SIAM J. MATRIX ANAL. APPL.
Vol. 17, No. 3, pp. 470–488, July 1996

002

A BLOCK-GTH ALGORITHM FOR FINDING THE STATIONARY VECTOR OF A MARKOV CHAIN*

DIANNE P. O'LEARY† AND YUAN-JYE JASON WU‡

Abstract. Grassman, Taksar, and Heyman have proposed an algorithm for computing the stationary vector of a Markov chain. Analysis by O'Cinneide confirmed the results of numerical experiments, proving that the GTH algorithm computes an approximation to the stationary vector with low relative error in each component. In this work, we develop a block form of the GTH algorithm, more efficient on high-performance architectures, and show that it too produces a vector with low relative error. We demonstrate the efficiency of the algorithm on vector processors and on workstations with hierarchical memory.

Key words. Markov chain, GTH algorithm, stationary vector, relative error bounds

AMS subject classifications. 65F15, 60J10, 65G05

1. Introduction.

We consider the problem of computing the steady state distribution of a finite, discrete time, irreducible Markov chain. Equivalently, we seek the left eigenvector π corresponding to the eigenvalue 1 of a stochastic matrix P:

$$\pi P = \pi, \quad \pi e = 1, \quad Pe = e, \quad 0 \le p_{ij}, \quad i,j = 1,2,\ldots,n, \tag{1}$$

where e is the column vector of ones.

Grassman, Taksar, and Heyman [3] used probability theory to develop an algorithm (the *GTH algorithm*) for computing π by successively reducing the state space. The algorithm works with the *generator* matrix $G = P - I$ having zero row sums. It proved to be surprisingly accurate in numerical experiments and was later recognized as a variant of Gaussian elimination. The key difference is that the main diagonal element of the triangular factor is computed as the negative sum of the computed off-diagonal elements, and thus the row sum property is preserved. O'Cinneide [4] later analyzed the GTH algorithm, showing that the computed vector π has low relative error in each component.

No single algorithm runs at peak efficiency on each of the wide variety of computer architectures in current use. For some architectures, a simple count of arithmetic operations provides an accurate prediction of performance. For machines with vector pipelines and multilevel memories, however, the number of loads and stores of data can be a more critical factor. For parallel architectures, the data layout and communication patterns are crucial.

A common approach to algorithm design is to consider a parameterized family of algorithms that can be tuned to different architectures. *Block-matrix algorithms* provide one such parameterization, and their use is widespread in portable libraries such as LAPACK. There is a considerable body of literature on the error analysis of such block algorithms. Backward error bounds are established, for example, in [2]. The O'Cinneide bounds for GTH are much stronger than these results, since

* Received by the editors January 13, 1994; accepted for publication (in revised form) by L. Kaufman August 25, 1995. This work was supported by NSF grant 91-15568. Access to the Cray and Convex machines was provided by the National Institute for Standards and Technology.

† Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742 (oleary@cs.umd.edu).

‡ Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439-4801 (jwu@mcs.anl.gov).

properties of the matrix G allowed him to obtain forward error bounds independent of the condition number of the matrix.

The purpose of our work is to define a *block-GTH algorithm* (§2), analyze its error properties (§3) to obtain results analogous to those of O'Cinneide, and determine the performance of the algorithm on various architectures (§4).

2. The block-GTH algorithm. Consider an irreducible *generator* G of dimension $n \times n$; i.e., G is a matrix with nonnegative off-diagonal elements and row sums equal to zero. We seek the row vector π satisfying

$$\pi G = 0, \quad \pi e = 1.$$

The GTH algorithm reduces G to lower triangular form. It is an iterative process, working with a matrix G_k of dimension $(n-k) \times (n-k)$ at the kth stage—a generator from which k states have been eliminated. Let $G_0 = G$, and partition as

$$G_k = \begin{bmatrix} A_k & B_k \\ C_k & D_k \end{bmatrix}, \tag{2}$$

where A_k is the $(1,1)$ element of G_k and B_k is the remaining part of the first row. Then if $p_k G_k = 0$, it is also true that

$$0 = p_k G_k \begin{bmatrix} 1 & -A_k^{-1} B_k \\ 0 & I \end{bmatrix} = p_k \begin{bmatrix} A_k & 0 \\ C_k & D_k - C_k A_k^{-1} B_k \end{bmatrix}.$$

Define

$$G_{k+1} = D_k - C_k A_k^{-1} B_k. \tag{3}$$

Note from (2) that $[C_k \ D_k]e = 0$ and $[A_k \ B_k]e = 0$, and so

$$[0, G_{k+1}]e = [C_k \ D_k]e - C_k A_k^{-1} [A_k \ B_k]e = 0.$$

Furthermore, the sign pattern is preserved, and G_{k+1} is a generator [4].

If we have a nonzero row vector p_{k+1} satisfying $p_{k+1} G_{k+1} = 0$, then the nonzero row vector defined by

$$p_k = \begin{bmatrix} -p_{k+1} C_k A_k^{-1} & p_{k+1} \end{bmatrix} \tag{4}$$

satisfies $p_k G_k = 0$. Thus, we have reduced the original problem to that of solving $p_{k+1} G_{k+1} = 0$, a problem with one fewer state.

The main difference between the GTH algorithm and standard Gaussian elimination is in the computation of A_k. In Gaussian elimination, this element is accumulated as a result of the updates (3). In the GTH algorithm, A_k is computed as the negative sum of the off-diagonal elements in the first row of G_k. A minor difference between the algorithms is that the GTH algorithm is usually formulated so that the last state (rather than the first one) is the first to be eliminated, but in this work we will eliminate the first state first, as in Gaussian elimination.

These relations form the basis for the GTH algorithm, which we now state more formally.

ALGORITHM GTH

FACTORIZATION PHASE

1. Let $G_0 = G$.

2. For $k = 0, 1, \ldots, n-2$

2.1. Partition G_k as in (2), where A_k is calculated as $A_k = -B_k e$.

2.2. Define G_{k+1} by (3).

End for.

BACKSUBSTITUTION PHASE

3. Let $p_{n-1} = 1$.

4. For $k = n-2, n-3, \ldots, 0$

4.1 Define p_k by (4).

End for.

5. Renormalize $\pi = p_0/(p_0 e)$.

The LU factors of G can be defined using quantities computed in the factorization phase of the algorithm:

$$G = \begin{array}{|c|c|c|c|} \hline A_0 & \multicolumn{3}{c|}{0} \\ \hline & A_1 & \multicolumn{2}{c|}{0} \\ \hline C_0 & C_1 & \ddots & \\ \hline & & & A_{n-1} \\ \hline \end{array} \qquad \begin{array}{|c|c|c|c|} \hline 1 & \multicolumn{3}{c|}{A_0^{-1}B_0} \\ \hline & 1 & \multicolumn{2}{c|}{A_1^{-1}B_1} \\ \hline 0 & 0 & \ddots & \\ \hline & & & 1 \\ \hline \end{array}$$

and we will make use of this fact later.

The GTH algorithm is easy to implement and numerically stable, but its efficiency on certain computer architectures can be disappointing. Notice, for example, that the (n, n) element of G is accessed and updated $n-1$ distinct times. It is well known that block-oriented algorithms can reduce the memory traffic for elimination algorithms, so we now direct our attention to developing a block-GTH algorithm.

The basis of the block-GTH algorithm is a *block* partitioning of the matrix G_k: we partition as in (2), but now A_k is an $l \times l$ matrix, rather than a single element. Similarly, B_k has l rows. The block size l can be tuned to achieve improved efficiency on various architectures, as discussed in §4. The generator G_{k+l} and its eigenvector p_{k+l} are expressed in terms of G_k and p_k by formulas similar to (3) and (4):

$$G_{k+l} = D_k - C_k A_k^{-1} B_k \,, \tag{5}$$

$$p_k = \begin{bmatrix} -p_{k+l} C_k A_k^{-1} & p_{k+l} \end{bmatrix} . \tag{6}$$

Rather than division by a scalar, (5) and (6) now require solution of linear systems involving the blocks A_k. This can easily be done using an LU factorization of these blocks.

