1984

BLAS, Linear, Algebra Modules and Supercomputers PARVEC Workshop #4

John R. Rice
Purdue University, jrr@cs.purdue.edu

Report Number:
84-501

BLAS, LINEAR ALGEBRA MODULES AND SUPERCOMPUTERS PARVEC WORKSHOP #4 •

John R. Rice
Computer Science
Purdue University

December 15, 1984
CSD TR 501

Abstract

About 20 people met on October 29-30 to discuss extensions to the Basic Linear Algebra Subroutines (BLAS), extensions motivated in large part by new supercomputer architectures. This report summaries the presentations made and presents the areas of consensus and open issues from the discussions that took place at the workshop.

• Workshop supported in part by Army Research Office Contract DAAG29-84-M-0194 and Office of Naval Research Contract NOOO14-84-K-0012

CONTENTS

1. SUMMARY AND PRINCIPAL CONCLUSIONS

On October 29 and 30, about 20 people met at Purdue University to consider extensions to the Basic Linear Algebra Subroutines (BLAS) and linear algebra software modules in general. The need for these extensions and new sets of modules is largely due to the advent of new supercomputer architectures which make it difficult for ordinary coding techniques to achieve even a significant fraction of the potential computing power. The participants represented most active groups in linear algebra software and were about equally divided among industry, universities and government laboratories. The workshop was organized by the Purdue Center for Parallel and Vector Computing (PARVEC) and supported by the Army Research Office and Office of Naval Research.

The workshop format was one of informal presentations with ample discussions followed by sessions of general discussions of the issues raised. This report is a summary of the presentations, the issues raised, the conclusions reached and the open issue discussions. Each participant had an opportunity to comment on this report, but it also clearly reflects the author's filtering of the extensive discussions.

Section 2 describes seven proposals for linear algebra software modules and Section 3 describes four presentations on the use of such modules. Discussion summaries are given next; Section 4 for those where near consensus was reached and Section 5 where the issues were left open. The 21 participants are listed at the end.

The principal conclusions reached in this workshop were:

(a) The *motivation for the BLAS* is both increased code efficiency and code clarity. Speed ups by 10 or 100 are reported for existing vector machines and even more advantage is expected as architectures grow more complex.

(b) The *audience for the BLAS* are math software experts and the person with a massive, compute-bound application.

(c) There are three *directions for extending the BLAS* now: Matrix-vector operations (Dongarra-Hammarling proposal), Sparse vector operations (Dodson-Lewis proposal) and multiple vector operations (Komzsik proposal).

(d) An aggressive and systematic effort must be made to achieve a *de facto standard for the BLAS* extension.

(e) A *test program* is a very important part of any extension.

The ten open issues identified are, very briefly:

(a) How to name the BLAS?

(b) Should one include array operations that are not linear algebra operations?

(c) Should very high level operations be included?

(d) Should the BLAS have explicit machine dependency parameters?

(e) Do multiprocessor architectures require a special set of BLAS?

(f) How are machine implementations to be distributed systematically?

(g) Should abstract, programming language independent BLAS be defined?

(h) Should a general classification of linear algebra software be constructed?

(i) Can the linear algebra software modules be organized into a natural heirarchy?

(j) Who should pay for producing such software?

2. THE PROPOSALS

A number of detailed proposals for BLAS, extensions, higher level modules and testing were made. This section summarizes these, further detail is given in the references cited in the subsection titles. These references are not necessarily formal papers or reports, one may contact the authors for related printed material which might exist.

Some of the proposals presented at this workshop have been modified and combined in the report Tech. Memo. 41, Mathematics and Computer Science Division, Argonne Nat. Lab.: *A Proposal for an Extended Set of Fortran Basic Linear Algebra Subprograms*, by J. Dongarra, J. DuCroz, S. Hammarling and R. Hanson.

