JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING 24, 224–229 (1995)

Parallel B-Spline Surface Interpolation on a Mesh-Connected Processor Array

FUHUA CHENG,*[,1,2] G. W. WASILKOWSKI,*[,3] JIAYE WANG,† CAIMING ZHANG,† AND WENPING WANG‡

*Department of Computer Science, University of Kentucky, Lexington, Kentucky 40506; †Department of Computer Science, Shandong University, Jinan, Shandong, China; and ‡Department of Computing Science, University of Alberta, Edmonton, Canada T6G 2H1

A parallel implementation of Chebyshev method is presented for the B-spline surface interpolation problem. The algorithm finds the control points of a uniform bicubic B-spline surface that interpolates $m \times n$ data points on an $m \times n$ mesh-connected processor array in constant time. Hence it is optimal. Due to its numerical stability, the algorithm can successfully be used in finite precision floating-point arithmetic. © 1995 Academic Press, Inc.

1. INTRODUCTION

Modeling the shape of an image or an object usually requires the technique of parametric curve and surface interpolation [4]. B-spline curves and surfaces are frequently used in the interpolation process due to their *local property*, small *energy*, and *numerical stability* [4, 7]. The B-spline surface interpolation process, extensively studied before, is basically a problem of solving a system of linear equations.[4]

Due to the block tridiagonal form of the system of linear equations, a Gaussian elimination method is an optimal sequential algorithm to construct a uniform bicubic B-spline surface that interpolates a grid of $m \times n$ data points since its time is proportional to $m \times n$. An iterative algorithm based on the line SOR (SLOR) method has also been devised [5]. Relaxation in each line of the SLOR method is performed in parallel by using cyclic reduction [11]. If n processors are available for the relaxation process, the algorithm requires $O(m \log n)$ time to find the control points of a uniform bicubic B-spline curve that interpolates $m \times n$ points. A parallel algorithm on a systolic array was proposed by Ajjanagadde and Patniak [1]. The algorithm uses the Gauss–Seidel iteration method to find the control points of the interpolating surface. It requires $p \times n$ cells for $m \times n$ ($m \geq n$) data points where p is the number of iterations specified by the user. It requires $2n + m + p$ clock cycles to get all the control points. More recently, another parallel algorithm has been proposed by Cheng and Goshtasby [6]. The algorithm is based on a curve interpolation technique and cyclic reduction. The surface interpolation problem is split into two curve interpolation steps; each can be carried out in parallel by cyclic reduction. It requires $O(\log m + \log n)$ time to find the control points on $m \times n$ processors. None of the parallel algorithms, however, is the optimal solution yet, as the *cost* of an optimal parallel algorithm ((execution time) $\times$ (number of processors)) should equal the lower bound $O(m \times n)$ of the number of operations needed to solve the problem sequentially [2].

In this paper, we show that the Chebyshev iterative method leads to an optimal parallel algorithm on a mesh-connected processor array. The algorithm requires constant time to find the control points of a uniform B-spline surface that interpolates $m \times n$ points. The number of processors in the mesh-connected processor array is $m \times n$. Therefore, the cost of the algorithm is minimal. Moreover, this algorithm has very good numerical properties: errors due to finite precision floating-point arithmetic are very small.

The rest of the paper is presented in the following order. In Section 2, we define the problem. In Section 3, a parallel implementation of the Chebyshev iterative method for the problem is presented. In Section 4, we discuss the complexity of the algorithm in infinite precision arithmetic model. The complexity of the algorithm in finite precision arithmetic model is studied in Section 5. The concluding remarks are given in Section 6. The Appendix sketches the proof of results reported in Section 5.

2. DEFINITIONS AND PROBLEM FORMULATION

Given a grid of 3D data points $\mathbf{V}_{i,j}$, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$, our goal is to find a uniform bicubic B-spline surface that interpolates the given points. We will follow the traditional approach to construct such a cubic B-spline surface, i.e., representing it as a piecewise surface of $(m - 1) \times (n - 1)$ uniform bicubic B-spline patches whose $m \times n$ vertices (corners) interpolate the given data points. A uniform bicubic B-spline surface with $(m - 1) \times (n - 1)$ patches can be defined as

[1] Supported in part by IBM.

[2] Currently visiting the University of Tokyo and the University of Aizu, Japan.

[3] Supported in part by the NSF Grant CCR-91-14042.

