D R A F T

Document for the Basic Linear Algebra Subprograms (BLAS) Standard

Basic Linear Algebra Subprograms Technical (BLAST) Forum

December 6, 1997

Contents

The discussion in each chapter is not limited only to the points listed, any related topics can be included at the discretion of the chapter authors. Final acceptance of each section will be decided at the next meeting using *Robert's Rules*. See BLAST Forum homepage for more detailed rules.

Chapter 1

Introduction

1.1 Summary

This Chapter gives an introduction to the work of the BLAS Technical Forum (BLAST Forum) and lists the perceived functionality of numerical linear algebra library kernels as discussed at meetings of the BLAST Forum. Included are kernels needed for dense matrices, sparse direct methods, sparse iterative methods as well as related kernels needed in non-traditional linear algebra computations such as CFD and finite elements. This chapter covers overall functionality without regard for whether or not the function is required in a particular context, such as sparsity.

This chapter of the document starts by discussing the background, motivation and history and then presents tables listing the functionality. Further information on the BLAS technical Forum can be found at the URL:

http://www.netlib.org/utk/papers/blast-forum.html

Comments on this document and active participation in the Forum are encouraged.

1.2 Background, Motivation and History

< This section to be worked on>

A technical forum has been established to consider expanding the Basic Linear Algebra Subprograms (BLAS) in a number of directions in the light of modern software, language, and hardware developments.

Various working groups have been established to consider issues such as the overall functionality, language interfaces, sparse BLAS, distributed BLAS and extensions to the existing BLAS.

- history of BLAS, need for a standard(s)
- previous efforts to standardize and promulgate
- motivations for extensions, scope approach, and public meetings
- what was accomplished in BLAST
- industry, academic, and laboratory involvement
- explanation of the BLAST process
- where the state-of-the-art can go from here on out

1.3 Organization of the Document

<This section needs to be written>

1.4 Nomenclature and Conventions Used in this Document

The following notation is used throughout the document.

<Need to gather up notation to this section>

- w, x, y, z - vectors
- Greek letters - scalars (but not exclusively Greek letters)
- x_i - an element of a one-dimensional array
- x_I - subset of entries of x indexed by I
- $y[i]$ - an array from an array of arrays
- x_{C} - compressed (sparse) array, only non-zeros are stored, associated with an index set I
- e - vector whose elements are all unity

<The rest of this section needs to be written>

1.5 Overall Functionality

<This section still needs some re-writing>

This section summarizes, in tabular form, the discussions of the Functionality Working Group. Note that this section does not consider issues such as storage formats or data types.

The list is fairly exhaustive and it is not assumed that everything will necessarily be included in the final report. Some indication of the priority of the functions is given in the tables, together with some justification for the inclusion of the functions. The list includes the functionality covered by the existing Level 1, 2 and 3 BLAS [38, 23, 25, 24].

This section considers overall functionality, irrespective of whether or not a function is required in the dense or the sparse case. Many of the operations could usefully also be made available as multiple instances and that is discussed in a separate chapter.

In this version of the document, the functionality is categorized more or less along traditional BLAS lines, but the Forum discussed other schemes and it may be appropriate to categorize the functionality in more than one way in the final document.

If what you need is not listed here, please comment and if possible join the Forum, to let us know what you need.

1.5.1 Justifications for extensions to the BLAS

In the original BLAS, each level was categorised by the type of operation; Level 1 covered scalar and vector operations, Level 2 covered matrix vector operations, while Level 3 covered matrix-matrix operations. The functionality tables in this document are categorised in a similar manner, but with some additional categories to cover operations which it was realized were missing in the original BLAS.

The motivation for the new operations is proven functionality. No function was to be added because it just neatly fit into a framework and no subroutine was to be discarded because it did not fit into a particular framework.

Many of the new functions are auxiliary routines in LAPACK [6]. Only after the LAPACK project was begun was it realized that there were operations like the matrix copy routine, the computation of a norm of a matrix and the generation of Householder transformations that occurred so often that it was wise to make separate subroutines for them.

