

MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES



3 4456 0287847 4

ORNL/TM-12006

oml

**OAK RIDGE
NATIONAL
LABORATORY**

MARTIN MARIETTA

**Reduction to Condensed Form
for the Eigenvalue Problem on
Distributed Memory Architectures**

Jack Dongarra
Robert A. van de Geijn

OAK RIDGE NATIONAL LABORATORY
CENTRAL RESEARCH LIBRARY
CIRCULATION SECTION
4800N ROOM 175
LIBRARY LOAN COPY
DO NOT TRANSFER TO ANOTHER PERSON

If you wish someone else to see this
report, send in notes with report and
the library will arrange a loan.

UCRL 99540 9-78

MANAGED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

This report has been reproduced exactly from the best available copy.

Available to DOD and DCE contractors from the Office of Scientific and Technical Information, P. O. Box 62, Oak Ridge, TN 37831; prices available from (615) 576-8401, FTS 526 8401.

Available to the public from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161.
NTIS price codes - Report Copy A07 Microfilm A01

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, product, or process disclosed, or represents that it would be accepted by anyone for any liability owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its approval, endorsement, or recommendation by the United States Government, and the use of the name of any agency of the United States Government for purposes of advertising or promoting a product is not authorized. Authors expressed their appreciation to the following individuals for their contribution to the work:

Engineering Physics and Mathematics Division

Mathematical Sciences Section

**REDUCTION TO CONDENSED FORM FOR THE EIGENVALUE PROBLEM
ON DISTRIBUTED MEMORY ARCHITECTURES**

Jack Dongarra†
Robert A. van de Geijn‡

Computer Science Department†
University of Tennessee
Knoxville, TN 37996-1301

Mathematical Sciences Section†
Engineering Physics and Mathematics
P.O. Box 2008, Bldg. 6012
Oak Ridge National Laboratory
Oak Ridge, TN 37831-6367

Computer Science Department‡
University of Texas, Austin
Austin, TX 78712

DATE PUBLISHED: JANUARY 1992

This work was supported in part by the National Science Foundation and Technology Center Cooperative Agreement No. CCR-8809615 and by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy.

Prepared by the
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831
managed by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under Contract No. DE-ACO5-84OR21400



3 4456 0287847 4

CONTENTS

Abstract	1
1. Introduction	1
2. Assumptions and Notation	2
3. Unblocked Algorithms	3
3.1 Sequential Implementation: Hessenberg Reduction	3
3.2 Sequential Implementation: Tridiagonal Reduction	3
3.3 Parallel Implementation: Hessenberg Reduction	4
3.4 Parallel Implementation: Tridiagonal Reduction	5
4. Blocked Algorithms	5
4.1 Sequential Implementation: Blocked Hessenberg Reduction	5
4.2 Sequential Implementation: Blocked Tridiagonal Reduction	6
4.3 Parallel Implementation: Blocked Hessenberg Reduction	6
4.4 Parallel Implementation: Blocked Tridiagonal Reduction	8
5. Experiments	8
5.1 Reduction to Hessenberg Form	9
5.2 Reduction to Tridiagonal Form	10
6. Conclusion	10
Acknowledgements	11
References	11

Reduction to Condensed Form for the Eigenvalue Problem on Distributed Memory Architectures *

Jack J. Dongarra [†] and Robert A. van de Geijn [‡]

December 3, 1991

Abstract

In this paper, we describe a parallel implementation for the reduction of general and symmetric matrices to Hessenberg and tridiagonal form, respectively. The methods are based on LAPACK sequential codes and use a panel-wrapped mapping of matrices to nodes. Results from experiments on the Intel Touchstone Delta are given.

1 Introduction

In this paper, we are concerned with the parallel implementation on distributed memory MIMD parallel computers of the LAPACK routines for performing the reduction to Hessenberg form and the reduction to tridiagonal form. These reductions are an important first step in the computation of the eigenvalues of matrices.

The LAPACK project is an effort to update the classical linear algebra software packages LINPACK and EISPACK to allow more efficient use of shared memory or traditional supercomputers. Efficiency is attained by writing these routines as much as possible in Level 2 and 3 BLAS [5, 6], reducing the ratio of memory accesses to floating point operations executed and allowing for encapsulation of parallel operations on shared memory architectures.

While parallel implementations of algorithms for solving linear systems have been widely studied [4, 9], the reduction to condensed form has not enjoyed the same attention. A parallel unblocked Hessenberg reduction algorithm based on column wrapped storage is given in [10, 11]. In [8], a reduction based on Gaussian transformations is reported. The reduction of symmetric matrices assuming row wrapped and grid wrapped storage is addressed in [2, 3]. Our approach is different in that we start with highly efficient sequential code [7]. Efficiency on each node is attained by

*This work was supported in part by the National Science Foundation Science and Technology Center Cooperative Agreement No. CCR-8809615 and by the Applied Mathematical Science Research Program, Office of Energy Research, U.S. Department of Energy, under Contract DE-AC05-84OR21400.

[†]Dept. of Computer Sciences, Univ. of TN, Knoxville, TN 37996, and Mathematical Sciences Section, ORNL, Oak Ridge, TN 37831, dongarra@cs.utk.edu

[‡]Dept. of Computer Sciences, Univ. of TX, Austin, TX 78712, rvdg@cs.utexas.edu. Most of this work was performed while this author was on leave at the Univ. of TN.

use of Level 1, 2, and 3 BLAS. Communication is through a proposed communication library, the Basic Linear Algebra Communication Subprograms (BLACS) [1], which makes the code portable.

The paper is organized as follows: Assumptions and notation are given in Section 2. As an introduction to the parallel implementation of blocked algorithms, unblocked algorithms and their parallel implementation are given in Section 3. Blocked versions are discussed in Section 4. Results from experiments on the Intel Touchstone Delta system can be found in Section 5. Concluding remarks are given in the final section.

2 Assumptions and Notation

We will assume that our multicomputer consists of p nodes, labeled P_0, \dots, P_{p-1} which are connected by some communication network that allows broadcasting of messages and combining of global data (in the form of global summation).

For our formulae, we adopt the following notation: Scalars, vectors, and matrices are denoted by lower case Greek, lower case, and upper case arabic letters, respectively. The i th element of a vector is denoted by a corresponding greek letter with subscript i (χ_i, η_i, ν_i , and ν_i for vectors x, y, u , and v , respectively). Given a vector x , the vector consisting of its elements i, \dots, j is denoted by $x_{i:j}$. Given matrix A , the submatrix consisting of elements of rows i, \dots, j and columns k, \dots, l is denoted by $[A]_{i:j,k:l}$. If all rows are involved, the notation $[A]_{*,k:l}$ will be used. Superscripts are generally reserved for iteration indices.