A. **A Proposal for an Extended Set of Basic Linear Algebra Subprograms.** Jack Dongarra and Sven Hammarling, 26 pages.

This paper presents proposals to extend the BLAS in the direction of matrix-vector operations (the original BLAS are all vector-vector operations). The 35 new operations are indicated in Figure 1 using the following naming conventions (adapted from LINPACK). Each name is of the form

t m m o o o

where

t = data type

S = REAL D = DOUBLE PRECISION

C = COMPLEX Z = DOUBLE COMPLEX

m m = matrix data structure

- GE - General matrix
- GB - General band matrix
- HU - Hermitian matrix stored in upper triangle
- HL - Hermitian matrix stored in lower triangle
- HP - Hermitian matrix stored in packed form
- SU - Symmetric matrix stored in upper triangle
- SL - Symmetric matrix stored in lower triangle
- SP - Symmetric matrix stored in packed form
- HB - Hermitian band matrix
- SB - Symmetric band matrix
- UT - Upper triangular matrix
- UP - Upper triangular matrix in packed form
- LT - Lower triangular matrix
- UB - Upper triangular band matrix

LB - Lower triangular band matrix

o o o = operation (2 or 3 characters)

MV - Matrix-vector product
MVT - Matrix-vector product, (conjugate) transpose of the matrix
R1 - Rank-one update
R2 - Rank-two update
SL - Solution of triangular equations
SLT - Solution of triangular equations, (conjugate) transpose of the matrix

complex	*real*	MV	MVT	R1	R2	SL	SLT
CGE	SGE	*	*	*			
CGB	SGB	*	*				
CHU	SSU	*		*	*		
CHL	SSL	*		*	*		
CHP	SSP	*		*	*		
CHB	SSB	*					
CUT	SUT	*	*			*	*
CUP	SUP	*	*			*	*
CLT	SLT	*	*			*	*
CUB	SUB	*	*			*	*
CLB	SLB	*	*			*	*

Figure 1. The proposed 35 matrix-vector extension of the BLAS. The rows indicate the matrix types involved and the columns the operations. The data type characters *C* and *S* may be replaced with *Z* and *D*, respectively.

To illustrate this naming scheme we give two examples:

Example: $y = \alpha Ax + y$

SUBROUTINE SGEMV(M, N, ALPHA, X, INCX, A, LDA, Y, INCY) carries out this assignment using the conventions

S : single precision (REAL) type
GE : general matrix structure
MV : matrix-vector product

where

M : row range of matrix A

N	:	column range of matrix A
ALPHA	:	constant in computation
X	:	input vector
INCX	:	increment in indices of X
A	:	input matrix
LDA	:	leading dimension of A in storage allocation
Y	:	input/output vector
INCY	:	increment in indices of Y

Example: $A = \alpha xyH + \bar{\alpha} yxH$

SUBROUTINE CHPR2(N, ALPHA, X, INCX, Y, INCY, AP) carries out this assignment using the conventions

C	:	complex type
HP	:	Hermitian matrix in packed form structure
R2	:	rank-two update operation

where

N	:	row and column range of the square matrix A
ALPHA	:	constant in the computation
X	:	input vector
INCX	:	increment in indices of X
Y	:	input vector
INCY	:	increment in indices of Y
AP	:	input/output matrix (in packed form)

This proposal contains a complete discussion of the specifications for the proposed set of extensions. It is noted that some desirable operations (e.g., rank-one update of a band matrix) can be obtained from the other proposed operations. There is also a discussion of the trade-offs between extending the BLAS in many directions and in keeping the size of the set reasonable. Details of implementations are given for three operations including variations suitable for different Fortran environments.

B. **Proposed Extensions to the Basic Linear Algebra Subprograms.** David Dodson and John Lewis, 24 slides and 15 pages.

This paper reviews the motivation, selection and implementations of the BLAS in general and then goes on to propose a set of sparse vector extensions to the BLAS. Several bodies of sparse matrix software have been examined carefully to ascertain the BLAS most likely to be used. It is concluded that:

(i) Sparse extensions are not needed for

ROTG,ROTM,ROTMG	:	Givens rotations
NRM2,ASUM	:	vector norms
SCAL,AMAX	:	scale and index of maximum value

SWAP : swap vectors

(ii) extensions are needed for binary operations between one sparse and one dense vector.

(iii) the increment argument should not be included for sparse vectors.