A second group of these extensions extended the functionality of some of the existing BLAS. For example, the Level 3 BLAS for the rank k update of a symmetric matrix only allows a positive update, which means that it cannot be used for the reduction of a symmetric matrix to tridiagonal form (to facilitate the computation of the eigensystem of a symmetric matrix), or for the decomposition of an indefinite matrix, or for a quasi-Newton update in an optimization routine.

Some of the new BLAS might be considered special cases of the original BLAS. The number of Level 2 and 3 BLAS was purposely kept small, but each had a number of parameters that could extend their functionality. Most of these parameters in practice were either 1.0 or 0.0. Adding these parameters, rather than specifying different subroutines increased the opportunities for the user to make mistakes and increased the amount of overhead of a call to the routine. In this document many of the functions are presented in their general case, but we make recommendations about special cases in the Language Independent Specification chapter.

The extensions considered in the section on multiple instances can be thought of as multiple instances of the Level 1 BLAS. They tend to arise when working with narrow banded and sparse or structured matrices. The canonical example is the decomposition of many narrow banded matrices simultaneously. This type of problem arises when finding several eigenvectors using inverse iteration, when using parallel bisection to determine the eigenvalues of a symmetric matrix, when solving two dimensional separable partial differential equations or preconditioning an iterative method for a nonseparable differential equation. Certainly if one was on a shared memory multi-processor computer using a compiler with compiler directives one could specify that various Level 1 BLAS like IxAMAX, xSWAP and xAXPY be done simultaneously. However, when one is not in this situation and one wishes to use portable code, then the existence of a BLAS to cover this situation is helpful.

Other contenders for multiple instance routines are routines for generating and applying Givens transformation, used for example to reduce a banded matrix to tridiagonal form in order to solve the symmetric eigenvalue problem. Simultaneous Givens transformations are also useful when solving the tridiagonal positive definite symmetric generalized eigenvalue problem and when using the QR algorithm to solve the standard tridiagonal symmetric eigenvalue problem.

For large matrix problems, it might not be feasible or desirable to store the full matrix. In this situation one cannot simply use the matrix-vector multiplication routines in the present BLAS. However, one may be able to form that product with some simultaneous inner product routines because they have some repeated submatrices. This feature is particular true of Toeplitz and Hankel matrices used in signal and image processing.

In the following tables, the last column of each table gives the justification for a routine's inclusion.

- "LEGACY" refers to the fact that it is a legacy BLAS,
- "AUX" means that it has appeared as an auxiliary routine in LAPACK, reference numbers refer to items in the bibliography,

- "DERIV" means that it is a commonly used subset of an existing BLAS which deserves its own subroutine, and
- "COMMON" means that it appears so commonly in existing software, that it is surprising that it is not presently a BLAS.

A question mark in the last column suggests that it will probably be a new BLAS if there is sufficient support and a "-" sign suggests that it will probably never be in the final list. Not all of the auxiliary subroutine in LAPACK are included and some have question marks or - signs next to them because their utility outside of LAPACK has not yet been demonstrated sufficiently.

1.5.2 Scalar and Vector Operations

This sub-section gives the operations corresponding to the traditional Level 1 BLAS operations. Table 1.1 lists the scalar and vector reduction operations, Table 1.2 lists the vector operations, and Table 1.3 lists those vector operations that involve only data movement. The operations are intended to apply to both real and complex arguments. For the sake of compactness the complex operators are omitted, so that whenever a transpose operation is given the conjugate transpose should also be assumed for the complex case.