We will use the following mapping of matrices to nodes: Given $A \in \mathbf{R}^{n \times n}$ and panel width $m \geq 1$, assume for simplicity that $n = r * m$ and partition

$$A^{(k)} = \begin{pmatrix} A_1^{(k)} & A_2^{(k)} & \dots & A_r^{(k)} \end{pmatrix}$$

where $A_j^{(k)} \in \mathbf{R}^{n \times m}$ is a panel of width m . The *panel-wrapped* storage scheme assigns $A_j^{(k)}$ to node $P_{(j-1) \bmod p}$. I.e., $A_{i+1}, A_{i+p+1}, \dots$ are assigned to P_i . If $m = 1$, the result is the familiar *column-wrapped* storage scheme [9]. For notational convenience, we define $j \in P_i$ to be true if and only if column j of the matrix is assigned to node P_i .

The basic operations utilized by the reduction algorithms are the computation and application of Householder transformations:

Theorem 1 *Given a vector $x \in \mathbf{R}^n$, one can find a vector $u \in \mathbf{R}^n$ and scalar β s.t.*

$$(I - \beta uu^T)x = (\chi_1, \dots, \chi_k, \pm\eta, 0, \dots, 0)^T$$

where $\eta = \|x_{k+1:n}\|_2$.

Indeed, $u = (0, \dots, 0, \chi_{k+1} \mp \eta, \chi_{k+2}, \dots, \chi_n)^T$ and $\beta = 2/u^T u$ will give the desired result. The sign is chosen to correspond to the sign of χ_{k+1} , thereby minimizing roundoff error in the computation of u .

The transformation $I - \beta uu^T$ will subsequently be denoted by $H^{(k)}(x)$, where here the superscript indicates that elements χ_1, \dots, χ_k are not affected. This notation is consistent with the

previous use of superscripts since in the reduction algorithms the Householder transformation computed during the k th iteration has this property. We will also use the pair (u, β) to denote the transformation, i.e., $(u, \beta) = H^{(k)}(x)$ will denote the vector u and scalar β s.t. $H^{(k)}(x) = (I - \beta uu^T)$. Since u and β are not uniquely defined, we will always take u to be normalized so that it has a unit k th element.

3 Unblocked Algorithms

In this section, we explain how simple algorithms for the reductions to Hessenberg and tridiagonal forms for the eigenvalue computation can be implemented on sequential and parallel architectures.

3.1 Sequential Implementation: Hessenberg Reduction

The reduction of matrix $A^{(1)} = A$ to Hessenberg form can be written as $A^{(n-1)}$, where

$$A^{(k+1)} = H^{(k)} A^{(k)} H^{(k)} = H^{(k)} H^{(k-1)} \dots H^{(1)} A^{(1)} H^{(1)} \dots H^{(k-1)} H^{(k)}$$

Here $H^{(k)} = H^{(k)}([A^{(k)}]_{*,k})$. Letting $(u, \beta) = H^{(k)}$,

$$A^{(k+1)} = H^{(k)} A^{(k)} H^{(k)} = A^{(k)} - \beta uv^T - \beta wu^T$$

where

$$v^T = u^T A^{(k)} \quad \text{and} \quad w = A^{(k)} u - \beta(u^T A u) u \tag{1}$$

This yields the following algorithm for reducing a matrix to Hessenberg form:

Algorithm 2 *Hessenberg Reduction*

```

do  $k = 1, \dots, n - 2$ 
  compute  $(u, \beta) = H^{(k)}([A]_{*,k})$ 
   $v^T = u^T A$ 
   $w = Au - \beta(u^T Au)u$ 
  update  $A = A - \beta uv^T - \beta wu^T$ 
enddo
```

3.2 Sequential Implementation: Tridiagonal Reduction

If A is symmetric, then Equations (1) can be replaced by $y = \beta Au$ and $v = w = y - 1/2\beta u^T y u$, and the matrix is being reduced to tridiagonal form. In this case, it is only necessary to update the lower triangular part of matrix A at each iteration.

3.3 Parallel Implementation: Hessenberg Reduction

Given p processing nodes $\mathbf{P}_0, \dots, \mathbf{P}_{p-1}$, our parallel implementation will assume that the columns of A have been assigned to the nodes in column-wrapped fashion.

This choice of assignment allows us to parallelize Algorithm 2 as follows:

1. For all k , updating of column j of matrix A is performed by node $\mathbf{P}_{(j-1) \bmod p}$.
2. During the k th iteration, the computation of (u, β) is performed by \mathbf{P}_i such that $k \in \mathbf{P}_i$, i.e., $\mathbf{P}_{(k-1) \bmod p}$, after which it is distributed to all nodes.
3. Subtracting the j th column of βuv^T from column j requires only j th element of v , v_j , to be known to the node that owns column j . This is convenient, since $v_j = u^T[A]_{*,j}$, which can be formed by this node once u has been received. This means v can be computed in parallel, leaving the different elements of v on the nodes that computed them.
4. Subtracting the j th column of βwu^T from column j requires both v_j and $w = Au$ to be known to node $\mathbf{P}_{(j-1) \bmod p}$. Vector $w \in \mathbf{R}^n$ is computed as follows: Let B_i equal the columns of A that are assigned to node \mathbf{P}_i . If the corresponding elements of u are appropriately packed into a vector u_i^* , then $Au = \sum_{\text{all nodes}} y_i$, where $y_i = B_i u_i^*$. Hence Au can be formed by first computing partial results y_i in parallel on all nodes, followed by a global summation of the partial results, leaving Au on all nodes. Next, $u^T Au = u^T y$ and w can be formed. Notice that there is some (insignificant) redundant computation in this last step, since all nodes perform the same computation.