The other main implementation issue is the correction of the main diagonal elements of A_k. To avoid memory traffic, we wish to do this with minimal access to the elements of B_k. Notice that the matrix

$$H_k = \begin{bmatrix} A_k & B_k e \\ 0 & 0 \end{bmatrix} \tag{7}$$

is also a generator, and the diagonal corrections that would be generated in step 2 of the GTH algorithm applied to this matrix are the same as those that GTH would generate for the original problem at the corresponding steps. For instance, after elimination in the first row of H_k, the elements $h_i \equiv (B_k e)_i$ are updated as

$$\bar{h}_i = h_i - \frac{\bar{a}_{i1} h_1}{\bar{a}_{11}} \tag{8}$$

$$= \sum_j b_{ij} - \frac{\bar{a}_{i1} \sum_j b_{1j}}{\bar{a}_{11}}$$

$$= \sum_j \left(b_{ij} - \frac{\bar{a}_{i1} b_{1j}}{\bar{a}_{11}} \right), \tag{9}$$

where $\bar{a}_{i1}$ is the updated value of a_{i1}, j ranges over the column indices in B_k, and $i = 2, \ldots, l$. Equation (9) shows that the update to the row sum vector $B_k e$ in (8) is mathematically equivalent to taking the row sum after correcting the matrix B in (9).

This is the basis for the block-GTH algorithm. For convenience in notation, we assume that l evenly divides n, although varying block sizes can be easily handled.

ALGORITHM BLOCK-GTH-I

FACTORIZATION PHASE

1. Let $G_0 = G$. Given an integer l between 1 and n, let $\widehat{n} = n/l$.

2. For $k = 0, l, \ldots, (\widehat{n} - 1)l$

2.1. Partition G_k as in (2), where A_k is an $l \times l$ matrix.

2.2. Apply the factorization phase of algorithm GTH to the matrix H_k defined by (7).

2.3. If $k \neq (\widehat{n} - 1)l$, define G_{k+l} by (5):

$$G_{k+l} = D_k - C_k(A_k^{-1} B_k),$$

where the factors from 2.2 are used to compute the expression in parentheses.

End for.

BACKSUBSTITUTION PHASE

3. Let $p_{(\widehat{n}-1)l}$ be computed from the backsubstitution phase of the GTH algorithm applied to $G_{(\widehat{n}-1)l}$.

4. For $k = (\widehat{n} - 2)l, (\widehat{n} - 3)l, \ldots, 0$

4.1. Define p_k by (6), again using the factors of A_k.

End for.

5. Renormalize $\pi = p_0/(p_0 e)$.

As an alternative to block-GTH-I, which relies on a block lower triangular factor with A_k on the main diagonal, we can compute a standard LU factorization of G:

$$G = \begin{array}{|c|c|c|c|} \hline L_0 & \multicolumn{3}{c|}{0} \\ \hline & L_l & 0 & \\ \cline{2-3} C_0U_0^{-1} & C_lU_l^{-1} & \ddots & \\ \cline{3-4} & & & L_{(\widehat{n}-1)l} \\ \hline \end{array} \qquad \begin{array}{|c|c|c|c|} \hline U_0 & \multicolumn{3}{c|}{L_0^{-1}B_0} \\ \hline & U_l & L_l^{-1}B_l & \\ \cline{2-3} 0 & 0 & \ddots & \\ \cline{3-4} & & & U_{(\widehat{n}-1)l} \\ \hline \end{array} .$$

This algorithm takes the following form.

ALGORITHM BLOCK-GTH-II

FACTORIZATION PHASE

1. Let $G_0 = G$. Given an integer l between 1 and n, let $\widehat{n} = n/l$.

2. For $k = 0, l, \ldots, (\widehat{n}-1)l$

2.1. Partition G_k as in (2), where A_k is a $l \times l$ matrix.

2.2. Apply the factorization phase of algorithm GTH to the matrix H_k defined by (7), applying the same updates to C_k (i.e., computing $C_kU_k^{-1}$).

2.3. If $k \neq (\widehat{n}-1)l$, define G_{k+l} by (5):

$$G_{k+l} = D_k - (C_kU_k^{-1})(L_k^{-1}B_k).$$

End for.

BACKSUBSTITUTION PHASE

3–5. Use the backsubstitution phase of algorithm GTH, organizing the computations by single rows or by blocks of l rows.

3. Error analysis. As we mentioned before, the left eigenvector computed by the GTH algorithm has a small entry-wise relative error bound. Our next task is a rounding error analysis for the block-GTH algorithm in order to demonstrate that it preserves this error property.

Let us introduce some notation first. We use the special symbols $\langle\gamma\rangle$ from Appendix 3 of [5]. Let $\mathbf{u}$ be the unit roundoff in floating-point arithmetic. Then we write

$$\langle\gamma\rangle = \frac{(1+a_1)(1+a_2)\cdots(1+a_\alpha)}{(1+b_1)(1+b_2)\cdots(1+b_\beta)}$$

whenever $|a_i| \leq \mathbf{u}$, $|b_i| \leq \mathbf{u}$, and $\alpha + \beta = \gamma$. The $\langle\gamma\rangle$ symbols satisfy the relations

$$\langle\gamma\rangle\ \langle\alpha\rangle = \langle\gamma+\alpha\rangle$$

and

$$\frac{\langle\gamma\rangle}{\langle\alpha\rangle} = \langle\gamma+\alpha\rangle$$

and make floating-point expressions simple and clear. Let us denote the floating-point operators with a "hat." The error analysis of floating-point operations is based on the following rules:

1. $\langle\alpha\rangle\ a\ \widehat{\pm}\ \langle\beta\rangle\ b = \langle\alpha+1\rangle\ a \pm \langle\beta+1\rangle\ b$,
2. $\langle\alpha\rangle\ a\ \widehat{*}\ \langle\beta\rangle\ b = \langle\alpha+\beta+1\rangle\ a * b$,

3. $\langle\alpha\rangle\ a\ \widehat{/}\ \langle\beta\rangle\ b = \langle\alpha+\beta+1\rangle\ a/b$.

Note that these rules also hold if we interchange the operators on the left-hand side with those on the right-hand side. A fundamental property upon which we heavily rely is that if no cancellation occurs in forming a sum or difference (i.e., if the two operands have the same sign), then

$$\langle\alpha\rangle\ a\ \widehat{\pm}\ \langle\beta\rangle\ b = \langle\max(\alpha,\beta)+1\rangle\ (a \pm b).$$

The following theorem gives the error bounds for the GTH algorithm.

THEOREM 1 (see O'Cinneide [4]). *For any stochastic matrix P of order n with stationary vector π, the accuracy of the left eigenvector $\tilde{\pi}$ computed by the GTH algorithm using floating-point arithmetic is characterized by*

$$\tilde{\pi}_i = \langle 2\phi(n)+n\rangle\ \pi_i,\ \ i = 1,\ldots,n,$$

where $\phi(n) = (2n^3+6n^2-8n)/3$. Furthermore, if $(2\phi(n)+n)\mathbf{u} \leq .1$, then

$$\frac{|\tilde{\pi}_i - \pi_i|}{\pi_i} \leq 1.06(2\phi(n)+n)\mathbf{u}\ .$$

The formula for $\phi(n)$ is derived by induction [4] and makes use of a theorem of Tweedie [6], which says that if two irreducible generators G and $\tilde{G}$ of order m have the property that $\tilde{g}_{ij} = \langle\alpha\rangle\ g_{ij},\ i \neq j$, then their eigenvectors satisfy

$$\tilde{p}_i = \langle 2m\alpha\rangle\ p_i\ ,\ \ i = 1,\ldots,m\ .$$

The proof strategy is shown in Figure 1.

