$, and $1 \le j \le n - 1$. Each patch is defined by a bicubic polynomial $s_{i,j}(u, v) = (1/36)[u^3, u^2, u, 1]NQ_{i,j}N^t[v^3, v^2, v, 1]^t$, where

$$N = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix} \quad \text{and}$$

$$Q_{i,j} = \begin{pmatrix} C_{i-1,j-1} & C_{i-1,j} & C_{i-1,j+1} & C_{i-1,j+2} \\ C_{i,j-1} & C_{i,j} & C_{i,j+1} & C_{i,j+2} \\ C_{i+1,j-1} & C_{i+1,j} & C_{i+1,j+1} & C_{i+1,j+2} \\ C_{i+2,j-1} & C_{i+2,j} & C_{i+2,j+1} & C_{i+2,j+2} \end{pmatrix},$$

where $C_{i,j} = (c_{i,j}^{(1)}, c_{i,j}^{(2)}, c_{i,j}^{(3)})$, $0 \le i \le m+1$, $0 \le j \le n+1$, are the control points of the surface to be determined. Our task is to compute these control points.

Each data point can be expressed by a weighted average of nine control points: $B_{i,j} = (1/36)(C_{i-1,j-1} + 4C_{i,j-1} + C_{i+1,j-1} + 4C_{i-1,j} + 16C_{i,j} + 4C_{i+1,j} + C_{i-1,j+1} + 4C_{i,j+1} + C_{i+1,j+1})$ for $1 \le i \le m$, $1 \le j \le n$. They form a system of mn equations in $(m+2)(n+2)$ unknowns. In order to completely solve the system, commonly we need the following $2(m+n+2) = (m+2)(n+2) - mn$ additional equations to specify how the boundary control points are interpolated: $C_{0,j} = C_{1,j}$; $C_{m+1,j} = C_{m,j}$, $1 \le j \le n$; $C_{i,0} = C_{i,1}$; $C_{i,n+1} = C_{i,n}$, $0 \le i \le m+1$. For simplicity, we only consider $B_{i,j} = b_{i,j}^{(1)}$, $1 \le i \le m$ and $1 \le j \le n$. Throughout the remainder of this section, matrices are also represented by uppercase letters, vectors by bold lowercase letters, and scalars by plain lowercase letters. The superscript T corresponds to the transpose operation. That is, we consider the given set of data points

$$\begin{aligned}\mathbf{b}^{\mathrm{T}} &= 36(b_{1,1}^{(1)}, b_{1,2}^{(1)}, \ldots, b_{1,n}^{(1)}, b_{2,1}^{(1)}, b_{2,2}^{(1)}, \ldots, b_{2,n}^{(1)}, \ldots, \\ &\quad b_{m,1}^{(1)}, b_{m,2}^{(1)}, \ldots, b_{m,n}^{(1)}) \\ &= (b_{1,1}, b_{1,2}, \ldots, b_{1,n}, b_{2,1}, b_{2,2}, \ldots, b_{2,n}, \ldots, \\ &\quad b_{m,1}, b_{m,2}, \ldots, b_{m,n})\end{aligned}$$

and the corresponding control points (to be determined)

$$\begin{aligned}\mathbf{c}^{\mathrm{T}} &= (c_{1,1}^{(1)}, c_{1,2}^{(1)}, \ldots, c_{1,n}^{(1)}, c_{2,1}^{(1)}, c_{2,2}^{(1)}, \ldots, c_{2,n}^{(1)}, \ldots, \\ &\quad c_{m,1}^{(1)}, c_{m,2}^{(1)}, \ldots, c_{m,n}^{(1)}) \\ &= (c_{1,1}, c_{1,2}, \ldots, c_{1,n}, c_{2,1}, c_{2,2}, \ldots, c_{2,n}, \ldots, \\ &\quad c_{m,1}, c_{m,2}, \ldots, c_{m,n}).\end{aligned}$$

Employing those additional boundary conditions, our task becomes to solve $B\mathbf{c} = \mathbf{b}$, where B is a block tridiagonal matrix of the following form:

$$B_{mn\times mn} = \begin{pmatrix} 5A_n & A_n & & & & \\ A_n & 4A_n & A_n & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & A_n & 4A_n & A_n \\ & & & & A_n & 5A_n \end{pmatrix}_{m\times m},$$

and A_n has been defined in (1).