The resulting parallel implementation of Algorithm 2 is given by the following pseudo-code that drives each node \mathbf{P}_i :

Algorithm 3 *Parallel Hessenberg Reduction*

```

i = index of node (1)
do k = 1, ..., n - 2 (2)
  if k ∈  $\mathbf{P}_i$  then (3)
    compute  $(u, \beta) = H^{(k)}([A]_{*,k})$  (4)
    broadcast  $(u, \beta)$  to all nodes (5)
  else (6)
    receive  $(u, \beta)$  (7)
   $y_i = 0$  (8)
  do j = k, n (9)
    if j ∈  $\mathbf{P}_i$  (10)
       $v_j = u^T A$  (11)
       $y_i = y_i + v_j[A]_{*,j}$  (12)
    enddo (13)
  gsum  $y = \sum y_i$  (14)
   $w = y - \beta(u^T y)u$  (15)

```

do $j = k, n,$ (16)

 if $j \in \mathbf{P}_i$ then update $[A]_{*,j} = [A]_{*,j} - \beta v_j u - \beta v_j w$ (17)

enddo (18) enddo (19)

Statement (14) indicates that y is the result the global summation of vectors y_i . A minor redundancy exists since all processors compute w once y has been computed. This can be overcome by replacing statements (14) and (15) by

$y_i = y_i - \beta(u^T y)u$ (part of length $\approx (n - j)/p$) (14)

gsum $w = y_i$ (15)

so that all processors participate in subtracting $\beta(u^T y)u$ before the global summation.

3.4 Parallel Implementation: Tridiagonal Reduction

Parallel implementation of the reduction to tridiagonal form for a symmetric A proceeds similarly, with one major difference: Since only the lower triangular part of matrix A contains useful information, we compute y as follows: Let $A = L + R$, where L and R equal the lower triangular and strictly upper triangular parts of A , respectively. Notice that R^T equals the strictly lower triangular portion of L , and hence both are assigned to nodes in column-wrapped fashion. Now $y = Au = Lu + Ru$ can be computed by:

$y_i = 0$

do $j = k, n$

 if $j \in \mathbf{P}_i$ then

$\eta_j = \eta_j + u^T [L]_{*,k}$

$y_i = y_i + v_j [R]_{j,*}^T (= y_i + v_j [L]_{*,j})$

 enddo

$y_i = y_i - \beta(u^T y)u$ (part of length $\approx (n - j)/p$)

gsum $y = \sum y_i$

4 Blocked Algorithms

In [7] it is shown how reorganizing portions of the above algorithms in terms of Level 3 BLAS yields algorithms that perform considerably better on computers with vector processors and/or hierarchical memories. In this section we discuss sequential blocked algorithms for reduction to Hessenberg and tridiagonal form as well as their parallel implementation.

4.1 Sequential Implementation: Blocked Hessenberg Reduction

We first consider how the application of m Householder transformations can be combined:

$$H^{(k+m)} \dots H^{(k)} A^{(k)} H^{(k)} \dots H^{(k+m)} = A^{(k)} - UV^T - WU^T \quad (2)$$

where

$$\begin{aligned}
([U]_{*,j+1}, \beta) &= H^{(k+j)}([A^{(k+j)}]_{*,k+j}) \\
[V]_{*,j+1} &= A^{(k+j)T}[U]_{*,j+1} = (A^{(k)} - [U]_{*,1:j}[V]_{*,1:j}^T - [W]_{*,1:j}[U]_{*,1:j}^T)^T[U]_{*,j+1} \\
w &= A^{(k+j)}[U]_{*,j+1} = (A^{(k)} - [U]_{*,1:j}[V]_{*,1:j}^T - [W]_{*,1:j}[U]_{*,1:j}^T)[U]_{*,j+1} \\
[W]_{*,j+1} &= w - \beta(w^T[U]_{*,j+1})[U]_{*,j+1}
\end{aligned}$$

The general strategy for reorganizing Algorithm 2 now becomes:

1. Partition the matrix into panels of width m .
2. For $k = 1$, compute matrices U , V , and W by computing the successive Householder transformations. (Notice that for given j , in order to compute u , only the $(k + j)$ th column of $A^{(k+j)}$ needs to be formed.)
3. Update $A^{(k+m)} = A^{(k)} - UV^T - WU^T$. (Note: only columns $k + m, \dots, n$ need to be updated, since columns $k, \dots, k + m - 1$ were updated during the computation of U , V , and W .)
4. Repeat for $k = m + 1, 2m + 1, \dots$

Notice that the third step can now be written as two matrix-matrix operations. The bulk of the formation of the matrices requires m matrix-vector operations.

4.2 Sequential Implementation: Blocked Tridiagonal Reduction

The blocked algorithm for the reduction to tridiagonal form for the symmetric problem is reorganized similarly, except that in this case $W = V$, so Equation 2 becomes

$$H^{(k+m)} \dots H^{(k)} A^{(k)} H^{(k)} \dots H^{(k+m)} = A^{(k)} - UV^T - VU^T$$

and only the lower triangular portion of A is updated.

4.3 Parallel Implementation: Blocked Hessenberg Reduction

We now describe the parallel implementation of the blocked reduction to Hessenberg form. We will use panel-wrapped storage, where the panel width corresponds to m , the width of the panel used for the sequential blocked algorithm.

Understanding how to perform the computation in parallel is closely related to how matrices U , V , and W must be distributed in order to be able to perform the update in Equation 2. Partition V^T like $A^{(k)}$:

$$V^T = \left(V_1^T \ V_2^T \ \dots \ V_r^T \right)$$

If we update $A_j^{(k)}$ on node $\mathbf{P}_{(j-1) \bmod p}$, then U , W and V_j must be known to this node. Hence we must compute these matrices in such a way that U and W eventually reside on all nodes, while V^T is panel-wrapped distributed among the nodes.

Finally, we examine how the computation of U , V , and W can be distributed among the nodes. Assume the computation has progressed to where panel s is being reduced, i.e., $k = (s - 1)m + 1$. Assume the first j columns of U , V , and W have been computed and are distributed as desired. The computation of the $(j + 1)$ st column of these matrices proceeds as follows:

1. On node $\mathbf{P}_{(s-1) \bmod p}$, form the $(j + 1)$ st column of the current panel of $A^{(k+j)}$:

$$[A_s^{(k+j)}]_{*,j+1} = [A]_{*,k+j}^{(k+j)} = [A]_{*,k+j}^k - [U]_{*,1:j} [V]_{k+j,1:j}^T - [W]_{*,1:j} [U]_{k+j,1:j}^T$$

Since

$$[V]_{k+j,1:j} = [V_s]_{j+1,1:j}$$

all information for this operation is available on this node.

2. On $\mathbf{P}_{(s-1) \bmod p}$, compute $([U]_{*,j+1}, \beta)$ and distribute to all nodes.
3. Next, we must form three intermediate results, x , y , and z .

$$\begin{aligned} x &= [V]_{*,1:j}^T [U]_{*,j+1} \\ y &= [U]_{*,1:j}^T [U]_{*,j+1} \\ z &= [W]_{*,1:j}^T [U]_{*,j+1} \end{aligned}$$

The first requires partial sums of vectors to be accumulated on each processor, followed by a global summation of the results, leaving the results on all processors. The latter two can either be computed in the same way or they can be computed separately on each processor, leading to redundant computation, but less communication overhead.