[4] When viewed as digital filters, the B-spline curve and surface interpolation problem can also be solved using an *inverse filtering operation* [8].

$$\mathbf{S}(u, v) = \sum_{i=-2}^{m-1} \sum_{j=-2}^{n-1} \mathbf{P}_{i+2,j+2} N_{i,4}(u) N_{j,4}(v), \qquad (2.1)$$

where $N_{i,4}(u)$ are B-spline basis functions of order 4 (order = degree + 1) determined by the *u parameter knot vector* $(-2, -1, \ldots, m + 3)$, $N_{j,4}(v)$ are B-spline basis functions of order 4 determined by the *v parameter knot vector* $(-2, -1, \ldots, n + 3)$, and $\mathbf{P}_{i,j}$ are 3D *control points*. The *parameter range* of $\mathbf{S}(u, v)$ is $[1, m] \times [1, n]$.

To interpolate the given points at the vertices of the patches, $\mathbf{S}(u, v)$ must satisfy $\mathbf{S}(p, q) = \mathbf{V}_{p,q}$ for all $1 \le p \le m$ and $1 \le q \le n$. This is equivalent to finding $\mathbf{P}_{p,q}$ for which

$$\begin{aligned}\mathbf{P}_{p-1,q-1} &+ 4\mathbf{P}_{p,q-1} + \mathbf{P}_{p+1,q-1} + 4\mathbf{P}_{p-1,q} + 16\mathbf{P}_{p,q} \\ &+ 4\mathbf{P}_{p+1,q} + \mathbf{P}_{p-1,q+1} + 4\mathbf{P}_{p,q+1} + \mathbf{P}_{p+1,q+1} \\ &= 36\mathbf{V}_{p,q}, \quad p = 1, 2, \ldots, m;\ q = 1, 2, \ldots, n\end{aligned} \qquad (2.2)$$

Since there are more unknows than equations, we need extra conditions to get (2.2) solved uniquely. Several approaches can be used to construct additional conditions [1, 3]. For simplicity, we shall assume that we know the *boundary points* $\mathbf{P}_{p,0}$ and $\mathbf{P}_{p,n+1}$, $p = 0, 1, \ldots, m + 1$, $\mathbf{P}_{0,q}$ and $\mathbf{P}_{m+1,q}$, $q = 1, 2, \ldots, n$. (This can be done by either giving extra data points in addition to $\mathbf{V}_{p,q}$ or following the approach used in [1] to solve for the control points of the boundary curves of the interpolating surface. Actually, the algorithm proposed in the next section can be used to solve for the control points of the boundary curves as well.) Then, the equations in (2.2) can be written in matrix form as

$$A\mathbf{P} = \mathbf{F}, \qquad (2.3)$$

where $\mathbf{P} = [\mathbf{P}_1^T, \mathbf{P}_2^T, \ldots, \mathbf{P}_m^T]^T$ with $\mathbf{P}_p = [\mathbf{P}_{p,1}, \mathbf{P}_{p,2}, \ldots, \mathbf{P}_{p,n}]$, and $\mathbf{F} = [\mathbf{F}_1^T, \mathbf{F}_2^T, \ldots, \mathbf{F}_m^T]^T$ with $\mathbf{F}_p = [\mathbf{F}_{p,1}, \mathbf{F}_{p,2}, \ldots, \mathbf{F}_{p,n}]$. Each $\mathbf{F}_{p,q}$ is a function of $\mathbf{V}_{p,q}$ and some of the boundary points, and consequently is a known quantity. The matrix A is $mn \times mn$ block tridiagonal with diagonal blocks equal to $4B$ and super- and subdiagonal blocks equal to B, where B is an $n \times n$ tridiagonal matrix with 4 on the main diagonal and 1 on super- and subdiagonals.

3. THE ALGORITHM

The coefficient matrix A in (2.3) is sparse, positive definite and symmetric. Hence, it suits the Chebyshev method (see, e.g., [10]) very well.

Let a and b be positive numbers such that the interval $[a, b]$ contains all eigenvalues of the matrix A. Let $\mathbf{P}^{(0)}$ be an initial approximation of the solution $\mathbf{P} = A^{-1}\mathbf{F}$. Then, according to Chebyshev method, the $(k + 1)$st approximation $\mathbf{P}^{(k+1)}$ of $\mathbf{P}$ is given by

$$\mathbf{P}^{(k+1)} = \mathbf{P}^{(k)} + [r^{(k-1)}(\mathbf{P}^{(k)} - \mathbf{P}^{(k-1)}) - \mathbf{R}^{(k)}]/q^{(k)}, \quad k = 0, 1, 2, \ldots, \qquad (3.1)$$