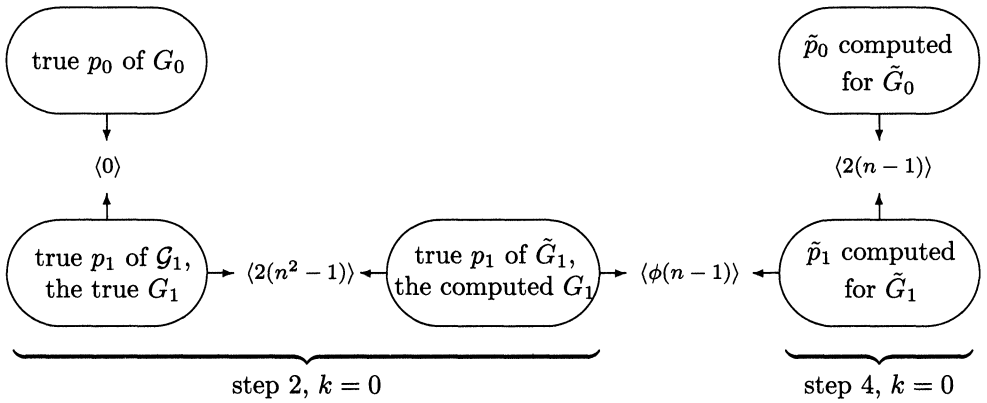

FIG. 1.

The bound for step 2 comes from verifying that given a generator G_0 in algorithm GTH, the relative error for the off-diagonal entries of the computed generator G_1 is $\langle n+1\rangle$. Since the generator G_1 is of order $n-1$, then by Tweedie's result the true eigenvector of computed G_1 has component-wise error bounded by

$$\langle 2(n+1)(n-1)\rangle\ = \langle 2(n^2-1)\rangle\ .$$

The bound for step 4 results from direct calculation. Combining these error bounds gives the recursion

$$\phi(n) = \phi(n-1) + 2n^2 + 2n - 4, \tag{10}$$

with the initial condition $\phi(1) = 0$.

This proof strategy yields a valid although far too pessimistic bound for the block-GTH algorithm. Suppose that we eliminate l states instead of 1. By an error analysis similar to [4], the error introduced into the eigenvector p_l in step 2 for the block algorithm is bounded by

$$\langle l^2(3^l - 1)\hat{n}^2 + \mathcal{O}(l^2 3^l \hat{n})\rangle .$$

We denote the error bound by $\psi(\hat{n} - 1)$. With the initial condition $\psi(1) = \phi(l)$, we have

$$\begin{aligned}\psi(\hat{n}) &= \frac{l^2}{3}(3^l - 1)\hat{n}^3 + \mathcal{O}((l3^l + l^2)\hat{n}^2) \\ &= \frac{3^l - 1}{3l}n^3 + \mathcal{O}(\frac{3^l}{l}n^2 + n^2) ,\end{aligned}$$

which is not tight for large block size l because of the exponential term. Therefore, a more delicate analysis is necessary.

We obtain a polynomial error bound for the computed eigenvector in GTH by repeatedly applying Tweedie's theorem to the generators resulting from eliminating one state only. This suggests reconsidering the error bound for one iteration in step 2 of block-GTH by accumulating error bounds when one state is eliminated instead of calculating the error bound for the eigenvector after eliminating l states. Our next task is to define the generators that are implicit in the intermediate steps of the block-GTH algorithm and determine the error bound for their off-diagonal entries. The proof strategy for block-GTH is shown in Figure 2.

Suppose that we have a given generator G_0 of order n. We need to determine an error bound $\psi_l(\hat{n})$ with polynomial growth in each iteration in step 2, where $\hat{n} = \lceil n/l \rceil$. Let $\tilde{G}_0 = G_0$. The generator $\tilde{G}_l$ is defined by the block-GTH algorithm, so we need to define the following generators.

$\mathcal{G}_k$, the generator of size $n-k$ that has the same eigenvector as $\tilde{G}_{k-1}$, for $k = 1, \ldots, l$.

$\tilde{G}_k$, the computed generator of size $n - k$, $k = 1, \ldots, l - 1$.

Since the block-GTH algorithm is closely related to GTH, it is useful to define $\mathcal{G}_k$ to be the generator resulting from eliminating the first state from $\tilde{G}_{k-1}$ by GTH using exact arithmetic. Note that the definition of $\mathcal{G}_1$ is the same for GTH and block-GTH (since G_0 is the same for both), but generators $\mathcal{G}_2, \ldots, \mathcal{G}_l$ differ for the two algorithms because they are defined in terms of the computed quantities $\tilde{G}_1, \ldots, \tilde{G}_{l-1}$. Our goal, then, is to study $\tilde{G}_k$ for the block-GTH algorithm and show that its eigenvector is close to the eigenvector of $\mathcal{G}_k$.

Throughout the following paragraphs, index k will vary between 1 and $l - 1$. A scalar with superscript k will denote a result after eliminating k states from G_0. An operator with a "hat" uses floating-point arithmetic. Since the error bound strongly depends on the specific computational formulas, we analyze the error by strictly following the order of operations in the block-GTH algorithm. We will derive an error bound for block-GTH-II. The bound for block-GTH-I is derived in the appendix.

Suppose that we partition the generator $\tilde{G}_k$ (including G_0) as

$$\tilde{G}_k = \begin{bmatrix} A_k & B_k \\ C_k & D_k \end{bmatrix} ,$$

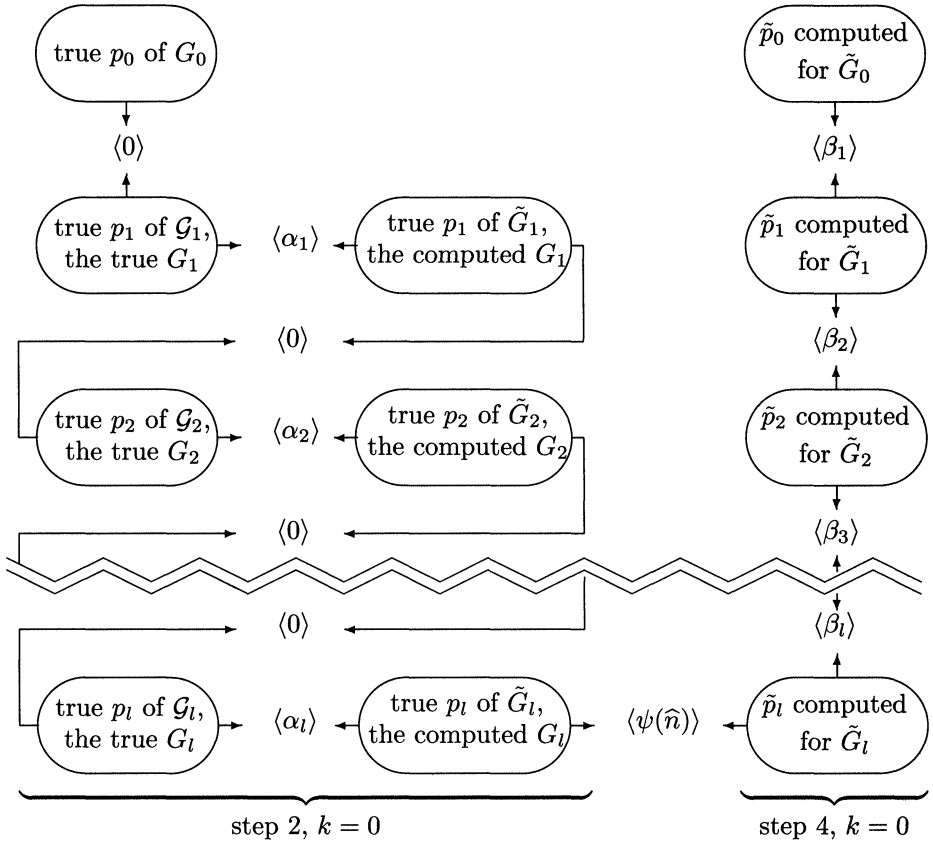

FIG. 2.

where A_k is of order $l - k$. Note that we define the order of A_k in a different way from the partition (2) for block-GTH in §2. It is convenient to index the elements of A_k as

$$\begin{bmatrix} a^k_{k+1,k+1} & \cdots & a^k_{k+1,l} \\ \vdots & \ddots & \vdots \\ a^k_{l,k+1} & \cdots & a^k_{ll} \end{bmatrix}. \tag{11}$$

Let $h^0 = B_0 e$. In step 2.2, we apply GTH to the matrix

$$H_0 = \begin{bmatrix} A_0 & h^0 \\ 0 & 0 \end{bmatrix}$$

using the following computations: for $k < i, j \leq l$,

$$s_k = -a^{k-1}_{kk} = a^{k-1}_{k,k+1} \mathbin{\hat{+}} \cdots \mathbin{\hat{+}} a^{k-1}_{kl} \mathbin{\hat{+}} h^{k-1}_k , \tag{12}$$

$$a^k_{ij} = a^{k-1}_{ij} \mathbin{\hat{+}} a^{k-1}_{ik} \mathbin{\hat{*}} (a^{k-1}_{kj} \mathbin{\hat{/}} s_k) , \tag{13}$$

$$h^k_i = h^{k-1}_i \mathbin{\hat{+}} a^{k-1}_{ik} \mathbin{\hat{*}} (h^{k-1}_k \mathbin{\hat{/}} s_k) , \tag{14}$$

and $s_l = -a^{l-1}_{ll} = h^{l-1}_l$.