Inner product	$r \leftarrow x^T y$,	Legacy
	$r \leftarrow r + x^T y$	Legacy
	$r \leftarrow \alpha r + \beta x^T y$	-
Vector norms	$r \leftarrow \|\|x\|\|_1$,	Legacy
	$r \leftarrow \|\|x\|\|_\infty$	Common
	$r \leftarrow \|\|x\|\|_2$,	Legacy
	$r \leftarrow \|\|x\|\|_p$	-
Sum	$r \leftarrow \sum_i x_i$	Common
Location of max value	$i = \arg\max_i \|x_i\|$	Legacy
Max value & location	$i, x_i, ; i = \arg\max_i x_i$	Common
Max absolute value & location	$i, \|x_i\|, i = \arg\max_i \|x_i\|$	Common
Min value & location	$i, x_i, ; i = \arg\min_i x_i$	Common
Min absolute value & location	$i, \|x_i\|, i = \arg\min_i \|x_i\|$	Common
Sum of squares	$(a, b) \leftarrow \sum x_i^2$,	Common
	$a \cdot b^2 = \sum x^2$	
Generate Givens rotation	$(c, s) \leftarrow \mathrm{rot}(a, b)$	Legacy
Generate Modified Givens rotation	$(z, r) \leftarrow \mathrm{mrot}(a, b)$	Legacy
Generate Jacobi rotation	$(c, s) \leftarrow \mathrm{jrot}(a, b)$	(Future) Aux
Generate Householder transform	$(\alpha, u) \leftarrow \mathrm{house}(x)$,	Aux
	$H = I - \alpha u u^T$	

Table 1.1: Reduction Operations

1.5.3 Matrix Vector Operations

This sub-section gives the operations corresponding to the traditional Level 2 BLAS operations. Table 1.4 lists the matrix-vector operations. The operations are intended to apply to both real and complex arguments, and the matrix arguments A, T, L and U are intended to be dense or

Zero	$x \leftarrow 0$	Common
Constant	$x \leftarrow \alpha e$	Common
Conjugate copy	$y \leftarrow \bar{x}$	Common
Scale	$x \leftarrow \alpha x,$	Legacy
	$y \leftarrow \alpha x$	Common
Group scale	$x_{I_i} \leftarrow a_i x_{I_i}$	Common,
Sum	$y \leftarrow \pm x + y,$	?
	$w \leftarrow \pm x + y$	?
Shift	$x \leftarrow x + \sigma e,$	?
	$y \leftarrow x + \sigma e$	?
AXPY & AXPY like	$y \leftarrow \alpha x + y,$	Legacy
	$w \leftarrow \alpha x + y$	Common
	$y \leftarrow \alpha x + \beta y,$	?
	$w \leftarrow \alpha x + \beta y$	Common,
	$y \leftarrow \alpha x + \beta y + \sigma e,$	?
	$w \leftarrow \alpha x + \beta y + \sigma e$	?
Absolute value	$x_i \leftarrow \|x_i\|$	
	$y_i \leftarrow \|x_i\|$	Common
Elementwise multiply	$x_i \leftarrow x_i y_i,$	Common,
	$w_i \leftarrow x_i y_i$	?
Elementwise divide	$x_i \leftarrow x_i / y_i,$	Common
	$w_i \leftarrow x_i / y_i$	?
Apply plane rotation	$(\; x \quad y \;) \leftarrow (\; x \quad y \;) R$	Legacy
Apply modified plane rotation	$(\; \tilde{x} \quad \tilde{y} \;) D \leftarrow (\; \tilde{x} \quad \tilde{y} \;) D R$	Legacy

Table 1.2: Vector Operations

Copy	$y \leftarrow x$	Common
Swap	$y \leftrightarrow x$	Legacy
Sort into order & return index vector	$(iv, y) \leftarrow \mathrm{sort}(x)$	Common
Sort according to index vector	$y \leftarrow \mathrm{sort}(x, iv)$	Common

Table 1.3: Data Movement with Vectors

banded or sparse. In addition, where appropriate, the matrix A can be symmetric (Hermitian) or triangular or general. The matrix T specifically represents a triangular matrix, either upper or lower, L represents a lower triangular matrix and U an upper triangular matrix, all three of which can be unit or non-unit triangular.

D represents at least a diagonal matrix, and maybe also a 2×2 blocked diagonal or a tridiagonal matrix. Q represents an orthogonal (unitary) matrix and it is an open question as to how Q should be stored.

For the sake of compactness the complex operators are omitted, so that whenever a transpose operation is given both the conjugate and conjugate transpose should also be assumed for the complex case.