They then propose 8 sparse operations:

DOTI	:	sparse dot product
DOTCI	:	conjugated sparse dot product
DOTUI	:	unconjugated sparse dot product
AXPYI	:	scalar times sparse vector + vector
ROTI	:	sparse Givens rotation
SCTR	:	scatter packed sparse vector
GTHR	:	gather vector into packed sparse vector
GHNRZ	:	GTHR with vacated elements set to zero

These BLAS each come in four versions depending on the date type of the vector elements.

Detailed specifications are given for all the proposed routines.

The authors conclude that sparse matrix BLAS could be useful, but that this cannot be achieved until a *small* number of standard representations are adopted for sparse matrices.

C. **Remarks on "A Proposal for a New Set of Basic Linear Algebra Subroutines".** Richard Hanson, 9 pages.

This paper generally agrees with the proposal of Section 2A as far as the operations in the extension goes. It is mentioned that both transpose and conjugates transpose operations are needed.

A proposal is made to use the increment arguments of the BLAS to specify decreasing steps from the high index by setting the increments to negative values.

Several points are made about names:

(i) The names of the proposed extension, in following the LINPACK scheme, are of a different structure than the existing BLAS names.

(ii) The total number of names will be very large, perhaps over 300. This makes it difficult to identify a particular routine.

(iii) Fortran 77 allows us to adopt a weak form of keyword naming for arguments.

A "generic family" of subroutines for the matrix-vector affine operation $y = a*B*x + y$ is proposed as follows:

__MVA[__] (M,N,__A,__X,INCX,__Y,__B,IDOPEB, KEYWORD)

where

(a) The leading blank represents the data type (as proposed in 2A above).

(b) The trailing (optional) blank is missing except for COMPLEX types where it is 'C' for conjugated and 'U' for unconjugated.

(c) The blanks on A, X, Y and B also represent data types (this is for documentation, in use these are user supplied names).

(d) The argument IDOPEB is an INTEGER dope vector carrying the information about the matrix data structure.

(e) The argument KEYWORD is a CHARACTER variable that specifies the matrix data structure. Thus 'BG' or 'GEN-BAND' or etc. would indicate a general band matrix.

This is illustrated in extreme form when the BLAS are organized according to the operations performed:

Example: Dot product, named DOT

```
VALUE 1 = DOT( 'REAL', 'DENSE',... )
VALUE 2 = DOT( 'DOUBLE', 'SPARSE',... )
```

Example: Matrix-vector product, named MVPROD

```
CALL MVPROD( 'REAL', 'BAND-MATRIX', 'SPARSE',... )
CALL MVPROD( 'COMPLEX', 'SYMMETRIC-PACKED', 'DENSE',... )
CALL MVPROD( 'C', 'SYM-P', 'D',... )
```

These examples do not show full argument lists and they oversimplify the situation. Note that aliases can be used for lengthy keywords. Hanson's main point is that it is very difficult to encode 300 routine names into six character Fortran names and that character string arguments for the BLAS could provide increased naturalness in the names.

D. **Standard MSC/NASTRAN Kernels**. Louis Komzsik, 8 pages.

NASTRAN is a very large structural engineering system marketed by MacNeal-Schwendler Corp. (MSC). They are interested in establishing and using a general set of linear algebra modules supported by hardware vendors. This paper presents six routines of particular importance to their software. Four of these are existing BLAS (with different names). Two others perform multiple BLAS operations.

DOT2R carries out the dot product of two sets of M vectors of length N. It has a switch to either store or accumulate the inner products computed. XPY2R carries out a multiple SAXPY, N SAXPY operations are carried out on two sets of vectors. In standard matrix terms this is

$$Y = \mathrm{diag}[S]*X + Y$$

where X and Y are N by M matrices, S is a vector of length N and diag[S] is an N by N matrix with S on the diagonal.

The motivation for multiple-vector BLAS arises from block processing of very large problems. Even if the overall problem is sparse (banded), the blocks brought in from secondary memory tend to be essentially dense.

E. **Summary of Functions and Names of the BLAS Subprograms.** Phuong Vu, 3 pages.

IMSL is in the process of reorganizing their library and they are introducing systematically a set of extended BLAS. This paper summarizes the operations they find widely used in their library codes and gives them names extending the original BLAS naming conventions. They are also using a number of extensions just in the data types.