4. Assuming x , y , and z have been computed,

$$[V]_{*,j+1} = A^{(k)T} [U]_{*,j+1} - [V]_{*,1:j} x - [U]_{*,1:j} z$$

can be computed, leaving the resulting column distributed among the nodes.

5. Computing $W_{*,j+1}$ requires

$$w = A^{(k)} [U]_{*,j+1} - [U]_{*,1:j} x - [W]_{*,1:j} y$$

to be computed. Just like the computation of w in Algorithm 3, this proceeds in two stages: columns of $A^{(k)}$ on each of the processors are summed after being multiplied by appropriate elements of $[U]_{*,j+1}$. Next, each of the vectors $[U]_{*,1:j} x$ and $[W]_{*,1:j} y$ is partitioned into p approximately equal subvectors and computation of each subvector is assigned to a node. After each node computes its section of these two vectors, and subtracts them from the partial sum of columns, a global summation computes the desired w , leaving the result on all nodes.

6. Finally,

$$[W]_{*,j+1} = w - \beta (w^T [U]_{*,j+1}) [U]_{*,j+1}$$

is formed on all nodes.

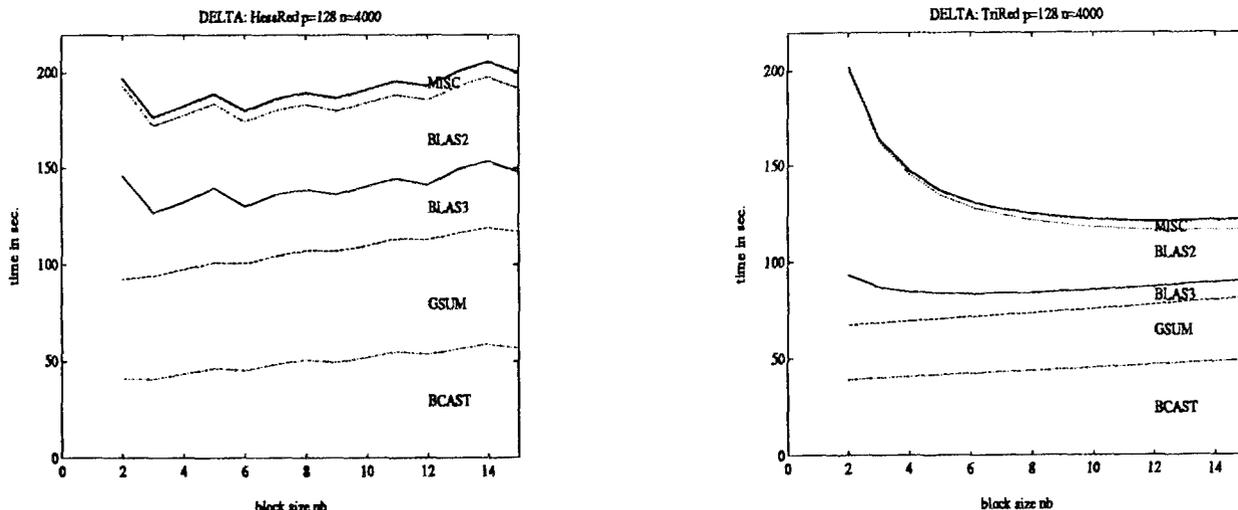


Figure 1: Total computation time for 128 nodes when $n = 4000$ and the block size n_b is varied. The space between two curves equals the time spent in the indicated operation. The times for the global sum (GSUM) and broadcast (BCAST) include some idle time that is due to load imbalance.

4.4 Parallel Implementation: Blocked Tridiagonal Reduction

The parallel implementation of the reduction to tridiagonal form for symmetric A proceeds similarly. Consider the steps given in Section 4.3: In Step 1, $[W]_{*,1:j} = [V]_{*,1:j}$; In Step 3, $z = x$, which can be either formed separately on all nodes or distributed among the nodes, which requires a global summation; Step 4-6 are merged, where $[W]_{*,j+1} = [V]_{*,j+1}$ is computed by

$$\begin{aligned} y &= \beta(A^{(k)}[U]_{*,j+1} - [V]_{*,1:j}x - [U]_{*,1:j}x) \\ [V]_{*,j+1} &= y - 1/2\beta([U]_{*,j+1}^T y)[U]_{*,j+1} \end{aligned}$$

where $\beta A^{(k)}[U]_{*,j+1}$ is computed using the same trick as in Section 3.4.

5 Experiments

In this section, we report the performance of the parallel reduction algorithms on the Intel Touchstone Delta system using the Portland Group compiler and assembly coded *single precision* BLAS routines by Kuck and Associates.

The Intel Touchstone Delta system is a distributed-memory, message-passing multicomputer of the Multiple Instruction Multiple Data (MIMD) class developed jointly by the Defense Advanced Research Projects Agency (DARPA) and the Intel Corporation [12]. It is comprised of 520 i860-based nodes, each having 16 Megabytes (MBytes) of memory, interconnected via a communications network having the topology of a two-dimensional rectangular grid. (Scaling is not restricted to a power-of-two increment typical of hypercube topologies.) It has a peak performance of ≈ 32