where

$$\mathbf{R}^{(k)} = A\mathbf{P}^{(k)} - \mathbf{F} \qquad (3.2)$$

$$r^{(-1)} = 0, \quad r^{(k-1)} = \frac{b-a}{4}\frac{t_{k-1}}{t_k}, \quad q^{(0)} = \frac{b+a}{2}, \quad q^{(k)} = \frac{b-a}{4}\frac{t_{k+1}}{t_k}, \qquad (3.3)$$

and t_k is the value of the Chebyshev polynomial of degree k at $(b + a)/(b - a)$, i.e.,

$$t_0 = 1, \quad t_1 = \frac{b+a}{b-a}, \quad t_{k+1} = 2\left(\frac{b+a}{b-a}\right)t_k - t_{k-1}, \quad k = 1, 2, \ldots. \qquad (3.4)$$

The convergence rate of this method is

$$\|\mathbf{P}^{(k)} - \mathbf{P}\|_2 \le 2\left(\frac{\sqrt{b} - \sqrt{a}}{\sqrt{b} + \sqrt{a}}\right)^k \|\mathbf{P}^{(0)} - \mathbf{P}\|_2, \qquad (3.5)$$

where $\|\cdot\|_2$ is the l_2-norm of the mn-dimensional Euclidean space $\mathbf{E}^{mn}$.

Since the eigenvalues of A are given by

$$\lambda_{k,l} = 4\left(2 + \cos\frac{k\pi}{m+1}\right)\left(2 + \cos\frac{l\pi}{n+1}\right), \quad 1 \le k \le m,\ 1 \le l \le n, \qquad (3.6)$$

in what follows we take $a = 4$ and $b = 36$. As an initial approximation $\mathbf{P}^{(0)}$ we propose $w\mathbf{F}$ for some special $0 < w < 1$, see (4.3) and (4.7) in Section 4. $\mathbf{F}$ was chosen as the initial approximation in [1] based on the fact that $\mathbf{P}$ is relatively close to $\mathbf{F}$ (as a result of the convex hull property of B-spline surfaces). However, such a choice results in slightly longer iterative process, as we shall see in Section 4.

We will consider a model of a synchronous $m \times n$ mesh-connected processor array as given in [9]. The processor array is denoted by $M[1 \ldots m, 1 \ldots n]$. Each processor at location (i, j), $1 \le i \le m$, $1 \le j \le n$, is denoted by $M[i, j]$. Processor $M[i, j]$ is directly connected to its neighbors $M[i, j - 1]$, $M[i - 1, j]$, $M[i + 1, j]$ and $M[i, j + 1]$, provided they exist. All mn processors work in parallel with a single clock, and they all run the same program. In one step a processor can communicate with its directly connected neighbors only. The communication includes sending a data item to and receiving a data item from its neighbors. Within each step, a processor can also perform some computation in addition to communicating with its neighbors. The computing time is defined to be the number of parallel steps times the number of operations performed within each parallel step.

The algorithm starts with initializing each processor $M[i, j]$ with three pieces of information: $\mathbf{F}_{i,j}$, $\mathbf{P}_{i,j}^{(0)}$

($= w\mathbf{F}_{i,j}$), and s, the number of iterations required to reach a user specified error tolerance ε. The total computation requires $2s$ steps: two steps for each iteration. By the end of step $2s$, $M[i, j]$ contains the kth approximation of $\mathbf{P}_{i,j}$ which satisfies the condition

$$\|\mathbf{P}^{(k)} - \mathbf{P}\|_2 \leq \varepsilon \quad \text{or} \quad \frac{\|\mathbf{P}^{(k)} - \mathbf{P}\|_2}{\|\mathbf{P}\|_2} \leq \varepsilon,$$

depending on whether the absolute or relative error criterion is used. Since the operations performed by each processor are the same, we shall discuss them for one processor only. Furthermore, although points are used in our discussion, it should be understood that the algorithm is performed for one component of the points only; the computation for the other two components is the same.