Since B can be factorized into $B = CD$, where

$$C = \begin{pmatrix} A_n & & & & \\ & A_n & & & \\ & & \ddots & & \\ & & & A_n & \\ & & & & A_n \end{pmatrix}_{m\times m} \quad \text{and}$$

$$D = \begin{pmatrix} 5I_n & I_n & & & & \\ I_n & 4I_n & I_n & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & I_n & 4I_n & I_n \\ & & & & I_n & 5I_n \end{pmatrix}_{m\times m},$$

the above system, $B\mathbf{c} = \mathbf{b}$, can be transformed into

$$A_{n\times n}\mathbf{h}^{\langle i\rangle} = \mathbf{b}^{\langle i\rangle} \text{ for } 1 \le i \le m, \tag{17}$$

$$A_{m\times m}\mathbf{c}^{[j]} = \mathbf{h}^{[j]} \text{ for } 1 \le j \le n, \tag{18}$$

where

$$\begin{aligned}&\mathbf{h}^{\langle i\rangle} = (h_{i,1}, h_{i,2}, \ldots, h_{i,n})^{\mathrm{T}}, \mathbf{b}^{\langle i\rangle} = (b_{i,1}, b_{i,2}, \ldots, b_{i,n})^{\mathrm{T}} \\ &\qquad \text{for } 1 \le i \le m, \\ &\mathbf{c}^{[j]} = (c_{1,j}, c_{2,j}, \ldots, c_{m,j})^{\mathrm{T}}, \mathbf{h}^{[j]} = (h_{1,j}, h_{2,j}, \ldots, h_{m,j})^{\mathrm{T}} \\ &\qquad \text{for } 1 \le j \le n.\end{aligned}$$

Therefore, B-spline surface interpolation becomes a two-part process, namely, solving the m special tridiagonal systems in (17) for $\mathbf{h}^{\langle i\rangle}$, $1 \le i \le m$, first and then solving the n tridiagonal systems in (18) for $\mathbf{c}^{[j]}$, $1 \le j \le n$. We now present the parallel algorithm for B-spline surface fitting on a mesh-connected computer. The mesh-connected computer with $p_1 \times p_2$ processors is shown in Fig. 2, where all the rows and columns are linear array processors. For simplicity, suppose $m = p_1 q_1$ and $n = p_2 q_2$.

The linear array processor in row j, $0 \le j \le p_1 - 1$, is responsible for solving $A_{n\times n}\mathbf{h}^{\langle k\rangle} = 36\mathbf{b}^{\langle k\rangle}$ for $jq_1 + 1 \le k \le (j+1)q_1$ by applying Algorithm 1 q_1 times. Since each system $A_{n\times n}\mathbf{h}^{\langle i\rangle} = \mathbf{b}^{\langle i\rangle}$ can be solved in $O(n/p_2)$ time, totally it takes $O(mn/(p_1p_2))$ time for solving (17) and it gives $\|A_{n\times n}\mathbf{h}^{\langle i\rangle} - \mathbf{b}^{\langle i\rangle}\| \le 0.173|b^t|\,\|\mathbf{b}^{\langle i\rangle}\|$. Therefore, we have

$$\|C\mathbf{h} - \mathbf{b}\| \le 0.173|b^t| \max_{1\le i\le m} \|\mathbf{b}^{\langle i\rangle}\| = 0.173|b^t|\,\|\mathbf{b}\|.$$

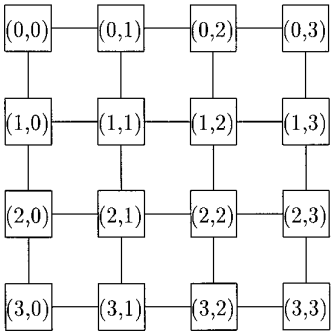

FIG. 2. A mesh-connected computer with 4×4 processors.

In contrast, the linear array processor in column j, $0 \le j \le p_2 - 1$, is responsible for solving $A_{m\times m}\mathbf{c}^{[k]} = \mathbf{h}^{[k]}$ for $jq_2 + 1 \le k \le (j+1)q_2$ by applying Algorithm 1 q_2 times. Each system $A_{m\times m}\mathbf{c}^{[i]} = \mathbf{h}^{[i]}$ can be solved in $O(m/p_1)$ time, so it takes $O(mn/(p_1p_2))$ time in total to solve (18). We have $\|A_{m\times m}\mathbf{c}^{[j]} - \mathbf{h}^{[j]}\| \le 0.173|b^t|\,\|\mathbf{h}^{[j]}\|$. Then we have

$$\|D\mathbf{c} - \mathbf{h}\| \le 0.173|b^t| \max_{1\le j\le n} \|\mathbf{h}^{[j]}\| = 0.173|b^t|\,\|\mathbf{h}\|.$$