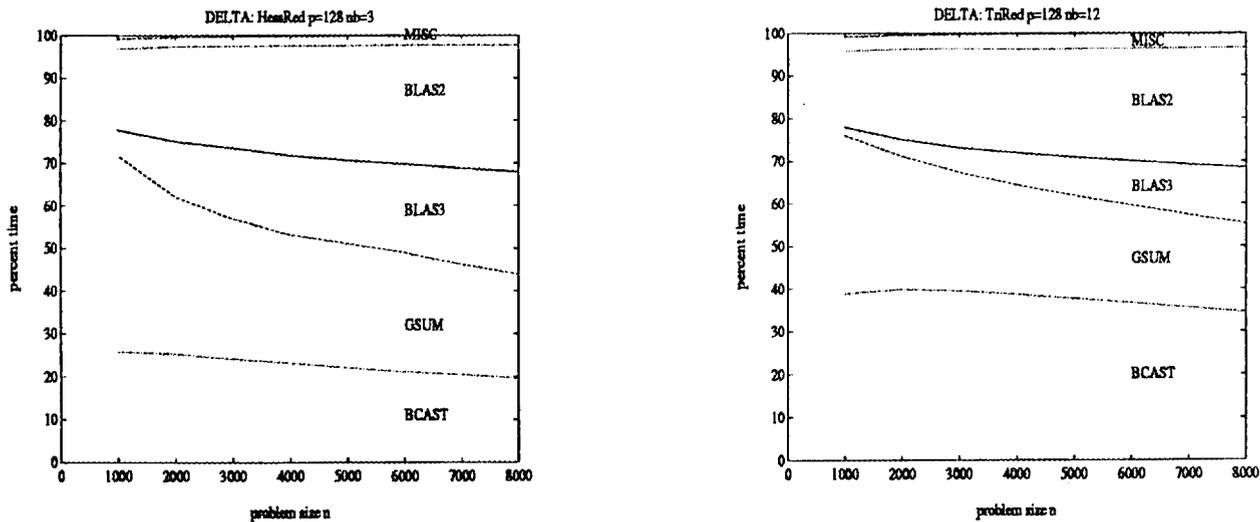


Figure 2: Allocation of execution time when $p = 128$, $nb = 3$ and the problem size n is varied. Again, the space between two curves equals the time spent in the indicated operation.

Gigaflops double precision, ≈ 40 Gigaflops single precision, and an aggregate system memory of ≈ 8 Gigabytes. The interconnection network employs a Mesh Routing Chip (MRC), developed at the California Institute of Technology, at each system node. Each MRC provides five channels, one for its associated node and four for its adjacent neighbors in the two-dimensional mesh. The channels are comprised of two, unidirectional buses: one for data flow into the MRC, one for data flow out of the MRC. The peak interprocessor communications bandwidth is ≈ 30 MBytes/s in each direction. The system supports explicit message-passing, with a latency of ≈ 75 microseconds via worm-hole routing using a packet-based protocol. Interconnect blocking is minimized by interleaving packets associated with distinct messages which need to traverse the same interconnect path.

5.1 Reduction to Hessenberg Form

Figure 1 shows the performance of the parallel reduction to Hessenberg form as a function of the problem size n and the block size nb for $p = 128$. Performance is most influenced by the performance of the Level 2 and 3 BLAS. From this graph, it can be concluded that $nb = 3$ yields reasonable performance. We will use this block size in subsequent discussions.

Communication overhead is the main contributor to the reduction in performance, as can be seen from Figures 1 and 2. In particular, the global summation and broadcast operations are major contributors to the total execution time. This is not surprising, considering a broadcast of a vector of length $O(n)$ and global summation of vectors of length n is required for each column of W that is formed (in addition to the summation of at least one smaller vector).

The performance attained as a function of problem size is clear from Figure 3. In this graph, $nb = 3$ and performance is given for various numbers of nodes. The overall performance is somewhat disappointing: The LAPACK reduction routine on a single processor attains about 45 MFLOPS.

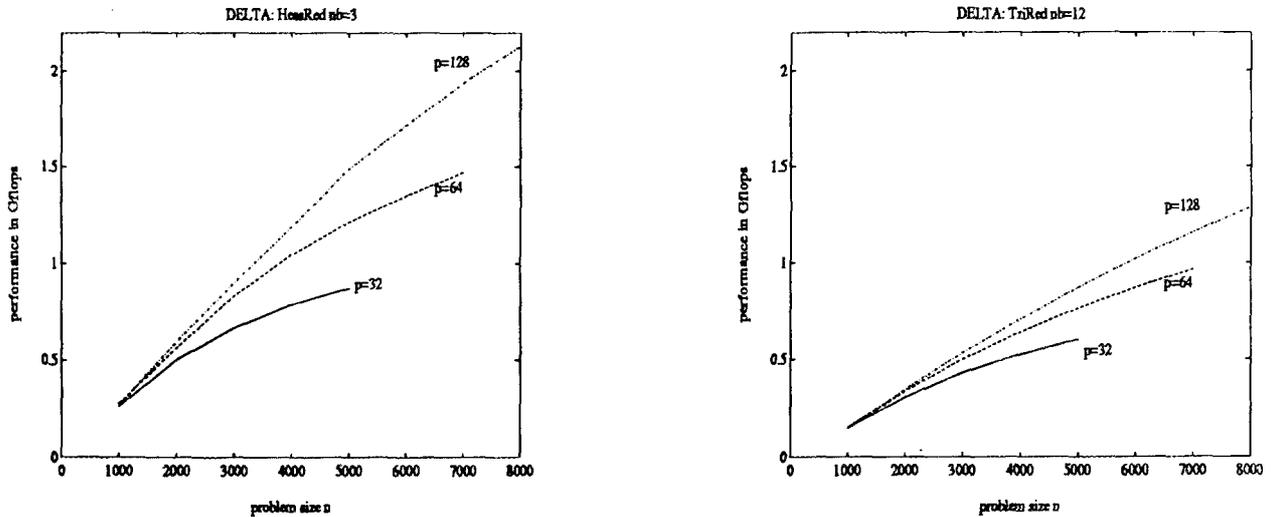


Figure 3: GFLOPS attained for various numbers of nodes when the problem size is varied. For the Hessenberg reduction, $nb = 3$, for the tridiagonal reduction, $nb = 12$.

5.2 Reduction to Tridiagonal Form

Figure 1 also shows the execution time for the parallel reduction to tridiagonal form. From this graph, it can be concluded that large block sizes yield better performance. This is due to the fact that during the update given by Equation 2 the submatrix must be updated one panel at a time, since the lower triangular part of the matrix A is wrapped onto the processors. For the same reason, the performance of the matrix-vector product (BLAS2) is affected.

The overall performance of the reduction to tridiagonal form is worse than that of the reduction to Hessenberg form (Figure 3). This can be explained as follows: The number of floating point operations is reduced by a factor 2.5 as compared to the reduction to Hessenberg form. The time spent in the broadcast is unchanged. The time spent in the global summation is approximately halved. As a result, the ratio of communication to computation is higher than for the reduction to Hessenberg form.

6 Conclusion

We have demonstrated that the LAPACK code for reducing a matrix to Hessenberg or tridiagonal form can be rewritten for current generation MIMD distributed memory computers in a relatively straight forward manner.

On the Intel Touchstone Delta, efficiency is hampered to a large degree by the cost of communication and the synchronous nature of the algorithm. If larger problems are solved, this becomes less significant. Although the Intel Touchstone Delta system has sufficient memory to store matrices of order 25000, we limited ourselves to problems that required less than 30 minutes to complete.

We have started to investigate different methods for mapping matrices to nodes. In [13], we show that wrapping onto logical tori greatly improves the performance of the LU factorization on the Intel Touchstone Delta. Future work will include the investigation of using this storage method for the reduction algorithms.