The computation is performed as follows. During the $(k + 1)$st iteration, $k \geq 0$, the computation required for $\mathbf{P}^{(k+1)}_{i,j}$ is given by

$$\begin{aligned}\mathbf{P}^{(k+1)}_{i,j} = \mathbf{P}^{(k)}_{i,j} &+ [r^{(k-1)}(\mathbf{P}^{(k)}_{i,j} - \mathbf{P}^{(k-1)}_{i,j}) - (\mathbf{P}^{(k)}_{i-1,j-1} \\ &+ 4\mathbf{P}^{(k)}_{i,j-1} + \mathbf{P}^{(k)}_{i+1,j-1} + 4(\mathbf{P}^{(k)}_{i-1,j} + 4\mathbf{P}^{(k)}_{i,j} \\ &+ \mathbf{P}^{(k)}_{i+1,j}) + \mathbf{P}^{(k)}_{i-1,j+1} + 4\mathbf{P}^{(k)}_{i,j+1} + \mathbf{P}^{(k)}_{i+1,j+1}) \\ &- \mathbf{F}_{i,j}]/q^{(k)}.\end{aligned} \tag{3.7}$$

This will be achieved in two steps: steps $2k + 1$ and $2k + 2$. In step $2k + 1$, $M[i, j]$ computes the following items:

$$\alpha^{(k)}_{i,j} \equiv (\mathbf{P}^{(k)}_{i-1,j} + \mathbf{P}^{(k)}_{i+1,j}) + 4\mathbf{P}^{(k)}_{i,j},$$
$$\beta^{(k)}_{i,j} \equiv r^{(k-1)}(\mathbf{P}^{(k)}_{i,j} - \mathbf{P}^{(k-1)}_{i,j}),\ q^{(k)}.$$

$r^{(k-1)}$ and $q^{(k)}$ are defined in (3.3). The computation of $\alpha^{(k)}_{i,j}$ requires communication of $M[i, j]$ with its horizontally connected neighbors, i.e., sending a copy of $\mathbf{P}^{(k)}_{i,j}$ to each of its horizontal neighbors $M[i - 1, j]$ and $M[i + 1, j]$, and receiving a copy of $\mathbf{P}^{(k)}_{i-1,j}$ and $\mathbf{P}^{(k)}_{i+1,j}$ from $M[i - 1, j]$ and $M[i + 1, j]$, respectively. This situation is illustrated in Fig. 1a where dotted lines indicate that no communication is required between vertically connected processors. In step $2k + 2$, $M[i, j]$ computes the following items:

$$\begin{aligned}\delta^{(k)}_{i,j} &\equiv \alpha^{(k)}_{i,j-1} + \alpha^{(k)}_{i,j+1} + 4\alpha^{(k)}_{i,j}, \\ &\mathbf{P}^{(k)}_{i,j} + (\beta^{(k)}_{i,j} - \mathbf{F}_{i,j} - \delta^{(k)}_{i,j})/q^{(k)}.\end{aligned} \tag{3.8}$$

The computation of $\delta^{(k)}_{i,j}$ requires communication of $M[i, j]$ with vertically connected neighbors, i.e., sending a copy of $\alpha^{(k)}_{i,j}$ to each of $M[i, j - 1]$ and $M[i, j + 1]$ and receiving a copy of $\alpha^{(k)}_{i,j-1}$ and $\alpha^{(k)}_{i,j+1}$ from $M[i, j - 1]$ and $M[i, j + 1]$, respectively. The situation is depicted in Fig. 1b. The second value of (3.8) equals the right-hand side of (3.7). Therefore, by the end of step $2k + 2$, $M[i, j]$ contains the $(k + 1)$st approximation of $\mathbf{P}_{i,j}$.

To compute $\mathbf{P}^{(k+1)}$, each processor needs to perform 18 arithmetic operations: 8 multiplications and 10 additions. Among them, 4 multiplications and 1 addition are needed to compute $r^{(k-1)}$ and $q^{(k)}$. One can save one multiplication per iteration by using a special version of the Chebyshev method due to Woźniakowski; see [12]. With initial values

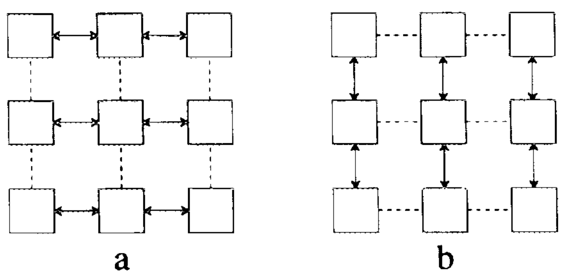

FIG. 1. Communication during odd steps (a) and even steps (b).

$$c = \frac{a + b}{2} = 20, \quad d = \left(\frac{b - a}{4}\right)^2 = 64,$$
$$q^* = \frac{a + b + 2\sqrt{ab}}{4} = 16,$$
$$r^{(-1)} = 0, \quad q^{(0)} = c,$$
$$r^{(0)} = \frac{2d}{c} = 6.4, \quad q^{(1)} = \frac{a + b}{4} + \frac{ab}{a + b} = 13.6,$$
$$\sigma^{(1)} = \frac{2\sqrt{ab}}{a + b}\frac{d}{q^*} = 2.4,$$

we have for $k = 2, 3 \ldots$

$$r^{(k-1)} = d/q^{(k-1)}, \quad \sigma^{(k)} = r^{(k-1)}\sigma^{(k-1)}/q^*, \quad q^{(k)} = q^* - \sigma^{(k)}.$$

The evaluation of t_k is no longer required. Most importantly, this approach is numerically stable, as proven in [12], which makes the Chebyshev method very appealing.