Matrix vector product	$x \leftarrow Ax,\ x \leftarrow A^T x$	Deriv
	$y \leftarrow \pm Ax,\ y \leftarrow \pm A^T x$	Deriv
	$y \leftarrow \alpha Ax + \beta y,\ y \leftarrow \alpha A^T x + \beta y$	Legacy
Triangular solve	$x \leftarrow T^{-1}x,\ x \leftarrow T^{-T}x$	Legacy
	$x \leftarrow \alpha T^{-1}x,\ x \leftarrow \alpha T^{-T}x$	-
	$y \leftarrow \alpha T^{-1}x,\ y \leftarrow \alpha T^{-T}x$	-
Rank one and symmetric ($A = A^T$)	$A \leftarrow \beta x y^T + \alpha A$	?
rank one & two updates	$A \leftarrow \beta x y^T + A$	Legacy
	$A \leftarrow \beta x x^T + A$	Legacy
	$A \leftarrow \beta x x^T + \alpha A,\ A \leftarrow \beta x D x^T + \alpha A$	?
	$A \leftarrow (\beta x) y^T + y(\beta x)^T + \alpha A$	-
	$A \leftarrow (\beta x D) y^T + y(\beta x D)^T + \alpha A$	-
Application of Q from	$x \leftarrow Qx,\ x \leftarrow Q^T x$	Aux
QR factorization	$y \leftarrow Q^T x\ y \leftarrow Q^T x$	Aux

Table 1.4: Matrix Vector Operations

1.5.4 Matrix Operations

This sub-section gives the operations corresponding to the traditional Level 3 BLAS operations, as well as a number of additional single matrix operations. Table 1.5 lists single matrix operations and matrix operations that involve $O(n^2)$ operations, Table 1.6 lists the $O(n^3)$ matrix-matrix operations and Table 1.7 lists those matrix operations that involve only data movement. The operations are intended to apply to both real and complex arguments. Where appropriate one or more of the matrices can also be symmetric (Hermitian) or triangular or general. The matrix T specifically represents a triangular matrix, either upper or lower, L represents a lower triangular matrix and U an upper triangular matrix, all three of which can be unit or non-unit triangular.

D represents at least a diagonal matrix, and maybe also a 2×2 blocked diagonal or a tridiagonal matrix. Q represents an orthogonal (unitary) matrix and it is an open question as to how Q should be stored.

For the sake of compactness the complex operators are omitted, so that whenever a transpose operation is given both the conjugate and conjugate transpose should also be assumed for the complex case.

Matrix set	$A \leftarrow \alpha I$,	Common
	$A \leftarrow \alpha I + \beta(e^T e - I)$	-
Matrix norms	$r \leftarrow \|\|A\|\|_1, r \leftarrow \|\|A\|\|_F, r \leftarrow \|\|A\|\|_\infty$	Aux
Maximum absolute value	$r \leftarrow \max \|a_{i,j}\|$	Common
Matrix scaling	$A \leftarrow \alpha A,\ B \leftarrow \alpha A$	Aux
Diagonal scaling	$A \leftarrow DA,\ A \leftarrow AD,\ A \leftarrow D_1 A D_2$	Common
Matrix add	$A \leftarrow \pm A + B,\ C \leftarrow \pm A + B$	Common
Matrix add and scale	$A \leftarrow \alpha A + \beta B,\ C \leftarrow \alpha A + \beta B$	?
	$A \leftarrow \alpha A + \beta B^T$	
	$A \leftarrow A + BD$	
Elementwise multiply	$a_{ij} \leftarrow a_{ij} b_{ij},\ c_{ij} \leftarrow a_{ij} b_{ij}$	-
Elementwise divide	$a_{ij} \leftarrow a_{ij}/b_{ij},\ c_{ij} \leftarrow a_{ij}/b_{ij}$	-