Acknowledgements

We would like to thank Al Geist for commenting on an early draft of this paper. Special thanks are given to the members of the Concurrent Supercomputing Consortium for their cooperation and contribution of time and access to the Touchstone Delta System.

References

- [1] E. Anderson, A. Benzoni, J. Dongarra, S. Moulton, S. Ostrouchov, B. Tourancheau, and R. van de Geijn. Basic Linear Algebra Communication Subprograms. In *Sixth Distributed Memory Computing Conference Proceedings*, pages 287–290. IEEE Computer Society Press, 1991.
- [2] H.Y. Chang, S. Utku, M. Salama, and D. Rapp. A parallel Householder tridiagonalization strategem using scattered row decomposition. *Intl. J. Num. Meth. Eng.*, 26:857–873, 1987.
- [3] H.Y. Chang, S. Utku, M. Salama, and D. Rapp. A parallel Householder tridiagonalization strategem using scattered square decomposition. *Parallel Computing*, 6:297–312, 1988.
- [4] Jack Dongarra and Susan Ostrouchov. LAPACK block factorization algorithms on the Intel iPSC/860. LAPACK Working Note 24, Technical Report CS-90-115, University of Tennessee, Oct. 1990.
- [5] Jack J. Dongarra, Jeremy Du Croz, Sven Hammarling, and Iain Duff. A set of level 3 basic linear algebra subprograms. *ACM Trans. Math. Soft.*, 16(1):1–17, March 1990.
- [6] Jack J. Dongarra, Jeremy Du Croz, Sven Hammarling, and Richard J. Hanson. An extended set of FORTRAN basic linear algebra subprograms. *ACM Trans. Math. Soft.*, 14(1):1–17, March 1988.
- [7] Jack J. Dongarra, Sven J. Hammarling, and Danny C. Sorensen. Block reduction of matrices to condensed forms for eigenvalue computations. *Journal of Computational and Applied Mathematics*, 27, 1989.
- [8] G.A. Geist and G.J. Davis. Finding eigenvalues and eigenvectors of unsymmetric matrices using a distributed-memory multiprocessor. *Parallel Computing*, 13:199–209, 1990.
- [9] I.C.F. Ipsen, Y. Saad, and M.H. Schultz. Complexity of dense-linear-system solution on a multiprocessor ring. *Lin. Alg. Appl.*, 77:205–239, 1986.

- [10] G.W. Juszczak. Efficient portable parallel matrix computations. Master's thesis, University of Texas at Austin, 1989. Technical Report TR-89-38.
- [11] J.W. Juszczak and R.A. van de Geijn. An experiment in coding portable parallel matrix algorithms. In *Proceedings of the Fourth Conference on Hypercube Concurrent Computers and Applications*, 1989.
- [12] S.L. Lillevik. The Touchstone 30 Gigaflop DELTA Prototype. In *Sixth Distributed Memory Computing Conference Proceedings*, pages 671–677. IEEE Computer Society Press, 1991.
- [13] R.A. van de Geijn. Massively parallel LINPACK benchmark on the Intel Touchstone Delta and iPSC/860 systems. Computer Science report TR-91-28, Univ. of Texas, 1991.

INTERNAL DISTRIBUTION

- | | | | |
|--------|-----------------|--------|------------------------------------|
| 1. | B. R. Appleton | 23. | T. H. Rowan |
| 2-3. | T. S. Darland | 24-28. | R. F. Sincovec |
| 4. | E. F. D'Azevedo | 29-33. | R. C. Ward |
| 5. | J. M. Donato | 34. | P. H. Worley |
| 6-10. | J. J. Dongarra | 35. | A. Zucker |
| 11. | T. H. Dunigan | 36. | Central Research Library |
| 12. | G. A. Geist | 37. | ORNL Patent Office |
| 13. | M. R. Leuze | 38. | K-25 Applied
Technology Library |
| 14. | E. G. Ng | 39. | Y-12 Technical Library |
| 15. | C. E. Oliver | 40. | Laboratory Records - RC |
| 16. | B. W. Peyton | 41-42. | Laboratory Records Department |
| 17-21. | S. A. Raby | | |
| 22. | C. H. Romine | | |

EXTERNAL DISTRIBUTION

43. Cleve Ashcraft, Boeing Computer Services, P.O. Box 24346, M/S 7L-21, Seattle, WA 98124-0346
44. Robert G. Babb, Department of Computer Science and Engineering, Oregon Graduate Institute, 19600 N.W. Walker Rd., Beaverton, OR 97006
45. David H. Bailey, NASA Ames Research Center, Mail Stop 258-5, Moffett Field, CA 94035
46. Jesse L. Barlow, Department of Computer Science, Pennsylvania State University, University Park, PA 16802
47. Edward H. Barsis, Computer Science and Mathematics, P. O. Box 5800, Sandia National Laboratories, Albuquerque, NM 87185
48. Eric Barszcz, NASA Ames Research Center, MS T045-1, Moffett Field, CA 94035
49. Robert E. Benner, Parallel Processing Division 1413, Sandia National Laboratories, P. O. Box 5800, Albuquerque, NM 87185
50. Donna Bergmark, Cornell Theory Center, Engineering and Theory Center Building, Ithaca, NY 14853-3901
51. Chris Bischof, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Ave., Argonne, IL 60439
52. Ake Bjorck, Department of Mathematics, Linkoping University, S-581 83 Linkoping, Sweden

53. Jean R. S. Blair, Department of Computer Science, Ayres Hall, University of Tennessee, Knoxville, TN 37996-1301
54. Daniel Boley, Department of Computer Science, University of Minnesota, 200 Union St. S.E. Rm.4-192 Minneapolis, MN 55455
55. Roger W. Brockett (EPMD Advisory Committee), Wang Professor of Electrical Engineering and Computer Science, Division of Applied Sciences, Harvard University, Cambridge, MA 02138
56. James C. Browne, Department of Computer Sciences, University of Texas, Austin, TX 78712
57. Bill L. Buzbee, Scientific Computing Division, National Center for Atmospheric Research, P.O. Box 3000, Boulder, CO 80307
58. Donald A. Calahan, Department of Electrical and Computer Engineering, University of Michigan, Ann Arbor, MI 48109
59. John Cavallini, Office of Scientific Computing, Office of Energy Research, ER-7, Germantown Building, U.S. Department of Energy, Washington, DC 20545
60. Ian Cavers, Department of Computer Science, University of British Columbia, Vancouver, British Columbia V6T 1W5, Canada
61. Tony Chan, Department of Mathematics, University of California, Los Angeles, 405 Hilgard Ave., Los Angeles, CA 90024
62. Jagdish Chandra, Army Research Office, P.O. Box 12211, Research Triangle Park, NC 27709
63. Eleanor Chu, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
64. Melvyn Ciment, National Science Foundation, 1800 G Street N.W., Washington, DC 20550
65. Thomas Coleman, Department of Computer Science, Cornell University, Ithaca, NY 14853
66. Paul Concus, Mathematics and Computing, Lawrence Berkeley Laboratory, Berkeley, CA 94720
67. Jane K. Cullum, IBM T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598
68. George Cybenko, Center for Supercomputing Research and Development, University of Illinois, 104 S. Wright St., Urbana, IL 61801-2932
69. George J. Davis, Department of Mathematics, Georgia State University, Atlanta, GA 30303
70. John J. Dornig, (EPMD Advisory Committee), Department of Nuclear Engineering Physics, Thornton Hall, McCormack Rd., University of Virginia, Charlottesville, VA 22901