4. COMPLEXITY WITH INFINITE PRECISION

The number of operations performed in each parallel iteration step equals 17. Hence, the complexity of the algorithm with infinite precision equals $17 \times s$ where s is the number of iterations. Depending on whether absolute or relative error criterion is used, the number s is given by (4.5) or (4.9) when $\mathbf{P}^{(0)} = w\mathbf{F}$ with optimally chosen w; see (4.3) or (4.7). We begin with the absolute error criterion. Note that

$$\begin{aligned}\|\mathbf{P}^{(0)} - \mathbf{P}\|_2 = \|w\mathbf{F} - A^{-1}\mathbf{F}\|_2 &= \|(wI - A^{-1})\mathbf{F}\|_2 \\ &\leq \|wI - A^{-1}\|_2\|\mathbf{F}\|_2.\end{aligned} \tag{4.1}$$

Hence, due to (3.5),

$$\|\mathbf{P}^{(k)} - \mathbf{P}\|_2 \leq 2^{-k+1}\|\mathbf{P}^{(0)} - \mathbf{P}\| \leq 2^{-k+1}\|wI - A^{-1}\|_2\|\mathbf{F}\|_2. \tag{4.2}$$

Since the eigenvalues $\lambda(A^{-1})$ of A^{-1} are bounded by 1/36 and 1/4, we have the following inequality for the eigenvalues of $wI - A^{-1}$:

$$|\lambda(wI - A^{-1})| = |w - \lambda(A^{-1})| \le \max(|w - 1/4|, |w - 1/36|).$$

The smallest value of the right-hand side occurs when

$$w = 5/36. \tag{4.3}$$

Then $\|wI - A^{-1}\|_2 \le |5/36 - 1/4| = 1/9$, and (4.2) yields

$$\|\mathbf{P}^{(k)} - \mathbf{P}\|_2 \le \frac{2^{-k+1}}{9} \|\mathbf{F}\|_2. \tag{4.4}$$

This shows that for $\mathbf{P}^{(0)} = 5\mathbf{F}/36$, the Chebyshev method reaches the error tolerance ε with

$$s = \left\lceil \log_2 \frac{2\|\mathbf{F}\|_2}{9\varepsilon} \right\rceil. \tag{4.5}$$

The right-hand side of (4.5) does not depend on m and n, it depends on ε and $\|\mathbf{F}\|_2$ only. $\|\mathbf{F}\|_2$ can be estimated in the input phase. Therefore, as long as the norm of $\mathbf{F}$ and the error tolerance are fixed, the number of iterations required by the algorithm is a constant and, consequently, the execution time of the algorithm is $O(1)$.

If $\mathbf{P}^{(0)} = \mathbf{F}$ were chosen as the initial approximation, the algorithm would require more iteration steps to reach the user specified error tolerance. In this case, we would have

$$\|\mathbf{P}^{(0)} - \mathbf{P}\|_2 = \|\mathbf{F} - A^{-1}\mathbf{F}\|_2 = \|(I - A^{-1})\mathbf{F}\|_2 \le \|I - A^{-1}\|_2\|\mathbf{F}\|_2.$$

Since $\|I - A^{-1}\|_2 \le 35/36$, $\|\mathbf{P}^{(k)} - \mathbf{P}\|_2 \le 2^{-k+1}(35/36)\|\mathbf{F}\|_2$. Hence, to reach the user specified error tolerance, $s = \lceil \log_2(35\|\mathbf{F}\|_2/18\varepsilon)\rceil$ would be required. Note that this is larger than s in (4.5) by at least 3.