The new operations they will use are:

XPY	:	$Y = X + Y$	Add vectors X and Y
ADD	:	$X = X + S$	Add scalar S to vector K
SUB	:	$X = X - S$	Subtract scalar S from vector X
SUM	:	sum of values	Sum elements in vector
PROD	:	product of values	Product of elements in vector
MAX	:	maximum of values	Maximum of elements in vector
MIN	:	minimum of values	Minimum of elements in vector
SET	:	set values	Set all elements of vector to a scalar
HPROD	:	Hadamard product	Element by element product of two vectors
XYZ	:	Triple product	Weighted dot product with three vectors
HOUAP	:	Householder transformation	Apply Householder transformation

They also are using the sparse BLAS extension discussed above.

F. **On Testing the BLAS.** Richard Hanson, 9 slides.

A strategy is proposed to develop a self-contained testing program for the BLAS. The objectives are to make the test robust and somewhat machine independent. One idea is for the test to compute the desired result by a robust (but perhaps inefficient method) and compare with the proposed BLAS implementation. A grade is given on the basis of the relative percentage error for a set of internally generated data. An example of this grade is

$$g = \log_b [\, |(s-t)/s| / (\sqrt{n}\,\eta)]$$

where

t	=	true value	s	=	computed value
n	=	vector length	η	=	arithmetic epsilon
b	=	machine base			

An analysis of this grade not only provides confidence for correct values, but it also gives clues to the source of errors that are present.

Testing is also discussed in the papers of 2A and 2C above.

G. **Do We Need Sparse BLAS at all?** Ian Duff, 3 slides.

Duff plays the devil's advocate and observes:

(i) There should be no sparse BLAS at the matrix level because sparse data structures are not well established.

(ii) The sparse BLAS for common operations like SAXPY and DOT are often not the best because in-line, machine tailored code should be used.

(iii) Some pre-emption of names has already occurred by the Cray and CDC libraries. For example, they use GATHER and Q8VGATHR, respectively, for the proposed GATH.

3. THE USE OF LINEAR ALGEBRA MODULES

The use of BLAS and linear algebra modules is mentioned in many of the papers in Section 2 and is also a central topic of the workshop discussions summarized in Section 4. Three papers at the workshop which consider this topic are summarized here.

A. **Applicability of an Extended Set of BLAS.** Jeremy DuCroz, 12 slides and 9 pages.

This paper demonstrates the wide applicability of the BLAS extension discussed in Section 2A. The presentation has two forms. First, for selected LINPACK routines, it is shown how they can be rewritten using the extended BLAS so as to provide much shorter and cleaner code. It is also noted that, at this new, higher module level, one has the potential for much more efficient vectorization on existing machines.

Second, the single precision LINPACK and EISPACK routines are examined to note the number of places the new BLAS can be used profitably. The frequency of use of the 35 relevant routines is:

No. of uses	0	1	2	3	7	13
No. of BLAS	14	9	7	2	2	1

B. **Matrix Multiplication on MAX Hardware.** Steve Oslon, 16 pages.

This working note analyzes in some detail the implementation of matrix multiplication using the Floating Point Systems 164-MAX array processor. Four implementations are considered using FPS linear algebra modules. The principal conclusion reached is that higher level modules, similar to the multiple BLAS of Section 2D, are essential to obtaining maximum - or even good - performance on this machine.

C. **Introductory Remarks.** Jack Dongarra, 6 slides.

These remarks note that the existing BLAS do not allow one to achieve anything close to maximum performance on current vector computers. Ordinary Gauss elimination is examined and the reasons for this disappointing performance shown. The computation is then recast using a higher level matrix-vector module and near peak performance is achieved. For solving a system of 100 linear equations, the execution rate is raised to close to Cray maximum rate using this approach.

D. **Performance of a Subroutine Library on Vector Processing Machines.** C. Daly and J.J. DuCroz, 13 pages.

This paper describes at length a strategy to achieve high levels of performance on vector processing machines by using selected linear algebra modules in the NAG library routines. One objective of this strategy is to achieve this high performance without perturbing the user interface or library structure. The strategy used two classes of matrix-vector kernel routines (BLAS) (x and y are vectors, A, L and U are matrices).