The next task is to determine error bounds for the off-diagonal entries of A_k relative to corresponding entries of $\mathcal{G}_k$. Let $\langle \gamma_k \rangle$ be the error bound for s_k. From (12) we see that this bound arises from the error bound for h_k^{k-1} plus $l-k$ additions, and from (13) we conclude that the error for the off-diagonal entries of A_k is bounded by $\langle \gamma_k + 3 \rangle$ (since the entries in A_{k-1} have no error relative to $\mathcal{G}_k$). We determine γ_k by studying the h_i^k. Initially, h^0 has a component-wise error bound $\langle n-l-1 \rangle$ coming from $n-l-1$ additions. From (12), we have an error bound $\langle n-2 \rangle$ for s_1 relative to the sum of the off-diagonal entries of the first row of G_0. From (14), we have

$$\begin{aligned} h_i^1 &= h_i^0 \mathbin{\widehat{+}} a_{i1}^0 \mathbin{\widehat{*}} (h_1^0 \mathbin{\widehat{/}} s_1) \\ &= \widehat{\sum_j} b_{ij}^0 \mathbin{\widehat{+}} a_{i1}^0 \mathbin{\widehat{*}} \left[\left(\widehat{\sum_j} b_{1j}^0 \right) \mathbin{\widehat{/}} s_1 \right], \end{aligned}$$

where the summation is taken over the $n-l$ column indices of B_0. By using the rules of floating-point operations, we have

$$\begin{aligned} h_i^1 &= \langle n-l-1 \rangle \sum_j b_{ij}^0 \mathbin{\widehat{+}} \langle (n-l-1)+2 \rangle \sum_j a_{i1}^0 * (b_{1j}^0 / s_1) \\ &= \langle n-l-1 \rangle \sum_j b_{ij}^0 \mathbin{\widehat{+}} \langle (n-l-1)+4 \rangle \sum_j a_{i1}^0 \mathbin{\widehat{*}} (b_{1j}^0 \mathbin{\widehat{/}} s_1) \\ &= \langle (n-l-1)+5 \rangle \sum_j [b_{ij}^0 + a_{i1}^0 \mathbin{\widehat{*}} (b_{1j}^0 \mathbin{\widehat{/}} s_1)] \\ &= \langle (n-l-1)+6 \rangle \sum_j [b_{ij}^0 \mathbin{\widehat{+}} a_{i1}^0 \mathbin{\widehat{*}} (b_{1j}^0 \mathbin{\widehat{/}} s_1)] . \end{aligned} \tag{15}$$

Let us define the entries of B_1 by

$$b_{ij}^1 = b_{ij}^0 \mathbin{\widehat{+}} a_{i1}^0 \mathbin{\widehat{*}} (b_{1j}^0 \mathbin{\widehat{/}} s_1) . \tag{16}$$

Then we obtain

$$h_i^1 = \langle (n-l-1)+6 \rangle \sum_j b_{ij}^1 .$$

For $i=2$, the above equation and (12) imply that s_2, the (1,1) entry of $\tilde{G}_1$, has error bound $\langle (n-3)+6 \rangle$ relative to the (1,1) entry of $\mathcal{G}_1$.

For the next update, we have

$$\begin{aligned} h_i^2 &= h_i^1 \mathbin{\widehat{+}} a_{i2}^1 \mathbin{\widehat{*}} (h_2^1 \mathbin{\widehat{/}} s_2) \\ &= \langle (n-l-1)+6 \rangle \sum_j b_{ij}^1 \mathbin{\widehat{+}} \langle (n-l-1)+8 \rangle \sum_j a_{i2}^1 * (b_{2j}^1 / s_2) , \end{aligned}$$

which is similar to the first line of (15), so we can define B_2 and directly derive

$$h_i^2 = \langle (n-l-1)+12 \rangle \sum_j b_{ij}^2 .$$

Then s_3, the (1,1) entry of $\tilde{G}_2$, has error bound $\langle (n-4)+2*6 \rangle$ relative to the (1,1) entry of $\mathcal{G}_2$. Continuing this process, we define matrix B_k as

$$\begin{bmatrix} b_{k+1,1}^k & \cdots & b_{k+1,n-l}^k \\ \vdots & \ddots & \vdots \\ b_{l1}^k & \cdots & b_{n-l,n-l}^k \end{bmatrix} , \tag{17}$$

and we have an error bound for s_k (including s_l) as $\langle (n-k-1)+6(k-1)\rangle \equiv \gamma_k$. Therefore, we have shown that there is an error bound $\langle \gamma_k + 3\rangle$ for each off-diagonal entry of A_k and for each entry of B_k, relative to the corresponding element in $\mathcal{G}_k$. The matrix C_0 has been updated in a similar way during step 2.2, and by defining

$$C_k = \begin{bmatrix} c^k_{1,k+1} & \cdots & c^k_{1l} \\ \vdots & \ddots & \vdots \\ c^k_{n-l,k+1} & \cdots & c^k_{n-l,l} \end{bmatrix} \tag{18}$$

to be the matrix after k updates, it is easy to see that the entry-wise relative error between C_k and the corresponding entries of $\mathcal{G}_k$ is also bounded by $\langle \gamma_k + 3\rangle$.

Next, we consider the matrix D_k. After finishing step 2.2, we have an LU factorization of the matrix A_0 as

$$A_0 = L\,U = \begin{bmatrix} -s_1 & & & \\ a^0_{21} & -s_2 & & 0 \\ \vdots & \vdots & \ddots & \\ a^0_{l1} & a^1_{l2} & \cdots & -s_l \end{bmatrix} \begin{bmatrix} 1 & (-a^0_{12} \mathbin{\widehat{/}} s_1) & \cdots & (-a^0_{1l} \mathbin{\widehat{/}} s_1) \\ & 1 & \cdots & (-a^1_{2l} \mathbin{\widehat{/}} s_2) \\ 0 & & \ddots & \vdots \\ & & & 1 \end{bmatrix}.$$

The computations for $\tilde{G}_l$ can be expressed as

$$\tilde{G}_l = D_0 - (C_0 U^{-1})(L^{-1}B_0)\,.$$

Now, let us focus on the computation of the entry (i,j) of $\tilde{G}_l$, where $1 \le i,j \le n-l$. We need to compute $(L^{-1}B_0)$ first. Let b be the jth column of the matrix B_0, and let x be the jth column of the matrix $(L^{-1}B_0)$. Then the solution to the triangular linear system $Lx = b$ is computed as

$$\begin{aligned} x_1 &= b^0_{1j} \mathbin{\widehat{/}} (-s_1)\,, \\ x_m &= (b^0_{mj} \mathbin{\widehat{-}} a^0_{m1} \mathbin{\widehat{*}} x_1 \mathbin{\widehat{-}} \cdots \mathbin{\widehat{-}} a^{m-2}_{m,m-1} \mathbin{\widehat{*}} x_{m-1}) \mathbin{\widehat{/}} (-s_m) \\ &= b^{m-1}_{mj} \mathbin{\widehat{/}} (-s_m), \quad m = 2,\ldots,l\,. \end{aligned}$$