It should be pointed out that other iterative methods can be used in the algorithm as long as the number of iterations does not depend on the size of the problem. Nevertheless, the Chebyshev method seems to be the best choice for this particular problem in that the rate of convergence of the Chebyshev method is much faster than the rate of many other methods. For instance, it is possible to show (following a similar approach as above) that the SOR method (with the relaxation value being 1/20, the best case) provides the following rate of convergence:

$$\|\mathbf{P}^{(k)} - \mathbf{P}\|_2 \le \frac{1}{9}\left(\frac{4}{5}\right)^k \|\mathbf{F}\|_2.$$

Therefore, after four iterations, the error in (4.4) would only be 1.4% of $\|\mathbf{F}\|_2$, while the error in SOR is still 4.5% of $\|\mathbf{F}\|_2$.

We now proceed with the relative error criterion. Note that with $\mathbf{P}^{(0)} = w\mathbf{F}$, we have

$$\|\mathbf{P}^{(0)} - \mathbf{P}\|_2 = \|w\mathbf{F} - \mathbf{P}\|_2 = \|(wA - I)\mathbf{P}\|_2 \le \|wA - I\|_2\|\mathbf{P}\|_2. \tag{4.6}$$

The eigenvalues $\lambda(I - wA)$ of $wA - I$ are bounded by $\max(|1 - 4w|, |1 - 36w|)$. Moreover,

$$w = 1/20 \tag{4.7}$$

minimizes the above maximum with $\|I - (1/20)A\|_2 \le 4/5$. Therefore, (3.5) and (4.6) yield

$$\frac{\|\mathbf{P}^{(k)} - \mathbf{P}\|_2}{\|\mathbf{P}\|_2} \le 2^{-k+1}\frac{4}{5}. \tag{4.8}$$

This shows that, with $\mathbf{P}^{(0)} = \mathbf{F}/20$, the Chebyshev method reaches a specified relative error tolerance ε_r in s steps with

$$s = \left\lceil \log_2 \frac{8}{5\varepsilon_r} \right\rceil. \tag{4.9}$$

In particular, to get k correct binary digits, $k + 2$ iterations would be sufficient.

5. COMPLEXITY WITH FINITE PRECISION

In this section, we analyze the complexity of the algorithm when the operations are performed in a finite precision floating-point arithmetic environment with a unit *roundoff error* u. For simplicity, we assume *rounding* in the arithmetic operations. Hence, $u = 2^{-m}$, where m is the number of binary digits in the mantissa of a floating-point number.

Let $\hat{\mathbf{P}}^{(k+1)}$ be the values of $\mathbf{P}^{(k+1)}$ computed in floating-point arithmetic after $k + 1$ iterations of the algorithm and let $\hat{\mathbf{e}}^{(k+1)}$ be the corresponding error

$$\hat{\mathbf{e}}^{(k+1)} = \mathbf{P} - \hat{\mathbf{P}}^{(k+1)}.$$

We have the following upper bound on the error when $\mathbf{P}^{(0)} = (1/20)\mathbf{F}$:

$$\|\hat{\mathbf{e}}^{(k+1)}\| \le 2^{-k}\|\mathbf{e}^{(0)}\|_2 + 164\,u\|\mathbf{P}\|_2 \le (2^{-k+2}/5 + 164\,u)\|\mathbf{P}\|_2. \tag{5.1}$$

The proof of (5.1) takes more than 10 pages of tedious estimations. Hence, we only outline it in the Appendix.

The bound (5.1) provides us with two pieces of information. First, the estimate of the relative error $\|\hat{\mathbf{e}}^{(k+1)}\|_2/\|\mathbf{P}\|_2$ converges to $164u$ when k tends to infinity. This means that one can solve the problem to within $m - 9$ most significant binary digits of the exact solution. Second, to approximate $\mathbf{P}$ by $\hat{\mathbf{P}}^{(k)}$ that carries, r, $r \le m - 9$, correct binary digits, k should satisfy $2^{-k+1}4/5 + 2^{-m}164 \le 2^{-(r+1)}$. Hence, we need at most k iterations with $k = r + 4$ if $r = m - 9$, $k = r + 3$ if $r = m - 10$, and $k = r + 2$ if $r \le m - 11$.

6. CONCLUDING REMARKS

The algorithm presented in this paper can easily be realized since communication required is simple and computation for all the processors is controlled by a central clock only. It is especially suited for interpolation problems with large number of interpolation points and fixed problem size. The algorithm is also appealing for its small numerical errors.

The computation architecture used in the algorithm could also be used in other computation-extensive problems in graphics and geometric modeling. For instance, a mesh-connected processor array seems to be well suited for B-spline surface evaluation using knot insertion. However, it needs further study.