Table 1.5: Matrix Operations – $O(n^2)$

Matrix matrix product	$A \leftarrow AB,\ A \leftarrow A^T B,\ A \leftarrow AB^T,\ A \leftarrow A^T B^T$	?
	$C \leftarrow AB,\ C \leftarrow A^T B,\ C \leftarrow AB^T C \leftarrow A^T B^T$	Deriv
	$C \leftarrow \alpha AB + \beta C,$	Legacy
	$C \leftarrow \alpha AB + \beta C, (C = C^T)$	-
	$F \leftarrow \alpha AB + \beta C$	-
	$C \leftarrow \alpha A^T B + \beta C$	Legacy
	$F \leftarrow \alpha A^T B + \beta C$	-
	$C \leftarrow \alpha AB^T + \beta C$	Legacy
	$F \leftarrow \alpha AB^T + \beta C$	-
	$C \leftarrow \alpha A^T B^T + \beta C$	Legacy
	$F \leftarrow \alpha A^T B^T + \beta C$	-
Triangular	$B \leftarrow \alpha T^{-1} B$	Legacy
	$C \leftarrow \alpha T^{-1} B$	-
	$B \leftarrow \alpha B T^{-1},\ C \leftarrow \alpha B T^{-1}$	-
Symmetric rank k & $2k$	$C \leftarrow \alpha AA^T + \beta C,\ C \leftarrow \alpha A^T A + \beta C$	Legacy
updates $(C = C^T)$	$C \leftarrow \alpha ADA^T + \beta C,\ C \leftarrow \alpha A^T DA + \beta C$	
	$C \leftarrow (\alpha A)B^T + B(\alpha A)^T + \beta C$	Legacy
	$C \leftarrow (\alpha A)^T B + B^T (\alpha A) + \beta C$	-
	$C \leftarrow (\alpha AD)B^T + B(\alpha AD)^T + \beta C$	?
	$C \leftarrow (\alpha DA)^T B + B^T (\alpha DA) + \beta C$	-
Application of Q from	$X \leftarrow QX,\ X \leftarrow XQ,\ X \leftarrow Q^T X,\ X \leftarrow XQ^T$	?
QR factorization	$Y \leftarrow QX,\ Y \leftarrow XQ,\ Y \leftarrow Q^T X,\ Y \leftarrow XQ^T$	?

Table 1.6: Matrix Matrix Operations

Matrix copy	$B \leftarrow A$	Aux
Matrix transpose	$A \leftarrow A^T,\ B \leftarrow A^T$	?
Get and put submatrix	$B \leftarrow a_{I,J}$	Aux, Common
	$b_{I,J} \leftarrow A,\ b_{I,J} \leftarrow A + b_{I,J}$	Aux, Common
Permute Matrix	$A \leftarrow PA,\ A \leftarrow AP$	Aux, Common
	$A \leftarrow QAP$	Aux, Common
Diagonal extract	$d \leftarrow \mathrm{diag}(A)$	?

Table 1.7: Data Movement with Matrices

Bibliography

[1] *The EM Symmetric Sparse Supernodal Format*. Elegant Mathematics, Inc., Bothell, Washington, 1993.

[2] *The EM Unsymmetric Sparse Supernodal Format*. Elegant Mathematics, Inc., Bothell, Washington, 1993.

[3] M. Aboelaze, N. Chrisochoides, and E. Houstis. The Parallelization of Level 2 and 3 BLAS Operations on Distributed Memory Machines. Technical Report CSD-TR-91-007, Purdue University, West Lafayette, IN, 1991.

[4] R. Agarwal, S. Balle, F. Gustavson, M. Joshi, and P. Palkar. A Three-Dimensional Approach to Parallel Matrix Multiplication. *IBM Journal of Research and Development*, 39(5):575–582, 1995.

[5] R. Agarwal, F. Gustavson, and M. Zubair. A High Performance Matrix Multiplication Algorithm on a Distributed-Memory Parallel Computer, Using Overlapped Communication. *IBM Journal of Research and Development*, 38(6):673–681, 1994.

[6] E. Anderson, Z. Bai, C. H. Bischof, J. Demmel, J. J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. C. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, PA, USA, 2nd edition, 1995. (Also available in Japanese, published by Maruzen, Tokyo, translated by Dr Oguni).

[7] P. Bangalore. The Data-Distribution-Independent Approach to Scalable Parallel Libraries. Master's thesis, Mississippi State University, 1995.

[8] R. E. Bank and D. J. Rose. Marching algorithms for elliptic boundary value problems. the constant coefficient case. *SIAM Journal of Numerical Analysis*, 14:792–829, 1977.

[9] Randolph E. Bank and Craig C. Douglas. Sparse matrix multiplication package (SMMP). *Advances in Computational Mathematics*, 1, 1993.