To obtain $\tilde{G}_l$ in block-GTH-II, we have

$$\begin{aligned} (i,j) \text{ entry of } \tilde{G}_l &= d^0_{ij} \mathbin{\widehat{-}} [\text{row } i \text{ of } (C_0U^{-1})] \mathbin{\widehat{*}} [\text{column } j \text{ of } (L^{-1}B_0)] \\ &= d^0_{ij} \mathbin{\widehat{+}} [c^0_{i1} \mathbin{\widehat{*}} (b^0_{1j} \mathbin{\widehat{/}} s_1) \mathbin{\widehat{+}} \cdots \mathbin{\widehat{+}} c^{l-1}_{il} \mathbin{\widehat{*}} (b^{l-1}_{lj} \mathbin{\widehat{/}} s_l)] \\ &= \langle l\rangle\, [d^0_{ij} + c^0_{i1} \mathbin{\widehat{*}} (b^0_{1j} \mathbin{\widehat{/}} s_1) + \cdots + c^{l-1}_{il} \mathbin{\widehat{*}} (b^{l-1}_{lj} \mathbin{\widehat{/}} s_l)]\,. \end{aligned} \tag{19}$$

If we define

$$d^k_{ij} = d^{k-1}_{ij} + c^{k-1}_{ik} \mathbin{\widehat{*}} (b^{k-1}_{kj} \mathbin{\widehat{/}} s_k)\,, \tag{20}$$

then (19) becomes

$$\begin{aligned} (i,j) \text{ entry of } \tilde{G}_l &= \langle l\rangle\, [d^{l-1}_{ij} + c^{l-1}_{il} \mathbin{\widehat{*}} (b^{l-1}_{lj} \mathbin{\widehat{/}} s_l)] \\ &= \langle l+2\rangle\, [d^{l-1}_{ij} + c^{l-1}_{il} * (b^{l-1}_{lj}/s_l)] \\ &= \langle \gamma_l + l + 2\rangle \ (i,j) \text{ entry of } \mathcal{G}_l\,, \end{aligned} \tag{21}$$

where the γ_l comes from the error bound for s_l.

Note that (20) also gives us the definition of D_k as

$$\begin{bmatrix} d_{11}^k & \cdots & d_{1,n-l}^k \\ \vdots & \ddots & \vdots \\ d_{n-l,1}^k & \cdots & d_{n-l,n-l}^k \end{bmatrix}, \tag{22}$$

with entry-wise error bound $\langle \gamma_k + 2 \rangle$. From (11), (17), (18), and (22), we have defined $\tilde{G}_k$ and shown that its off-diagonal entry-wise error can be bounded by $\langle \gamma_k + 3 \rangle$. By applying Tweedie's theorem, for $k = 1, \ldots, l-1$, the component-wise error bound for the eigenvector of $\tilde{G}_k$ relative to the eigenvector of $\mathcal{G}_k$ is $\langle \alpha_k \rangle = \langle 2(n-k)(\gamma_k + 3) \rangle$.

From (21), we have the off-diagonal entry-wise error bound $\langle \gamma_l + l + 2 \rangle$ for $\tilde{G}_l$ computed by block-GTH-II. Since $\tilde{G}_l$ is of order l, by applying Tweedie's theorem, we have $\langle \alpha_l \rangle = \langle 2(n-l)(\gamma_l + l + 2) \rangle$ as a component-wise error bound between the eigenvectors of $\tilde{G}_l$ and $\mathcal{G}_l$. Therefore, the error bound accumulated in one iteration of the factorization phase of block-GTH-II is bounded by

$$\begin{aligned} \sum_{k=1}^{l} \alpha_k &= \left[\sum_{k=1}^{l-1} 2(n-k)(\gamma_k + 3) \right] + 2(n-l)(\gamma_l + l + 2) \\ &= \frac{1}{3}[6ln^2 + (12l^2 - 6l - 6)n - (10l^3 + 9l^2 - 13l)]. \end{aligned} \tag{23}$$

As for the backsubstitution phase, we can also express the computations in the form

$$q = p_l(C_0 U^{-1}) ,$$

where p_l is the computed eigenvector of $\tilde{G}_l$. Note that $p_0 = [qL^{-1} \;\; p_l]$ is the eigenvector of G_0.

The type II process uses the same backsubstitution process as the GTH algorithm. Thus the component-wise error bound $\langle \beta_k \rangle$ between the computed vectors $\tilde{p}_k$ and $\tilde{p}_{k-1}$ in block-GTH-II is $\langle \gamma_k + (n-k+1) \rangle$, where the bound $\langle n-k+1 \rangle$ comes from one multiplication, $n-k-1$ additions, and one division. The error bound for one iteration in step 4 of block-GTH-II is

$$\begin{aligned} \sum_{k=1}^{l} \beta_k &= \sum_{k=1}^{l} [\gamma_k + n - k + 1] \\ &= 2ln + 2(l^2 - 2l) . \end{aligned} \tag{24}$$

Combining (23) and (24), we have established a polynomial error bound for block-GTH-II: we have

$$\begin{aligned} \widehat{\psi}_l(n) &= \psi_l(\widehat{n}) \\ &= \frac{1}{3}\left[2n^3 + \left(9l - \frac{3}{l} \right) n^2 - (3l^2 + 3l + 2)n - (6l^3 - 9l^2 + 3l) \right], \end{aligned} \tag{25}$$

with $\psi_l(1) = \phi(l)$.

Therefore, we have the following analogue to Theorem 1.

THEOREM 2. *For any stochastic matrix P of order n with stationary vector π, the accuracy of the left eigenvector $\widehat{\pi}$ computed by the block-GTH algorithms* I *and* II *with block size l $(1 \le l \le n)$ using floating-point arithmetic is characterized by*

$$\widehat{\pi}_i = \langle 2\widehat{\psi}_l(n) + n \rangle \, \pi_i, \quad i = 1, \ldots, n,$$

where $\widehat{\psi}_l(n)$ is defined by (25) *or* (A.31). *Furthermore, if $(2\widehat{\psi}_l(n) + n)\mathbf{u} \le .1$, then*

$$\frac{|\widehat{\pi}_i - \pi_i|}{\pi_i} \le 1.06(2\widehat{\psi}_l(n) + n)\mathbf{u} \, .$$

Note that for $1 \le l \le n$, $\widehat{\psi}_l(n) \ge \phi(n)$, with equality holding at $l = 1$ and n. However, we can sharpen $\widehat{\psi}_l(n)$ in several places. For example, if l does not evenly divide n, then the initial value $\psi_l(1)$ is less than $\phi(l)$. For $l \le n/2$, the error bound $\langle \gamma_k \rangle$ for s_k can be reduced to $\langle n - l \rangle$, which is independent of k, by summing (12) from left to right.

4. Performance of the block-GTH algorithm. In this section, we consider the implementation of the block-GTH algorithm and discuss some numerical results comparing the performance of GTH and block-GTH. Experiments were performed with single precision IEEE arithmetic on a DECstation 3100 (DEC) and a SUN SPARCstation 2 (SUN), each with a 64k byte cache memory; a Convex C3820 (Convex); and a Cray Y-MP4D/2/16 (CRAY).

We implemented the algorithm using standard software as much as possible, primarily the basic linear algebra subroutines from the BLAS collection [1]. On the SUN and DEC machines, we used Fortran versions of the BLAS; on the Convex and Cray, we used the manufacturer-supplied versions. The standard Fortran compilers (f77, f77, fc, and cf77) were used with default levels of optimization. We summarize the machine configurations and computing environments in Table 1.

TABLE 1

Machine	Operating system	Processor	Compiler	Word length
DEC	ULTRIX V4.1	1	f77 V3.2	32-bit
SUN	SUN4c_OS413A	1	f77 SC2.0.1	32-bit
Convex	ConvexOS 10.1	1	fc version 7.0	64-bit
CRAY	UNICOS 7.0.4.3	1	cf77 Release 6.0	64-bit

The principal implementation task is SGTHLU, which computes the LU factorization of the matrix A_k for type I block-GTH ($[A_k^T \; C_k^T]^T$ for type II). Note that this subroutine performs the standard GTH algorithm when the block size l equals the order n of the original generator.

The major time-consuming modules of the algorithm are shown in Table 2.

TABLE 2

Step	Routine	Source	Function
2.2	SGTHLU	Uses Level-1 BLAS	Apply GTH
2.3	STRSM	Level-3 BLAS	Update B_k
2.3	SGEMM	Level-3 BLAS	Update D_k

Since the block-GTH algorithm is a variant of Gaussian elimination, the complexity is of order n^3 and the factorization phase dominates the computational time. In

our implementation, the cost of factorization is

$$\frac{1}{6}(4n^3 + 3n^2 - 7n) ,$$

independent of the block size l.