Because of $\|C\mathbf{h} - \mathbf{b}\| \le 0.173|b^t|\,\|\mathbf{b}\|$, $\|D\mathbf{c} - \mathbf{h}\| \le 0.173|b^t|\,\|\mathbf{h}\|$, and $\|C\| \le 6$, it follows that

$$\begin{aligned}
\|C\mathbf{h}\| &\le \|C\mathbf{h} - \mathbf{b}\| + \|\mathbf{b}\| \\
&\le (1 + 0.173|b^t|)\|\mathbf{b}\|, \\
\|\mathbf{h}\| &\le \tfrac{1}{2}\|C\mathbf{h}\| \\
&\le \frac{1 + 0.173|b^t|}{2}\|\mathbf{b}\|, \\
\|B\mathbf{c} - \mathbf{b}\| &= \|CD\mathbf{c} - \mathbf{b}\| \\
&\le \|C\|\,\|D\mathbf{c} - \mathbf{h}\| + \|C\mathbf{h} - \mathbf{b}\| \\
&\le 6 \times 0.173|b^t|\,\|\mathbf{h}\| + 0.173|b^t|\,\|\mathbf{b}\| \\
&\le 0.173|b^t|[3(1 + 0.173|b^t|) + 1]\|\mathbf{b}\|.
\end{aligned}$$

By $\bar{\mathbf{c}} = B^{-1}\mathbf{b}$ and Corollary 2, we have

$$\|\mathbf{c} - \bar{\mathbf{c}}\| \le \tfrac{1}{2}\|D\mathbf{c} - D\bar{\mathbf{c}}\| \le \tfrac{1}{4}\|CD\mathbf{c} - CD\bar{\mathbf{c}}\| = \tfrac{1}{4}\|B\mathbf{c} - \mathbf{b}\|$$

and

$$\|\mathbf{b}\| = \|B\bar{\mathbf{c}}\| \le \|B\|_\infty\|\bar{\mathbf{c}}\| = 36\|\bar{\mathbf{c}}\|.$$

Combining the above two inequalities, we have

$$\begin{aligned}
\|\mathbf{c} - \bar{\mathbf{c}}\| &\le \tfrac{1}{4}\|B\mathbf{c} - \mathbf{b}\| \\
&\le \frac{0.173}{4}|b^t|[3(1 + 0.173|b^t|) + 1]\|\mathbf{b}\| \\
&\le 0.173 \times 9|b^t|[3(1 + 0.173|b^t|) + 1]\|\bar{\mathbf{c}}\|,
\end{aligned}$$

that is, $\|\mathbf{c} - \bar{\mathbf{c}}\|/\|\bar{\mathbf{c}}\| \le 1.557|b^t|[3(1 + 0.173|b^t|) + 1]$.

To save space, we omit the parallel pseudo code for B-spline surface fitting. Finally, we have the main result.

THEOREM 5. *Given $m \times n$ data points, the uniform B-spline surface fitting problem can be solved in $O(mn/(p_1p_2))$ time on the mesh-connected computer with $O(p_1p_2)$ processors; the relative error is $\le 1.557|(2 - \sqrt{3})^t|[3(1 + 0.173|(2 - \sqrt{3})^t|) + 1]$ for $t < min(m/p_1, n/p_2)$.*

When setting $p_1 = m$ and $p_2 = n$, a constant-time parallel solver for B-spline surface fitting can be obtained.

4. CONCLUSIONS

The significance of B-spline surface fitting is due to its popular use in the areas of computer graphics, CAD, CAM, image processing, etc. Our main contribution is to present a novel approximate parallel algorithm for solving B-spline interpolation problem and to show that on the mesh-connected computer with $O(p_1p_2)$ processors, our algorithm can be performed in $O(mn/(p_1p_2))$ time. The relative error analyses have also been given.

Using the same matrix perturbation method proposed in this paper and the Sherman–Morrison formula [3], the product-expansion based vectorized algorithms for solving B-spline curve and surface fitting were presented in [8, 9, 10]. The extension to solve the special tridiagonal systems has been developed in [23, 14]. Further, our parallel algorithm can be applied to solve the closed B-spline surface fitting problem on a torus (wraparound mesh) multiprocessor directly. In addition, the results of this paper can also be applied to solve the diagonally dominant block tridiagonal system to achieve better performance.

It is interesting to employ the other parallel tridiagonal solvers [21, 22] to handle the same surface fitting problem.

ACKNOWLEDGMENTS

The authors appreciate the help of the three referees, Dr. F. Cheng, Dr. J. Gustafson, and Dr. S. Sahni which led to an improved version of this paper. This research was supported in part by the National Science Council of the Republic of China under Contract NSC85-2121-M001-002.