[10] B. Bialecki and G. Fairweather. Matrix decomposition algorithms in orthogonal spline collocation for separable elliptic boundary value problems. *SIAM J. Sci. Comput.*, 16:330–347, 1995.

[11] J. Bilmes, K. Asanovic, C. Chin, and J. Demmel. Optimizing matrix multiply using phipac: a portable, high-performance, ansi c coding methodology. In *Proceedings of the International Conference on Supercomputing*. ACM SIGARC, 1997. Vienna, Austria.

[12] R. Bisseling and J. van der Vorst. Parallel Triangular System Solving on a mesh network of Transputers. *SIAM Journal on Scientific and Statistical Computing*, 12:787–799, 1991.

[13] L. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. *ScaLAPACK Users' Guide*. SIAM, Philadelphia, PA, 1997.

[14] R. Brent and P. Strazdins. Implementation of BLAS Level 3 and LINPACK Benchmark on the AP1000. *Fujitsu Scientific and Technical Journal*, 5(1):61–70, 1993.

[15] J. Choi, J. Dongarra, S. Ostrouchov, A. Petitet, D. Walker, and R. C. Whaley. A Proposal for a Set of Parallel Basic Linear Algebra Subprograms. In J. Dongarra, K. Masden, and J. Waśniewski, editors, *Applied Parallel Computing*, pages 107–114. Springer Verlag, 1995. (also LAPACK Working Note No.100).

[16] J. Choi, J. Dongarra, and D. Walker. PUMMA: Parallel Universal Matrix Multiplication Algorithms on Distributed Memory Concurrent Computers. *Concurrency: Practice and Experience*, 6(7):543–570, 1994. (also LAPACK Working Note No.57).

[17] J. Choi, J. Dongarra, and D. Walker. PB-BLAS: A Set of Parallel Block Basic Linear Algebra Subroutines. *Concurrency: Practice and Experience*, 8(7):517–535, 1996.

[18] A. Chtchelkanova, J. Gunnels, G. Morrow, J. Overfelt, and R. van de Geijn. Parallel Implementation of BLAS: General Techniques for Level 3 BLAS. Technical Report TR95-49, Department of Computer Sciences, UT-Austin, 1995. Submitted to Concurrency: Practice and Experience.

[19] J. Demmel, J. R. Gilbert, and X. S. Li. An asynchronous parallel supernodal algorithm for sparse Gaussian elimination, February 1997. To appear in SIAM J. Matrix Anal. Appl., special issue on Sparse and Structured Matrix Computations and Their Applications (Also LAPACK Working Note 124).

[20] J. Demmel and X. Li. Faster numerical algorithms via exception handling. *IEEE Trans. Comput.*, 43:983–992, 1994.

[21] D. S. Dodson. Corrigendum: Remark on "Algorithm 539: Basic Linear Algebra Subroutines for FORTRAN usage". *ACM Trans. Math. Software*, 9:140, 1983. (See also [38] and [22]).

[22] D. S. Dodson and R. G. Grimes. Remark on algorithm 539: Basic Linear Algebra Subprograms for Fortran usage. *ACM Trans. Math. Software*, 8:403–404, 1982. (See also [38] and [21]).

[23] D. S. Dodson, R. G. Grimes, and J. G. Lewis. Sparse extensions to the FORTRAN Basic Linear Algebra Subprograms. *ACM Trans. Math. Software*, 17:253–272, 1991. (Algorithm 692).

[24] J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling. A set of Level 3 Basic Linear Algebra Subprograms. *ACM Trans. Math. Software*, 16:1–28, 1990. (Algorithm 679).

[25] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson. An extended set of FORTRAN Basic Linear Algebra Subprograms. *ACM Trans. Math. Software*, 14:1–32, 399, 1988. (Algorithm 656).

[26] I. S. Duff, A. M. Erisman, and J. K. Reid. Direct methods for sparse matrices. In *Monographs on Numerical Analysis*. Oxford University Press, New York, 1986.

[27] Iain Duff, Michele Marrone, and Giuseppe Radicati. A proposal for user level sparse BLAS. Technical Report TR/PA/92/85, CERFACS, 1992.