For our numerical experiments, we defined the generator of order n to be a circulant matrix

$$G = \begin{bmatrix} \alpha & \beta & \cdots & \gamma & \gamma \\ \gamma & \alpha & \ddots & & \gamma, \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \gamma & & \ddots & \alpha & \beta \\ \beta & \gamma & \cdots & \gamma & \alpha \end{bmatrix} ,$$

with

$$\begin{aligned} \alpha &= -0.01, \\ \beta &= 0.0002, \\ \gamma &= 0.0098/(n-2) . \end{aligned}$$

It can be shown that this generator has a simple eigenvalue 0 with left eigenvector

$$\pi = (1/n)\, e .$$

We tested only the type II block-GTH algorithm.

First we examined the accuracy of the block-GTH algorithm. We set $n = 400$ and varied the block size as $l = 1, 2, \ldots, 49, 50$ and then $l = 60, 80, \ldots, 400$. Table 3 shows the resulting rounding errors. As predicted by the theory, the errors do not have strong dependence on block size: the errors produced by the block-GTH algorithm varied between .87 and 1.5 times the errors produced by the GTH algorithm.

TABLE 3
Rounding errors resulting from use of the block-GTH algorithm with different block sizes for a generator of order 400.

	Block size	Average difference	Largest difference	Largest rela. difference
	20	3.5700e−09	1.8000e−08	7.2000e−06
	40	3.8925e−09	1.8000e−08	7.2000e−06
D	60	4.2275e−09	1.7000e−08	6.8000e−06
E	80	3.9475e−09	2.2000e−08	8.8000e−06
C	100	5.1225e−09	2.0000e−08	8.0000e−06
&	120	3.4000e−09	1.9000e−08	7.6000e−06
S	140	3.3600e−09	1.8000e−08	7.2000e−06
U	160	3.0025e−09	1.6000e−08	6.4000e−06
N	180	3.2050e−09	1.9000e−08	7.6000e−06
	200	3.1250e−09	1.9000e−08	7.6000e−06
	400	3.4175e−09	1.7000e−08	6.8000e−06

Figure 3 shows the total CPU times for our implementation of the factorization phase of the block-GTH algorithm and the time taken by its three dominant subroutines (SGTHLU, STRSM, and SGEMM) as the block size changes. Although the

number of operations is independent of block size, the total execution time on the SUN and DEC machines had a significant drop around block size 40, due to efficient utilization of the cache memory. This timing phenomenon is primarily due to the behavior of SGEMM. This routine computes the matrix $\tilde{G}_l$ column-wise by computing linear combinations of columns of the matrix C_k. Since Fortran stores matrices column by column, we can expect that SGEMM will perform best if for all k, the matrix C_k and some portion of D_k and B_k fit in the cache memory.

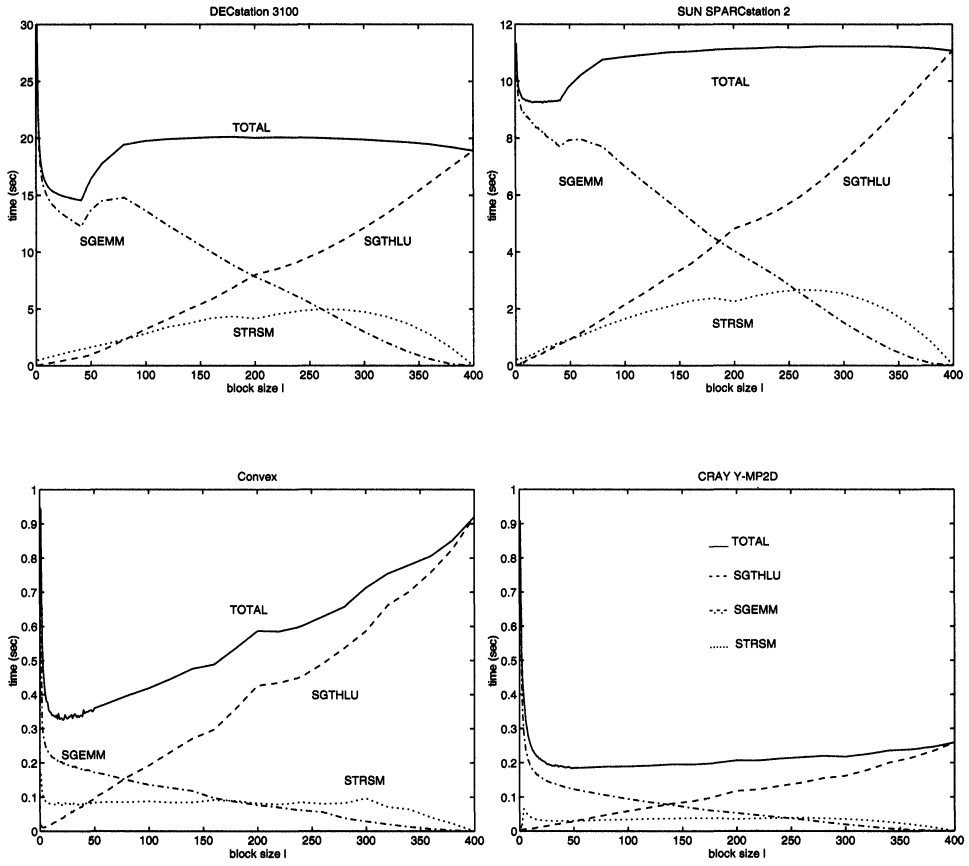

FIG. 3. *Block-GTH time as a function of block size for a generator of order* 400.

The biggest of the C matrices is C_0, of size $(n-l)l$. Each column of B_k has size l, and the D_0 matrix has columns of length $n-l$. Therefore, we predict that the optimal block size should occur at the largest integer l satisfying

$$(n-l)\,l + l + (n-l) \leq \frac{\text{cache memory size}}{\text{word size}}\,.$$

For the DEC and SUN, the cache capacity is 64k/4=16k words. The actual optimal block size also depends on other features of the machine architecture such as page size, cache line size, etc.

Next, we ran numerical experiments on generators of different orders. We varied the block size in increments of 1 until well past the predicted optimum, and then in

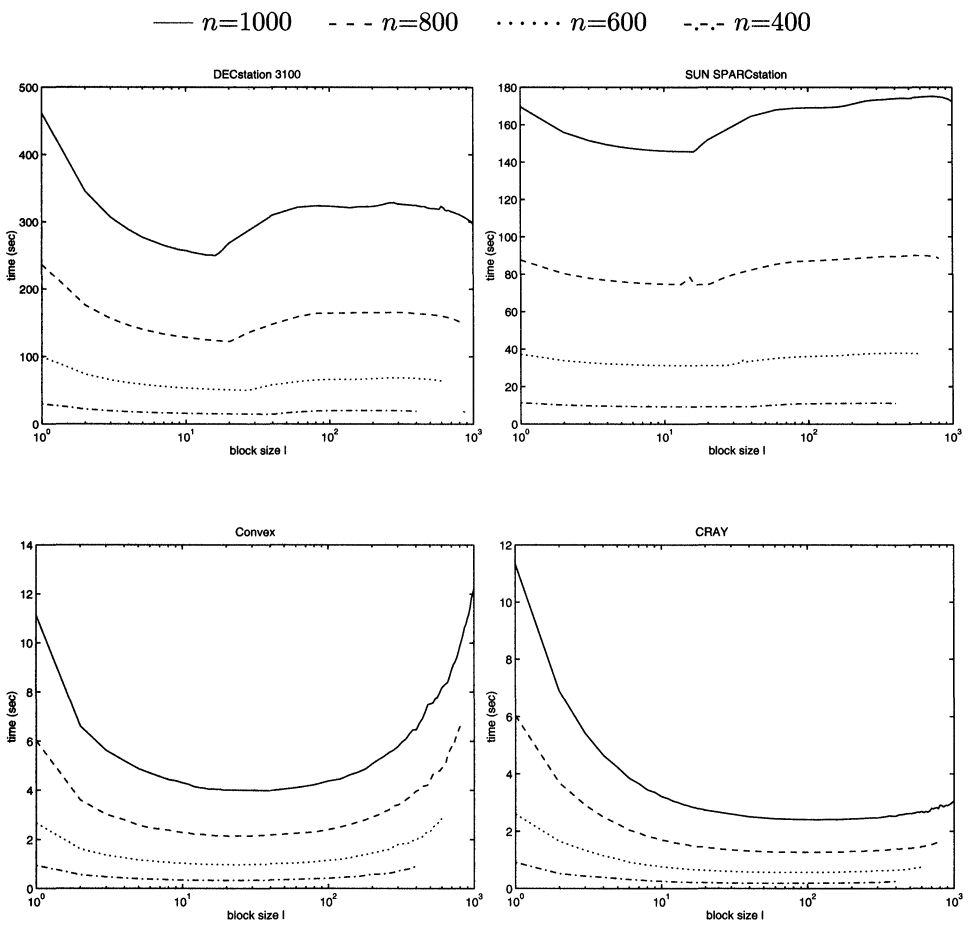

FIG. 4. *Block-GTH time as a function of block size for generators of order* 400, 600, 800, *and* 1000.