APPENDIX

We sketch the proof of (5.1) here. The proof is composed of four major steps; each step consists of relatively straightforward, yet lengthy estimates. We will give a sketch of each step only.

In the following, a quantity computed in finite-precision floating-point arithmetic will be capped with a "^" to distinguish it from the exact quantity. For instance, the computed values of $\sigma^{(k)}$, $r^{(k)}$, and $q^{(k)}$ will be denoted by $\hat{\sigma}^{(k)}$, $\hat{r}^{(k)}$, and $\hat{q}^{(k)}$, respectively.

Step 1. Let $e^{(r,k)}$, $e^{(q,k)}$, and $e^{(\sigma,k)}$ be the relative errors in computing $r^{(k)}$, $q^{(k)}$, and $\sigma^{(k)}$, i.e.,

$$e^{(r,k)} = |r^{(k)} - \hat{r}^{(k)}|/|r^{(k)}|, \quad e^{(q,k)} = |q^{(k)} - \hat{q}^{(k)}|/|q^{(k)}|,$$
$$e^{(\sigma,k)} = |\sigma^{(k)} - \hat{\sigma}^{(k)}|/|\sigma^{(k)}|.$$

Then $e^{(r,0)} \le u$, $e^{(r,1)} \le 2u$, $e^{(q,0)} = 0$, $e^{(q,1)} \le u$, $e^{(\sigma,1)} \le u$, and for $k \ge 2$

$$e^{(r,k)} \le \left(2 + \frac{6}{2^k + 1}\right) u < 3u,$$

$$e^{(q,k)} \le \left(1 + \frac{6}{2^{2k+2} + 1}\right) u < 2u, \quad \text{and}$$

$$e^{(\sigma,k)} \le 2u.$$

Step 2. Given a vector $\mathbf{F}$, the error δ in computing the residual $\gamma = A\mathbf{P} - \mathbf{F}$ satisfies

$$\|\delta\| = \|\gamma - \hat{\gamma}\| \le u(112\|\mathbf{P}\| + \|\mathbf{F}\|).$$

Step 3. Let $\zeta^{(k)}$ be the rounding error due to floating-point operations in $(k + 1)$st step. Then

$$\|\zeta^{(k)}\| \le u \le \|\mathbf{P}^{(k)}\| \left(1 + \frac{112}{q^{(k)}}\right) + \frac{u}{q^{(k)}} \|\mathbf{F}\| + \frac{2u + e^{(q,k)}}{q^{(k)}} \|\gamma^{(k)}\|$$
$$+ (5u + e^{(r,k-1)} + e^{(q,k)}) \frac{r^{(k-1)}}{q^{(k)}} \|\mathbf{P}^{(k)} - \mathbf{P}^{(k-1)}\|.$$

Since $\|\mathbf{P}^{(k)}\| \le \|\mathbf{e}^{(k)}\| + \|\mathbf{P}\|$, it follows that

$$\|\zeta^{(k)}\| \le u\|\mathbf{P}\|A^{(k)} + u\|\mathbf{F}\| \frac{1}{q^{(k)}} + \|\mathbf{e}^{(k)}\|B^{(k)}$$
$$+ \|\mathbf{e}^{(k)} - \mathbf{e}^{(k-1)}\|C^{(k)},$$

where

$$A^{(k)} = 1 + \frac{112}{q^{(k)}}, \qquad (A^{(k)} \underset{k}{\to} 8),$$

$$B^{(k)} = u + \frac{2u + 36e^{(q,k)} + 112u}{q^{(k)}}, \qquad \left(B^{(k)} \underset{k}{\to} \frac{45}{4} u\right),$$

$$C^{(k)} = \frac{r^{(k-1)}}{q^{(k)}} (5u + e^{(r,k-1)} + e^{(q,k)}), \quad (C^{(k)} \underset{k}{\to} 2u).$$

Step 4. This step uses a result of Woźniakowski (Thm. 3.1 on p. 197 of [12]) which states that

$$\|\hat{\mathbf{e}}^{(k+1)}\| \le 2^{-k}\|\mathbf{e}^{(0)}\| + \sum_{i=0}^{k} ((2 - \beta_{i+1})\|W_{k-i}(A)\|$$
$$+ (\beta_{i+1} - 1)\|R_{k-i}(A)\| \cdot \|\zeta^{(i)}\|)$$

with

$$\beta_{i+1} = \frac{5}{2}\frac{t_{i+1}}{t_{i+2}} = \frac{5}{2}\frac{2^{i+1} + 2^{-i-1}}{2^{i+2} + 2^{-i-2}} > \frac{5}{4}, \quad \|W_{k-i}(A)\| = \frac{1}{t_{k-i}},$$
$$\|R_{k-i}(A)\| = \frac{k - i + 1}{t_{k-i}}.$$