(i) Matrix-vector products

$y = Ax + y$	
$y = A^T x + y$	
$y = Ax + y$	(A = symmetric)
$y = Ly$	(L = lower triangular)
$y = L^T y$	(L = lower triangular)

(ii) Solution of triangular linear system

solve $Ly = x$	(L = lower triangular)
solve $L^T y = x$	(L = lower triangular)
solve $Uy = x$	(U = upper triangular)

The implementation of these kernels is discussed for four machines: Cray-1, Cray-XMP, Cyber 205 and FPS 164. Substantial improvements in performance were obtained for all machines. The following excerpt shows how a matrix-vector subroutine improves performance compared to using the existing DOT BLAS. The problem is to compute $y = Ax + y$ where A is a 320 by 320 real matrix.

	Megaflops using DOT	Megaflops using subroutine
Cray 1	24(17)	73
Cray XMP	45(30)	157
Cyber 205	6	110
FPS-614	3.4	5.9

The numbers in parentheses for the Cray machines are for cases where memory bank conflicts degrade performance.

4. DISCUSSIONS: AREAS OF CONSENSUS

Section 1 summarizes several areas of consensus that arose from the discussions at this workshop. In this section we provide more details on the views expressed. Note that consensus means general agreement rather than unanimous agreement and, further, the presentation here has been filtered by the author.

A. Motivation for the BLAS.

The BLAS and their extensions have two primary motivations:

(i) *Increased efficiency.* This is the primary motivation of the original BLAS and it is a much stronger one now that new architectures are appearing. For sequential machines, the existing BLAS might speed up computation by 20, 50 or even 100 percent. This gain is certainly very worthwhile, but it does not dominate in most users minds. For existing vector machines, the BLAS speed up some computations by factors of 10 or even 100. Such speed ups are very clear to the users and have created a much larger demand for the BLAS.

More complex architectures are to come and it will be even more imperative to give the user good software building blocks. One instance was reported where a run's cost went from $365 to $2 by simply rewriting the Fortran DO-loops in a matrix multiplication.

(ii) *Increased Code Clarity.* The BLAS are used in large bodies of code (such as the IMSL and NAG libraries) to provide shorter, clearer and more uniform code. This is the most important motivation for the IMSL and NAG libraries to incorporate an extended set of BLAS into their libraries in a systematic way.

These two motivations for using the BLAS are reinforced by the following third one:

(iii) *Program Portability.* One major installation reported that they had to provide the BLAS on their IBM machines after they installed a Cray vector machine. Users developing programs on the IBM machines for Cray use needed the IBM BLAS. As new architectures become more common, and thus the necessity of using the BLAS more widespread, the desire for program portability will lead to the BLAS being more widely used even on sequential machines.

B. Audience of the BLAS.

Several candidate audiences were discussed: the numerical linear algebra expert, the math software expert, the general scientific programmer, the applications programmer or scientist with a massive, compute-bound problem and the casual user. It was quickly agreed that casual user and general scientific programmers are unlikely to use the BLAS; it is obvious that the numerical linear algebra and math software experts will make heavy use of them. Considerable discussion led to the conclusion that the BLAS audience also properly includes the user with a massive, compute-bound problem involving linear algebra. This conclusion means that the BLAS should be designed and distributed with these classes of users in mind. There was a minority opinion that the proper audience for the BLAS are only the experts in numerical linear algebra software.

C. Directions for Extending the BLAS.

There are three directions for extending the BLAS which are ready. These are:

(i) *Matrix-Vector Operations.* These are described in some detail in the Dongarra-Hammarling proposal (Section 2A) and there received wide support as being appropriate for inclusion in an extension of the BLAS. These operations include things like multiplying a vector by an upper triangular matrix. These operations had been considered for some time by most of the participants in the workshop.

(ii) *Sparse Vector Operations.* These are described in some detail in the Dodson-Lewis proposal (Section 2B) and these also received wide support for

inclusion. These operations include the scatter-gather of several current vector machines plus many operations involving one sparse and one dense vector. Operations on two sparse vectors were judged to be unnecessary. It was generally agreed that it is premature to include any operations involving sparse matrices. These operations had also been considered for some time by most participants.