increments of 20. Figure 4 shows the total time as a function of block size. Table 4 gives the timings, predicted and actual optimal block size, and the speedup, defined by

$$\text{speedup} = \frac{\text{the time for the GTH algorithm}}{\text{the best time for the block-GTH algorithm}}.$$

On the SUN, the timing gain for the block-GTH algorithm over the standard GTH algorithm is 18–20%, while it is 19–30% on the DEC machine. The predictions of optimal block sizes were quite accurate for the DEC, but were overestimates for the SUN. Using the predicted optimal block size on the SUN gave timing gains of 16–19%, not much less than the actual optimal.

On the Convex, a block size of 21, independent of the order of the matrix, performs quite well, while on the Cray, the performance varies only slightly for a large range of block sizes, with the optimal size about 12% of n.

Further timing gains could be achieved by using level-2 or level-3 BLAS in the implementation of SGTHLU.

TABLE 4
Timings, optimal block size, and speedup for generators of order $n = 400, 600, 800,$ *and* 1000.

	Problem size n	GTH time (sec)	Block Size Predicted optimal	Block Size Actual optimal	Block-GTH Time (sec)	Block-GTH Mflop rate	Speedup
DEC	400	18.91	44	41	14.55	2.9	1.30
	600	64.18	28	27	50.11	2.9	1.28
	800	151.72	20	20	122.49	2.8	1.24
	1000	296.83	16	16	249.94	2.7	1.19
SUN	400	11.07	44	25	9.24	4.6	1.20
	600	37.35	28	19	31.23	4.6	1.20
	800	88.56	20	17	74.45	4.6	1.19
	1000	172.25	16	16	145.57	4.6	1.18
Convex	400	0.92		21	0.33	130	2.82
	600	2.85		24	0.96	150	2.95
	800	6.61		21	2.13	160	3.10
	1000	12.24		40	3.98	168	3.07
CRAY	400	0.26		48	0.18	238	1.41
	600	0.75		60	0.56	258	1.32
	800	1.64		100	1.27	269	1.29
	1000	3.06		120	2.41	277	1.27

5. Conclusions. It is necessary to use block algorithms in order to attain good utilization of vector processors and cache memory. In this work we have shown that the GTH algorithm has a block implementation that can achieve a considerable increase in efficiency without sacrificing accuracy. Future work will deal with the parallel implementation of the algorithm.

Appendix A. Derivation of the error bound for the block-GTH-I algorithm. The block-GTH-I differs from block-GTH-II only at step 2.3 for computing $\tilde{G}_l$ and in the backsubstitution phase, so the definitions for A_k and B_k in (11) and (17) remain the same for $k = 1, \ldots, l-1$. Thus we have an error bound $\langle \gamma_k + 3 \rangle$ for each off-diagonal entry of A_k and for each entry of B_k.

The computations of $\tilde{G}_l$ for block-GTH-I can be expressed as

$$\tilde{G}_l = D_0 - C_0(U^{-1}(L^{-1}B_0)) .$$

We save the original matrix C_0, so there is one more linear system to solve for computing $U^{-1}x$. Let y be the jth column of the matrix $U^{-1}(L^{-1}B_0)$. Applying backsubstitution to the triangular linear system $U\,y = x$, we have

$$\begin{aligned} y_l &= x_l \ , \\ y_m &= x_m \mathbin{\hat{+}} (a_{ml}^{m-1} \mathbin{\hat{/}} s_m) \mathbin{\hat{*}} y_l \mathbin{\hat{+}} \cdots \mathbin{\hat{+}} (a_{m,m+1}^{m-1} \mathbin{\hat{/}} s_m) \mathbin{\hat{*}} y_{m+1}, \\ & \qquad m = l-1, \ldots, 1 \ . \end{aligned}$$

Note that all x_m are negative, so all y_m are negative and there is no cancellation. To obtain $\tilde{G}_l$, we have

$$\begin{aligned} (i,j) \text{ entry of } \tilde{G}_l &= d_{ij}^0 \mathbin{\hat{-}} [\text{row } i \text{ of } C_0] \mathbin{\hat{*}} [\text{column } j \text{ of } U^{-1}(L^{-1}B_0)] \\ &= d_{ij}^0 \mathbin{\hat{-}} [c_{i1}^0 \mathbin{\hat{*}} y_1 \mathbin{\hat{+}} c_{i2}^0 \mathbin{\hat{*}} y_2 \mathbin{\hat{+}} \cdots \mathbin{\hat{+}} c_{il}^0 \mathbin{\hat{*}} y_l] \\ &= \langle l+1 \rangle \, [d_{ij}^0 - c_{i1}^0 * y_1 - c_{i2}^0 * y_2 - \cdots - c_{il}^0 * y_l] \ . \end{aligned} \tag{A.26}$$

By expanding y_1, we have

(i,j) entry of $\tilde{G}_l$

$$
\begin{aligned}
&= \langle l+1\rangle \{d_{ij}^0 - c_{i1}^0 * [(-b_{1j}^0 \mathbin{\widehat{/}} s_1) \mathbin{\widehat{+}} (a_{1l}^0 \mathbin{\widehat{/}} s_1) \mathbin{\widehat{*}} y_l \mathbin{\widehat{+}} \cdots \mathbin{\widehat{+}} (a_{12}^0 \mathbin{\widehat{/}} s_1) \mathbin{\widehat{*}} y_2] \\
&\qquad -c_{il}^0 * y_l - \cdots - c_{i2}^0 * y_2\} \\
&= \langle l+1\rangle \{d_{ij}^0 + \langle l\rangle [c_{i1}^0 * (b_{1j}^0 \mathbin{\widehat{/}} s_1) - c_{i1}^0 * (a_{1l}^0 \mathbin{\widehat{/}} s_1) * y_l - \cdots \\
&\qquad -c_{i1}^0 * (a_{12}^0 \mathbin{\widehat{/}} s_1) * y_2] - c_{il}^0 * y_l - \cdots - c_{i2}^0 * y_2\} \\
&= \langle (l+1)+l\rangle \{d_{ij}^0 + c_{i1}^0 * (a_{1l}^0 \mathbin{\widehat{/}} s_1) - [c_{il}^0 + c_{i1}^0 * (b_{1j}^0 \mathbin{\widehat{/}} s_1)] * y_l - \cdots \\
&\qquad -[c_{i2}^0 + c_{i1}^0 * (a_{12}^0 \mathbin{\widehat{/}} s_1)] * y_2\} .
\end{aligned}
$$

The updated matrix C_1 is not explicitly present, but we can define

$$
c_{im}^1 = c_{im}^0 + c_{i1}^0 * (a_{1m}^0 \mathbin{\widehat{/}} s_1), \quad m = 2, \ldots, l . \tag{A.27}
$$

From (20) and (A.27), the update becomes

$$
\text{entry of } \tilde{G}_l = \langle (l+1)+l\rangle \, [d_{ij}^1 + c_{il}^1 * y_l + \cdots + c_{i2}^1 * y_2] ,
$$

which is similar to (A.26). Therefore, we can expand $y_m, m = 2, \ldots, l-1$, one by one and repeat the same process to obtain

$$
\begin{aligned}
&(i,j) \text{ entry of } \tilde{G}_l \\
&= \langle (l+1)+l+\cdots+2\rangle \, [d_{ij}^{l-1} + c_{il}^{l-1} * y_l] \\
&= \left\langle \sum_{m=2}^{l+1} m \right\rangle [d_{ij}^{l-1} + c_{il}^{l-1} * (b_{lj}^{l-1} \mathbin{\widehat{/}} s_l)] \\
&= \left\langle 1 + \sum_{m=2}^{l+1} m \right\rangle [d_{ij}^{l-1} + c_{il}^{l-1} * (b_{lj}^{l-1}/s_l)] \\
&= \left\langle \gamma_l + \sum_{m=1}^{l+1} m \right\rangle \; (i,j) \text{ entry of } \mathcal{G}_l ,
\end{aligned} \tag{A.28}
$$

where the γ_l comes from s_l.