REFERENCES

1. Ajjanagadde, V. G., and Patnaik, L. M., Systolic architecture for B-spline surfaces. *Int. J. Parallel Programming* **15,** 6 (1986), 551–565.
2. Bartels, R. H., Beatty, J. C., and Barsky, B. A., "*An Introduction to Splines for Use in Computer Graphics and Geometric Modeling.*" Morgan Kaufmann, San Mateo, CA, 1987.
3. Bartlett, M. S. An inverse matrix adjustment arising in discriminant analysis. *Ann. Math. Statist.* **22** (1951), 107–111.
4. Cheng, F., and Goshtasby, A. B-spline surface interpolation using SLOR method with parallel relaxation. Technical Report 96-87, Department of Computer Science, University of Kentucky, Lexington, 1987.
5. Cheng, F., and Goshtasby, A. A parallel B-spline surface fitting algorithm. *ACM Trans. Graphics* **8,** 1 (1989), 41–50.
6. Cheng, F., Wasilkowski, G. W., Wang, J., Zhang, C., and Wang, W. Parallel B-spline surface interpolation on a mesh-connected processor array. *J. Parallel Distrib. Comput.* **24,** 2 (1995), 224–229.

7. Chung, K. L., and Lin, F. C. A cost-optimal parallel algorithm for B-spline surface fitting. *CVGIP: Graph. Models Image Process.* **53,** 6 (1991), 601–605.
8. Chung, K. L., and Shen, L. Z. Vectorized algorithm for B-spline curve fitting on CRAY X-MP EA/16se. *Proc. of Supercomputing.* IEEE Computer Society, Silver Spring, MD, 1992, pp. 166–169.
9. Chung, K. L., Peng, Y. C., and Yan, W. M. Vectorization of B-spline surface fitting. *1st Workshop on Computer Graphics.* Academia Sinica, Republic of China, 1993, pp. 21–26.
10. Chung, K. L., and Yan, W. M. Computing quadratic B-spline curve fitting on CRAY X-MP. *Proc. National Computer Symposium,* Chiayi, Republic of China, 1993, pp. 401–407.
11. Chung, K. L., and Chang, H. W. Novel pipelining and processor allocation strategy for monoid computations on unshuffle-exchange network. *Parallel Process. Lett.* **3,** 2 (1993), 189–193.
12. Chung, K. L. On parallel algorithm for B-spline surface fitting. Tech. Report, Dept. of Inform. Mgmt., National Taiwan Inst. of Tech., Apr., 1994.
13. Chung, K. L., and Yan, W. M. Parallel B-spline surface fitting on mesh. Tech. Report, Department of Information Management, National Taiwan Institute of Technology, Dec. 1994.
14. Chung, K. L., Yan, W. M., and Wu, J. G. A parallel algorithm for solving special tridiagonal systems on ring networks. *Computing,* in press.
15. de Boor, C. *A Practical Guide to Splines.* Springer-Verlag, New York, 1978.
16. Golub, G. H., and Van Loan, C. F. *Matrix Computations.* 2nd ed., Chap. 4. The Johns Hopkins University Press, Baltimore, 1989.
17. Goshtasby, A., Cheng, F., and Barsky, B. A. B-spline curves and surfaces viewed as digital filters. *Comput. Vision Graphics Image Process.* **52,** 2 (1990), 264–275.
18. Hockney, R. W. A fast direct solution of Poisson's equation using Fourier analysis. *J. Assoc. Comput. Mach.* **12** (1965), 95–113.
19. Leighton, F. T. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* Morgan Kaufmann, San Mateo, CA, 1992.
20. Lin, F. C., and Chung, K. L. Cost-optimal B-spline surface fitting on hypercube. *Inter. Conf. on Parallel Process.* 1990, pp. III–185–192.
21. Sun, X. H., Zhang, H., and Ni, L. Efficient tridiagonal solvers on multicomputers. *IEEE Trans. Comput.* **41,** 3 (1992), 286–296.
22. Wang, H. H. A parallel method for tridiagonal equations. *ACM Trans. Math. Software* **7,** 2 (1981), 170–183.
23. Yan, W. M., and Chung, K. L. A fast algorithm for solving special tridiagonal systems. *Computing* **52** (1994), 203–211.

Received January 13, 1995; revised March 22, 1996; accepted March 25, 1996

KUO-LIANG CHUNG received the B.S., M.S., and Ph.D. in computer science and information engineering from the National Taiwan University of the Republic of China. He is now a professor in the Department of Information Management of the National Taiwan Institute of Technology. His current research interests include machine vision, computer graphics, data compression, parallel and distributed computing, and matrix computations. He is a member of ACM, IEEE, and SIAM.

WEN-MING YAN received the B.S. and M.S. in mathematics from the National Taiwan University of the Republic of China. He is now a lecturer in the Department of Computer Science and Information Engineering of the National Taiwan University. His current research interests include parallel computing and matrix computations.