(iii) *Multiple Vector-Vector Operations.* These operations are discussed in the Komzsik proposal (Section 2D) and also are part of the uses discussed by Oslon (Section 3B). These operations are essentially the repeated application of the existing BLAS to large sets of vectors. For example, one might take inner products of all the rows of one submatrix with all the columns of another submatrix. These operations had not been considered by most participants before the workshop. Thus, their was considerable discussion of their value and, by the end, a majority (but not all) felt they should be included. Their value is in allowing one to move very large blocks of data in preparation for large sets of vector-vector operations. This seems important on several machine architectures and should not cause inefficiencies on others where it is not important.

D. **Standardization Procedure.**

The following are the steps agreed upon as appropriate for informally standardizing the extensions to the BLAS:

1. Obtain a reasonable concensus among those people actively interested in the area. This workshop is one step in this process. There have been previous, less formal, discussions in the summer of 1984 at the Gatlinburg, IFIP WG2.5 and SIAM meetings.
2. Prepare a specific proposal. This will probably be done by Jack Dongarra, Sven Hammarling, John Lewis and others.
3. Circulate this proposal widely. Revise it in the light of comments received.
4. Publish an algorithm in the ACM Trans. on Mathematical Software which includes:
   (a). A simple, direct Fortran 77 implementation of each subroutine.
   (b). A program to test the correctness of installation and machine implementations of the extended BLAS.
   (c). Some interesting machine language implementations.
5. Obtain endorsements, approvals and support from professional societies, working groups, manufacturers, software houses and others in the mathematical software area.
6. Have the TOMS algorithm become a "center" of the BLAS subroutines by allowing new machine implementations to be published as remarks and then appended to the extended BLAS algorithm code.

E. **Test Program.**

It was agreed that a good test program for the BLAS is both extremely valuable and difficult to do. The current test program has some shortcommings, the most significant of which is that it always uses very short vectors. Of course, when it was written, vector machines were not in use. It was reported that one manufacturer's implementation of the BLAS had an error which would have been detected had the existing test program been used. Since the test program was not used, an erroneous

BLAS routine was widely used for almost a year. Note that this program is to be part of the TOMS algorithm; Richard Hanson and others will probably prepare it.

5. DISCUSSION: OPEN ISSUES

There were numerous areas of disagreement, confusion and incomplete information. Some of these were due to new and uncertain computing environments (e.g., the effect of multiprocessor system or the Ada language). Others were simple differences in viewpoints or preferences. Ten such areas are listed here and briefly discussed.

A. Naming Conventions.

Different strategies for naming the BLAS were discussed at length without a clear concensus emerging. The principal factors involved or points made were:

(i) There will be a very large number of BLAS.

(ii) Six character Fortran names for large sets of programs become incomprehensible.

(iii) Compatibility with earlier names is important.

(iv) A simple, easy to remember naming system is important.

(v) Certain "parameters" of the BLAS can be moved from the subroutine names to keyword-type arguments.

(vi) Names independent of any programming language might be feasible.

B. Other Directions for Extending the BLAS.

Several other directions for extending the BLAS were discussed inconclusively. We list them in approximate order of the level of discussion and briefly summarize the points of view expressed.

(i) *Nonlinear-Algebra Operations.* There are some vector and array operations that occur widely in mathematical software that are not, nevertheless, parts of common linear algebra programs. Examples are the operations of summing the elements of a vector or sequence and the weighted inner product. Some feel that these occur throughout important sets of mathematical software (e.g., the IMSL and NAG libraries) and should also be included in the BLAS. Others feel that there is already enough complexity in trying to cover the linear algebra needs. Fairly strong views were expressed on both sides.

(ii) *Higher-level Operations.* Part of the extension of the BLAS can be viewed as raising the BLAS level from vector-vector operations to matrix-vector operations or even to matrix-matrix operations. One could attempt to define a hierarchy of software modules from simple vector operations to complete solutions of linear systems or eigenvalue problems. The emerging concepts of Fortran 8X modules and Ada packages illustrate the need for such hierarchies. There is, however, considerable overlap with existing software sets (LINPACK, EISPACK and the relevant chapters of the IMSL and NAG libraries). No one volunteered to pursue this direction further.