Note that from (A.27), the matrix C_k defined in (18) has an entry-wise error bound $\langle \gamma_k + 1\rangle$. Again, from (11), (17), (18), and (22), we have defined $\tilde{G}_k$ and shown that its off-diagonal entry-wise error can be bounded by $\langle \gamma_l + 3\rangle$. From (A.28), the entry-wise error bound for $\tilde{G}_l$ computed by block-GTH-I relative to $\mathcal{G}_l$ is $\langle \gamma_l + \sum_{m=1}^{l+1} m\rangle$. By applying Tweedie's theorem, the error accumulated for one iteration in step 2 of block-GTH-I is bounded by

$$
\begin{aligned}
\sum_{k=1}^{l} \alpha_k &= \left\lceil \sum_{k=1}^{l-1} 2(n-k)(\gamma_k + 3) \right\rceil + 2(n-l) \left(\gamma_l + \sum_{m=1}^{l+1} m \right) \\
&= \frac{1}{3}[6ln^2 + (15l^2 - 3l - 12)n - (13l^3 + 12l^2 - 19l)] .
\end{aligned} \tag{A.29}
$$

For the backsubstitution phase, let $p_l = [p_{l1} \cdots p_{l,n-l}]$ be the computed eigenvector of $\tilde{G}_l$. Then we have the vector

$$
\left[\widehat{\sum_i} p_{li} \mathbin{\widehat{*}} c_{i1}^0 \;\; \cdots \;\; \widehat{\sum_i} p_{li} \mathbin{\widehat{*}} c_{il}^0 \right]
$$

resulting from computing $(p_l C)$ first. Then the solution to the triangular linear system $qU = p_l C$ is

$$q_1 = \widehat{\sum_i} p_{li} \,\hat{*}\, c_{i1}^0 ,$$

$$q_k = \widehat{\sum_i} p_{li} \,\hat{*}\, c_{ik}^0 \,\hat{+}\, (a_{1k}^0 \,\hat{/}\, s_1) \,\hat{*}\, q_1 \,\hat{+}\, \cdots \,\hat{+}\, (a_{k-1,k}^{k-2} \,\hat{/}\, s_{k-1}) \,\hat{*}\, q_{k-1} ,$$

$$k = 2, \ldots, l .$$

From the first equation, we have $q_1 = \delta_1 \sum_i p_{li} * c_{i1}^0$, with $\delta_1 = n - l$. To find an error bound for q_k in the second equation, suppose that we have $q_m = \langle \delta_m \rangle \sum_i p_{li} * c_{im}^{m-1}$. We know that δ_m is an increasing function of m, so

$$\begin{aligned} q_k = \langle n-l \rangle \sum_i p_{li} * c_{ik}^0 \,\hat{+}\, \langle \delta_1 + 1 \rangle \sum_i [(a_{1k}^0 \,\hat{/}\, s_1) * p_{li} * c_{i1}^0] \\ \hat{+}\, \langle \delta_2 + 1 \rangle \sum_i [p_{li} * (a_{2k}^1 \,\hat{/}\, s_2) * c_{i2}^1] \,\hat{+}\, \cdots \\ \hat{+}\, \langle \delta_{k-1} + 1 \rangle \sum_i [p_{li} * (a_{k-1,k}^{k-2} \,\hat{/}\, s_{k-1}) * c_{i,k-1}^{k-2}] . \end{aligned}$$

Note that the floating-point additions are carried out from left to right. Using (A.27), we have

$$\begin{aligned} q_k &= \langle \delta_1 + 2 \rangle \left[\sum_i p_{li} * c_{ik}^0 + \sum_i p_{li} * (a_{1k}^0 \,\hat{/}\, s_1) * c_{i1}^0 \right] \\ &\quad \hat{+}\, \langle \delta_2 + 1 \rangle \sum_i [p_{li} * (a_{2k}^1 \,\hat{/}\, s_2) * c_{i2}^1] \,\hat{+}\, \cdots \\ &\quad \hat{+}\, \langle \delta_{k-1} + 1 \rangle \sum_i [p_{li} * (a_{k-1,k}^{k-2} \,\hat{/}\, s_{k-1}) * c_{i,k-1}^{k-2}] \\ &= \langle \delta_1 + 2 \rangle \sum_i p_{li} * c_{ik}^1 \,\hat{+}\, \langle \delta_2 + 1 \rangle \sum_i [p_{li} * (a_{2k}^1 \,\hat{/}\, s_2) * c_{i2}^1] \,\hat{+}\, \cdots \\ &\quad \hat{+}\, \langle \delta_{k-1} + 1 \rangle \sum_i [p_{li} * (a_{k-1,k}^{k-2} \,\hat{/}\, s_{k-1}) * c_{i,k-1}^{k-2}] . \end{aligned}$$

Since all δ_m are integers and $\delta_m + 1 \leq \delta_{m+1}$, we can repeat the same process and obtain

$$q_k = \langle \delta_{k-1} + 2 \rangle \sum_i p_{li} * c_{ik}^{k-1} .$$

Therefore, $\delta_k = \delta_{k-1} + 2$, and the solution to this recursion is $\delta_k = n - l + 2(k-1)$.

By an analysis similar to that for block-GTH-II, we have $\langle \gamma_k + \delta_k + (l - k + 1) \rangle$ as the component-wise error bound $\langle \beta_k \rangle$ for the computed p_k for block-GTH-I. The error bound for one iteration in step 4 of block-GTH-I is

$$\begin{aligned} \sum_{k=1}^{l} \beta_k &= \sum_{k=1}^{l} [\gamma_k + n + k - 1] \\ &= 2ln + 3l^2 - 5l . \end{aligned} \tag{A.30}$$

Finally, combining (A.29) and (A.30), we obtain a polynomially-growing error bound $\widehat{\psi}_l$ for block-GTH-I:

$$\begin{aligned}\widehat{\psi}_l(n) &= \psi_l(\widehat{n}) \\ &= \frac{1}{3}\left[2n^3 + \left(\frac{21l}{2} - \frac{6}{l} + \frac{3}{2}\right)n^2 - \left(\frac{9l^2}{2} + \frac{3l}{2} + 2\right)n - (6l^3 - 6l^2)\right]. \end{aligned} \tag{A.31}$$

This error bound agrees with (25), the error bound for block-GTH-II, in its highest order term.

Acknowledgments. We are grateful for the help of Ron Boisvert, James da Silva, Olafur Gudmundsson, Bill Pugh, and the referees.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[2] J. W. Demmel and N. J. Higham, *Stability of Block Algorithms with Fast Level 3 BLAS*, Tech. report, LAPACK Working Note 22, CS-90-110, University of Tennessee, Knoxville, TN, July 1990.

[3] W. K. Grassmann, M. I. Taksar, and D. P. Heyman, *Regenerative analysis and steady state distributions*, Oper. Res., 33 (1985), pp. 1107–1116.

[4] C. A. O'Cinneide, *Entrywise perturbation theory and error analysis for Markov chains*, Numer. Math., 65 (1993), pp. 109–120.

[5] G. W. Stewart, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

[6] R. L. Tweedie, *Perturbations of countable Markov chains and processes*, Ann. Inst. Statist. Math., 32 (1980), pp. 283–290.