71. Iain S Duff, Atlas Centre, Rutherford Appleton Laboratory, Chilton, Oxon OX11 0QX England
72. Patricia Eberlein, Department of Computer Science, SUNY at Buffalo, Buffalo, NY 14260
73. Stanley Eisenstat, Department of Computer Science, Yale University, P.O. Box 2158 Yale Station, New Haven, CT 06520
74. Lars Elden, Department of Mathematics, Linkoping University, 581 83 Linkoping, Sweden
75. Howard C. Elman, Computer Science Department, University of Maryland, College Park, MD 20742
76. Albert M. Erisman, Boeing Computer Services, P.O. Box 24346, M/S 7L-21, Seattle, WA 98124-0346
77. Ian Foster, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Ave., Argonne, IL 60439
78. Geoffrey C. Fox, Booth Computing Center 158-79, California Institute of Technology, Pasadena, CA 91125
79. Paul O. Frederickson, NASA Ames Research Center, RIACS, M/S T045-1 Moffett Field, CA 94035
80. Fred N. Fritsch, Computing & Mathematics Research Division, Lawrence Livermore National Laboratory, P. O. Box 808, L-316 Livermore, CA 94550
81. Robert E. Funderlic, Department of Computer Science, North Carolina State University, Raleigh, NC 27650
82. Dennis B. Gannon, Computer Science Department, Indiana University, Bloomington, IN 47405
83. David M. Gay, Bell Laboratories, 600 Mountain Ave., Murray Hill, NJ 07974
84. C. William Gear, Computer Science Department, University of Illinois, Urbana, IL 61801
85. W. Morven Gentleman, Division of Electrical Engineering, National Research Council, Building M-50, Room 344, Montreal Rd., Ottawa, Ontario, Canada K1A 0R8
86. J. Alan George, Vice President, Academic and Provost, Needles Hall, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
87. John R. Gilbert, Xerox Palo Alto Research Center, 3333 Coyote Hill Rd., Palo Alto, CA 94304
88. Gene H. Golub, Department of Computer Science, Stanford University, Stanford, CA 94305
89. Joseph F. Grcar, Division 8331, Sandia National Laboratories, Livermore, CA 94550
90. Sven Hammarling, Numerical Algorithms Group Ltd. Wilkinson House, Jordan Hill Road Oxford OX2 8DR, United Kingdom

91. Per Christian Hansen, UNI*C Lyngby, Building 305, Technical University of Denmark, DK-2800 Lyngby, Denmark
92. Richard Hanson, IMSL Inc., 2500 Park West Tower One, 2500 City West Blvd., Houston, TX 77042-3020
93. Michael T. Heath, National Center for Supercomputing Applications, 4157 Beckman Institute, University of Illinois, 405 North Mathews Avenue, Urbana, IL 61801-2300
94. Don E. Heller, Physics and Computer Science Department, Shell Development Co., P.O. Box 481, Houston, TX 77001
95. Nicholas J. Higham, Department of Mathematics, University of Manchester, Grt Manchester, M13 9PL, England
96. Charles J. Holland, Air Force Office of Scientific Research, Building 410, Bolling Air Force Base, Washington, DC 20332
97. Robert E. Huddleston, Computation Department, Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, CA 94550
98. Ilse Ipsen, Department of Computer Science, Yale University, P.O. Box 2158 Yale Station, New Haven, CT 06520
99. Elizabeth Jessup, University of Colorado, Department of Computer Science, Boulder, CO 80309-0430
100. Lennart Johnsson, Thinking Machines Inc., 245 First St., Cambridge, MA 02142-1214
101. Harry Jordan, Department of Electrical and Computer Engineering, University of Colorado, Boulder, CO 80309
102. Bo Kagstrom, Institute of Information Processing, University of Umea, 5-901 87 Umea, Sweden
103. Malvin H. Kalos, Cornell Theory Center, Engineering and Theory Center Building, Cornell University, Ithaca, NY 14853-3901
104. Hans Kaper, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Ave., Argonne, IL 60439
105. Robert J. Kee, Applied Mathematics Division 8331, Sandia National Laboratories, Livermore, CA 94550
106. Kenneth Kennedy, Department of Computer Science, Rice University, P.O. Box 1892, Houston, TX 77005
107. Thomas Kitchens, Department of Energy, Scientific Computing Staff, Office of Energy Research, ER-7, Office G-236 Germantown, Washington, DC 20585
108. Richard Lau, Code 1111MA, 800 N. Quincy Street, Boston Tower, 1 Arlington, VA 22217-5000
109. Alan J. Laub, Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106