C. Machine, Architecture and System Dependencies.

The discussion of supercomputers made it clear that radical changes may appear in our computing environments. A given machine may have its computational architecture changed, perhaps dynamically. A given architecture may have different programming, I/O and communications system. Three aspects of this situation were discussed:

(i) *Explicit Machine Dependency.* One can visualize the BLAS having arguments which give some information about the current computing environment. Such information could provide substantial benefits. Some realistic examples were presented, but no clear picture emerged. Many expressed the hope that such arguments would not be needed.

(ii) *Multiprocessor Versions.* The participants seemed comfortable with vector architectures and knowledgeable about the forthcoming multiprocessor architectures. There was, however, no clear concensus on whether special BLAS would be needed or beneficial for multiprocessors. Many hope that the effects of multiprocessors can be handled internally as are other machine dependencies. The discussions reflected the lack of experience with multiprocessors and uncertainity about how they will be used.

(iii) *Distribution of Machine Implementation.* Publishing machine implementation as Remarks in TOMS and then distributing the code along with the BLAS algorithm received generally favorable support. But what if the implementor (e.g., a manufacturer or software house) does not want to put the code in the public domain? No generally acceptable mechanism for including the implementation in the "standard BLAS", even to note its availability, was presented.

D. **Abstract BLAS and Other Languages.**

Associated with the idea of higher-level operations is the idea to provide an abstract, language independent specification of the BLAS. Three subtopics discussed were:

(i) *Abstract BLAS*: This would provide a mathematical (programming language independent) specification of the input and output for all the BLAS.

(ii) *Classification Schemes*: This would provide a framework to include the existing BLAS, the proposed extensions and future extensions of various types.

(iii) *Heirarchy of BLAS.* The levels of vector-vector, matrix-vector, matrix-matrix and "higher" are easy to understand. The hierarchy would make this division into levels more precise and refine the "higher" level.

Several thought these tasks were feasible, but no one volunteered to attempt any of them.

E. **The Economics of the BLAS.**

It is clear that creating the BLAS extension, developing the test program and making several high quality implementations is a formidable and expensive task. Once the BLAS become established as cost-effective software, we can hope that some (most?) manufacturers will implement them. It is much less clear who would be able to justify the initial effort. Both the IMSL and NAG libraries state that their interests primarily come from their internal benefits and not from any increased

sales. Neither see the BLAS as a viable separate software product.

It was remarked that machine manufacturers have a track record of "casual" implementations of software of this nature. The situation for the BLAS is similar to other "basic mathematical" software: its use is too diffuse to motivate the usual software sources to do an excellent job of it.

6. WORKSHOP PARTICIPANTS

The following people participated in the discussions at this workshop.

Aird, Thomas J.	IMSL	Houston, TX
Blakemore, Karen	Convex Computer Corp.	Richardson, TX
Dodson, David S.	Boeing Computer Services	Tukwila, WA
Dongarra, Jack	Argonne National Laboratory	Argonne, IL
DuCroz, Jeremy	NAG	Oxford, OX2 7DE ENGLAND
Duff, Iain S.	Comp. Sci. & Sys. Div.	Oxfordshire OX11 ORA ENGLAND
Dyksen, Wayne R.	Purdue University	W. Lafayette, IN
Gannon, Dennis B.	Purdue University	W. Lafayette, IN
Hammarling, Sven	NAG	Oxford, OX2 7DE ENGLAND
Hanson, Richard J.	Sandia National Lab.	Albuquerque, NM
Houstis, Elias	Purdue University	W. Lafayette, IN
Komzsik, Louis	MacNeil-Schwendler	Los Angeles, CA
Lawson, Charles	Jet Propulsion Lab.	Pasadena, CA
Luk, Franklin	Cornell University	Ithaca, NY
Mehrotra, Piyush	Purdue University	W. Lafayette, IN
Melhem, Rami	Purdue University	W. Lafayette, IN
Oslon, Steve	Floating Point Systems	Portland, OR
Parlett, Beresford N.	Univ. of California	Berkeley, CA
Rice, John R.	Purdue University	W. Lafayette, IN
Sorensen, Danny	Argonne National Lab.	Argonne, IL
Vu, Phuong	IMSL	Houston, TX