Since the right-hand side increases with β_{i+1}'s, we overestimate it by taking $\beta_{i+1} = 2$. Then

$$\|\hat{\mathbf{e}}^{(k+1)}\| \le 2^{-k}\|\mathbf{e}^{(0)}\| + 2 \sum_{i=0}^{k} 2^{-k+i}(k - i + 1)\|\zeta^{(i)}\|.$$

Using the estimate on $\|\zeta^{(i)}\|$ obtained in Step 3, one can verify that indeed

$$\|\hat{\mathbf{e}}^{(k+1)}\| \le 2^{-k}\|\mathbf{e}^{(0)}\| + 164\, u\, \|\mathbf{P}\| \le \left(2^{-k}\frac{4}{5} + 164u\right)\|\mathbf{P}\|$$

as claimed.

REFERENCES

1. Ajjanagadde, V. G., and Patnaik, L. M. Systolic architecture for B-spline surfaces. *Internat. J. Parallel Program.* **15**, 6 (Dec. 1986), 551-565.

2. Akl, S. G. *Parallel Sorting algorithms*. Academic Press, Orlando, FL, 1985.
3. Barsky, B. A. End conditions and boundary conditions for uniform B-spline curve and surface representation. *Computers in Industry* **3,** 1 & 2 (Mar. & June 1982), 17–29.
4. 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.
5. 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.
6. Cheng, F., and Goshtasby, A. A parallel B-spline surface fitting algorithm. *ACM Trans. Graphics* **8,** 1 (Jan. 1989), 41–50.
7. de Boor, C. *A Practical Guide to Splines*. Springer-Verlag, New York, 1978.
8. 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.
9. Schnorr, C. P., and Shamir, A. An optimal sorting algorithm for mesh-connected computers. In *Proc. 18th ACM Symp. Theory Comput.*, 1986, 255–263.
10. Steifel, E. Kernel polynomials in linear algebra and their numerical applications, *NBS. Appl. Math. Ser.* **49** (1958), 1–22.
11. Sweet, R. A. A cyclic reduction algorithm for solving tridiagonal systems of arbitrary dimension. *SIAM J. Numer. Anal.* **14,** 1977, 706–719.
12. Woźniakowski, H. Numerical stability of the Chebyshev method for the solution of large linear systems. *Numer. Math.* **28,** 2 (1977), 191–209.

FUHUA (FRANK) CHENG is an associate professor of computer science at the University of Kentucky. He received a B.S. and an M.S. in mathematics from the National Tsing Hua University in Taiwan, and an M.S. in mathematics, an M.S. in computer science, and a Ph.D. in approximation theory and numerical analysis from the Ohio State University. His research interests include computer graphics, geometric/solid modeling, and parallel computing in graphics and geometric modeling. Professor Cheng received the 1985 Dr. Sun Yat-Sen Technical Invention Award for his contribution to the design and development of the special hardware known as the Bezier Curve Generator.

G. W. WASILKOWSKI received a Ph.D. from University of Warsaw, Poland, in 1980. Currently, he is a professor of computer science at the University of Kentucky. His research interests include complexity of continuous problems and numerical analysis. He is a coauthor of two research monographs and some 50 research papers.

JIAYE WANG is a professor of computer science at Shandong University, China. He is currently a visiting professor at GINTIC Institute of Manufacturing Technology, Nanyang Technological University, Singapore. His research interests are in computational geometry, computer graphics, and computer-aided geometry design.

CAIMING ZHANG is a lecturer of computer science at the Shandong University, China. He received a B.S. and an M.E. in computer science from Shandong University, in 1982 and 1984, respectively, and a Ph.D. in computer science from the Tokyo Institute of Technology, Tokyo, Japan in 1994. He is currently a visiting scholar in the Department of Information Processing, Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology at Nagatsuta, Yokohama, Japan. His research interests include computational geometry for CAD/CAM, computer graphics, and image processing.

WENPING WANG received a B.Sc. and an M.Eng. in computer science from Shandong University, China, in 1983 and 1986, respectively. He received a Ph.D. in computer science from the University of Alberta, Canada, in 1992. He is currently a lecturer of computer science at the University of Hong Kong. His research interests include computer graphics, geometric modeling, and computational geometry.

Received May 14, 1993; revised February 11, 1994; accepted March 3, 1994