[28] A. Elster. Basic Matrix Subprograms for Distributed Memory Systems. In D. Walker and Q. Stout, editors, *Proceedings of the Fifth Distributed Memory Computing Conference*, pages 311–316. IEEE Press, 1990.

[29] R. Falgout, A. Skjellum, S. Smith, and C. Still. The Multicomputer Toolbox Approach to Concurrent BLAS and LACS. In *Proceedings of the Scalable High Performance Computing Conference SHPCC-92*. IEEE Computer Society Press, 1992.

[30] Message Passing Interface Forum. MPI: A Message Passing Interface Standard. *International Journal of Supercomputer Applications and High Performance Computing*, 8(3–4), 1994.

[31] G. Fox, S. Otto, and A. Hey. Matrix Algorithms on a Hypercube I: Matrix Multiplication. *Parallel Computing*, 3:17–31, 1987.

[32] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders. Methods for modifying matrix factorizations. *Math. Comp.*, 28:505–535, 1974.

[33] F. G. Gustavson. Two fast algorithms for sparse matrices: Multiplication and permuted transposition. *ACM Transactions on Mathematical Software*, 4(3):250–269, 1978.

[34] M. Heath and C. Romine. Parallel Solution Triangular Systems on Distributed Memory Multiprocessors. *SIAM Journal on Scientific and Statistical Computing*, 9:558–588, 1988.

[35] M. A. Heroux. A proposal for a sparse BLAS toolkit. Technical Report TR/PA/92/90, CERFACS, 1992.

[36] S. Huss-Lederman, E. Jacobson, A. Tsao, and G. Zhang. Matrix Multiplication on the Intel Touchstone DELTA. *Concurrency: Practice and Experience*, 6(7):571–594, 1994.

[37] L. Kaufman and D. D. Warner. High-order, fast direct methods for separable elliptical equations. *SIAM Journal of Numerical Analysis*, 21:672–695, 1984.

[38] C. L. Lawson, R. J. Hanson, D. Kincaid, and F. T. Krogh. Basic Linear Algebra Subprograms for FORTRAN usage. *ACM Trans. Math. Software*, 5:308–323, 1979. (Algorithm 539. See also [22] and [21]).

[39] G. Li and T. Coleman. A Parallel Triangular Solver for a Distributed-Memory Multiprocessor. *SIAM Journal on Scientific and Statistical Computing*, 9(3):485–502, 1988.

[40] G. Li and T. Coleman. A New Method for Solving Triangular Systems on Distributed-Memory Message-Passing Multiprocessor. *SIAM Journal on Scientific and Statistical Computing*, 10(2):382–396, 1989.

[41] W. Lichtenstein and S. L. Johnsson. Block-Cyclic Dense Linear Algebra. *SIAM Journal on Scientific and Statistical Computing*, 14(6):1259–1288, 1993.

[42] K. Mathur and S. L. Johnsson. Multiplication of Matrices of Arbitrary Shapes on a Data Parallel Computer. *Parallel Computing*, 20:919–951, 1994.

[43] O. McBryan and E. van de Velde. Matrix and Vector Operations on Hypercube Parallel Processors. *Parallel Computing*, 5:117–126, 1987.

[44] R. Pozo. Template numerical toolkit (tnt), 1995. `http://math.nist.gov/tnt`.

[45] Y. Saad. SPARSKIT: a basic tool kit for sparse matrix computations. Preliminary Version.

[46] P. Strazdins. A High Performance, Portable Distributed BLAS Implementation. In *Proceedings of the Sixth Parallel Computing Workshop*, Fujitsu Parallel Computing Center, 1996.

[47] Thinking Machines Corporation. *CMSSL for Fortran*, 1990.

[48] R. van de Geijn. *Using PLAPACK: Parallel Linear Algebra Package.* The MIT Press, Cambridge, Massachusetts, 1997.

[49] R. van de Geijn and J. Watts. SUMMA: Scalable Universal Matrix Multiplication Algorithm. Technical Report UT CS-95-286, LAPACK Working Note No.96, University of Tennessee, 1995.

[50] T. et al. Yang. Distributed memory parallel sparse lu. Ucsb preprint, University of California, Santa Barbara.