110. Robert L. Launer, Army Research Office, P.O. Box 12211, Research Triangle Park, North Carolina 27709
111. Charles Lawson, MS 301-490, Jet Propulsion Laboratory, 4800 Oak Grove Dr., Pasadena, CA 91109
112. Peter D. Lax, Courant Institute of Mathematical Sciences, New York University, 251 Mercer St., New York, NY 10012
113. James E. Leiss (EPMD Advisory Committee), Rt. 2, Box 142C, Broadway, VA 22815
114. John G. Lewis, Boeing Computer Services, P.O. Box 24346, M/S 7L-21, Seattle, WA 98124-0346
115. Jing Li, IMSL Inc., 2500 Park West Tower One, 2500 City West Blvd., Houston, TX 77042-3020
116. Joseph Liu, Department of Computer Science, York University, 4700 Keele St., North York, Ontario, Canada M3J 1P3
117. Franklin Luk, School of Electrical Engineering, Cornell University, Ithaca, NY 14853
118. Thomas A. Manteuffel, Department of Mathematics, University of Colorado - Denver, Denver, CO 80202
119. Peter Mayes, NAG Ltd., Wilkinson House, Jordan Hill Road, Oxford, OX2 8DR, United Kingdom
112. Paul C. Messina, Mail Code 158-79, California Institute of Technology, 1201 E. California Blvd. Pasadena, CA 91125
113. James McGraw, Lawrence Livermore National Laboratory, L-306, P.O. Box 808, Livermore, CA 94550
114. Cleve Moler, The Mathworks, 325 Linfield Place, Menlo Park, CA 94025
115. Neville Moray (EPMD Advisory Committee), Department of Mechanical and Industrial Engineering, University of Illinois, 1206 West Green Street, Urbana, IL 61801
116. Brent Morris, National Security Agency, Ft. George G. Meade, MD 20755
117. Dianne P. O'Leary, Computer Science Department, University of Maryland, College Park, MD 20742
118. James M. Ortega, Department of Applied Mathematics, Thornton Hall University of Virginia, Charlottesville, VA 22903
119. Chris Paige, Department of Computer Science, McGill University, 805 Sherbrooke St. W., Montreal, Quebec, Canada H3A 2K6
120. Roy P. Pargas, Department of Computer Science, Clemson University, Clemson, SC 29634-1906
121. Beresford N. Parlett, Department of Mathematics, University of California, Berkeley, CA 94720

122. Merrell Patrick, Department of Computer Science, Duke University, Durham, NC 27706
123. Robert J. Plemmons, Departments of Mathematics and Computer Science, North Carolina State University, Raleigh, NC 27650
124. Jesse Poore, Department of Computer Science, Ayres Hall, University of Tennessee, Knoxville, TN 37996-1301
125. Alex Pothan, Department of Computer Science, Pennsylvania State University, University Park, PA 16802
126. Michael J. Quinn, Computer Science Department, Oregon State University, Corvallis, OR 97331
127. Giuseppe Radicati di Brozolo, IBM European Center for Scientific and Engineering Computing, 00147 Roma, via Giorgione 159, Italy
128. Noah Rhee, Department of Mathematics, University of Missouri-Kansas City, Kansas City, MO 64110-2499
129. John K. Reid, Numerical Analysis Group, Central Computing Department, Atlas Centre, Rutherford Appleton Laboratory, Didcot, Oxon OX11 0QX, England
130. Werner C. Rheinboldt, Department of Mathematics and Statistics, University of Pittsburgh, Pittsburgh, PA 15260
131. John R. Rice, Computer Science Department, Purdue University, West Lafayette, IN 47907
132. Garry Rodrigue, Numerical Mathematics Group, Lawrence Livermore Laboratory, Livermore, CA 94550
133. Donald J. Rose, Department of Computer Science, Duke University, Durham, NC 27706
134. Ahmed H. Sameh, Computer Science Department, University of Illinois, Urbana, IL 61801
135. Michael Saunders, Systems Optimization Laboratory, Operations Research Department, Stanford University, Stanford, CA 94305
136. Robert Schreiber, RIACS, Mail Stop 230-5, NASA Ames Research Center, Moffet Field, CA 94035
137. Martin H. Schultz, Department of Computer Science, Yale University, P.O. Box 2158 Yale Station, New Haven, CT 06520
138. David S. Scott, Intel Scientific Computers, 15201 N.W. Greenbrier Pkwy., Beaverton, OR 97006
139. Lawrence F. Shampine, Mathematics Department, Southern Methodist University, Dallas, TX 75275

140. Kermit Sigmon, Department of Mathematics, University of Florida, Gainesville, FL 32611
141. Horst Simon, Mail Stop 258-5, NASA Ames Research Center, Moffett Field, CA 94035
142. Larry Snyder, Department of Computer Science and Engineering, FR-35, University of Washington, Seattle, WA 98195
143. Danny C. Sorensen, Department of Mathematical Sciences, Rice University, P. O. Box 1892, Houston, TX 77251
144. Rick Stevens, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Ave., Argonne, IL 60439
145. G. W. Stewart, Computer Science Department, University of Maryland, College Park, MD 20742
146. Quentin F. Stout, Department of Electrical and Computer Engineering, University of Michigan, Ann Arbor, MI 48109
147. Daniel B. Szyld, Department of Computer Science, Duke University, Durham, NC 27706-2591
148. W.-P. Tang, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
149. Michael Thomason, Department of Computer Science, Ayres Hall, University of Tennessee, Knoxville, TN 37996-1301
150. Bernard Tourancheau, LIP ENS-Lyon 69364 Lyon cedex 07, France
- 151-155. Robert A. van de Geijn, Department of Computer Science, University of Texas, Austin, TX 78712
156. Charles Van Loan, Department of Computer Science, Cornell University, Ithaca, NY 14853
157. James M. Varah, Centre for Integrated Computer Systems Research, University of British Columbia, Office 2053-2324 Main Mall, Vancouver, British Columbia V6T 1W5, Canada
158. Robert G. Voigt, ICASE, MS 132-C, NASA Langley Research Center, Hampton, VA 23665
159. Michael Vose, Department of Computer Science, Ayres Hall, University of Tennessee, Knoxville, TN 37996-1301
160. Phuong Vu, Cray Research Inc., 1408 Northland Dr., Mendota Heights, MN 55120
161. E. L. Wachspress, Department of Mathematics, University of Tennessee, Knoxville, TN 37996-1300
162. Daniel D. Warner, Department of Mathematical Sciences, O-104 Martin Hall, Clemson University, Clemson, SC 29631

163. D. S. Watkins, Department of Pure and Applied Mathematics, Washington State University, Pullman, WA 99164-2930
164. Mary F. Wheeler (EPMD Advisory Committee), Rice University, Department of Mathematical Sciences, P.O. Box 1892, Houston, TX 77251
165. Andrew B. White, Computing Division, Los Alamos National Laboratory, Los Alamos, NM 87545
166. Michael Wolfe, Oregon Graduate Institute, 19600 N.W. von Neumann Dr., Beaverton, OR 97006
167. Margaret Wright, Bell Laboratories, 600 Mountain Ave., Murray Hill, NJ 07974
168. David Young, University of Texas, Center for Numerical Analysis, RLM 13.150, Austin, TX 78731
169. Office of Assistant Manager for Energy Research and Development, U.S. Department to Energy, Oak Ridge Operations Office, P.O. Box 2001, Oak Ridge, TN 37831-8600
- 170-179. Office of Scientific Technical Information, P.O. Box 62, Oak Ridge, TN 37831