LAPACK Working Note 37
Two Dimensional Basic Linear Algebra Communication Subprograms*

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

October 28, 1991
Proposed Standard

Abstract

In this paper, we describe extensions to a proposed set of linear algebra communication routines for communicating and manipulating data structures that are distributed among the memories of a distributed memory MIMD computer. In particular, recent experience shows that higher performance can be attained on such architectures when parallel dense matrix algorithms utilize a data distribution that views the computational nodes as a logical two dimensional mesh. The motivation for the BLACS continues to be to increase portability, effiiency and modularity at a high level. The audience of the BLACS are mathematical software experts and people with large scale scientific computation to perform. A systematic effort must be made to achieve a *de facto* standard for the BLACS.

1 Introduction

This report outlines a second attempt to define a proposed set of linear algebra communication routines for the specification and manipulation of data structures that arise when linear algebra algorithms are implemented on distributed memory multicomputers. The scope of this set of routines is intentionally limited. We do not view this package as a complete communication library for all applications. It is intended primarily for software developers and to a lesser extent for experienced applications programmers in the area of numerical linear algebra. We see these routines complementing the existing Level 1, 2, and 3 BLAS, providing tools for the implementation of numerical algorithms in linear algebra for distributed memory MIMD machines.

It is important to realize what kind of algorithms inspired the BLACS as they stand. They are being developed as part of our effort to implement a subset of the LAPACK library on

*This project was supported in part by the National Science Foundation Science and Technology Center Cooperative Agreement No. CCR-8809615 and the Applied Mathematical Sciences subprogram of the 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.

distributed memory MIMD architectures [4, 5]. In implementing these routines, we decided to stress minimal change to the algorithms and codes, as well as portability of the final product. The codes are written in FORTRAN 77 in the SPMD (Single Program Multiple Data) paradigm.

Initially, matrices were mapped to nodes using (column) panel-wrapped storage, where a matrix is partitioned into panels of constant width that are wrapped onto the p nodes so that panels $p+ki$, $k = 0, 1, \ldots$, are assigned to nodes with index i = 1, [5, 4, 3, 2]. More recently, we have been experimenting with a data mappings that views the processing nodes as a two dimensional grid [1, 0]. Considerable gain in performance has been observed on the Intel Touchstone Delta and iPSC/860 systems. Typically, performance increase by a factor of two or better for similar sized problems. This has required us to rethink how the BLACS may be defined.

2 Parallel Computing Model

We assume that our multicomputer consists of p nodes, which are logically configured as a rectangular mesh, indexed by row and column index (i, j), where $0 \leq i < M$, $0 \leq j < N$ and $M \times N = p$. Each is equipped with CPU and local memory. The nodes are connected by some communication network that allows point-to-point communication. Messages that arrive over the network are buffered until absorbed by an appropriate call to a receive routine.

In our approach, it is actually not necessary to restrict the model to one process per node. Indeed, whenever we refer to a node, this could just as easily be a process, where many processes could be assigned to a single node.

3 Naming Conventions

In this section, we briefly preview the current set of primitives that are part of the 2D BLACS.

All names of BLACS subroutines are specified by at most seven letters. Their general form is:

`_XXYY2D` or `_GZZZ2D`

The first form is used for communication routines, the second for global operators. The position marked by `_` indicates the type of data to be communicated by the routine. Possible choices are:

`I`	`INTEGER`
`S`	`REAL`
`D`	`DOUBLE PRECISION`
`C`	`COMPLEX`
`Z`	`COMPLEX*16` or `DOUBLE COMPLEX`

3.1 Communication Primitives

The general form of names of communication primitives is

`_XXYY2D`

The position marked by `XX` indicates the data structure to be handled. Possible choices are:

`GE`	General rectangular matrix
`TR`	Trapezoidal matrix

The positions marked by YY indicate the function of the routine. Possible choices are:

SD	Send a message
RV	Receive a message
BS	Broadcast a message
BR	Receive a broadcast message

3.2 Global Operators

The general form of names of communication primitives is

_GZZZ2D

Combine routines for performing global "maximum", "minimum" and "summation" are provided. The position ZZZ indicates the operation:

MAX	Compute the maximum
MIN	Compute the minimum
SUM	Compute the summation

4 Argument List

4.1 Communication Primitives

The calling sequences for the proposed communication primitives are:

```
_GESD2D ( SCOPE, TOP,            M, N, A, LDA, IRDEST, ICDEST, MSGID )
_GERV2D ( SCOPE, TOP,            M, N, A, LDA, IRSRC,  ICSRC,  MSGID )
_GEBS2D ( SCOPE, TOP,            M, N, A, LDA,                 MSGID )
_GEBR2D ( SCOPE, TOP,            M, N, A, LDA, IRSRC,  ICSRC,  MSGID )

_TRSD2D ( SCOPE, TOP, UPLO, DIAG, M, N, A, LDA, IRDEST, ICDEST, MSGID )
_TRRV2D ( SCOPE, TOP, UPLO, DIAG, M, N, A, LDA, IRSRC,  ICSRC,  MSGID )
_TRBS2D ( SCOPE, TOP, UPLO, DIAG, M, N, A, LDA,                 MSGID )
_TRBR2D ( SCOPE, TOP, UPLO, DIAG, M, N, A, LDA, IRSRC,  ICSRC,  MSGID )
```

The function of the parameters depends largely on whether the routine sends data (_XXSD2D and _XXBS2D) or receives (_XXRV2D and _XXBR2D) data.

Parameters:

SCOPE	Scope of operation. Limited to ROW, COLUMN, or ALL. Input.
TOP	Network topology to be emmulated during communication. Input.
UPLO	Specifies if matrix is stored as Lower or Upper trapezoidal matrix. _TRSD2D/BS2D: Input; TRRV2D/BR2D: Output.
DIAG	Specifies if unit diagonal. TRSD2D/BS2D: Input; TRRV2D/BR2D: Output.
M	Row dimension of matrix. _XXSD2D/BS2D: Input; XXRV2D/BR2D: Output.
N	Column dimension of matrix. _XXSD2D/BS2D: Input; XXRV2D/BR2D: Output.

A	Array of data to be sent. XXSD2D/BS2D: Array of data to be sent. _XXRV2D/BR2D: Array where data is to be received.
LDA	Leading dimension of A. Input.
IRDEST	_XXSD2D/BS2D: Row index of destination node. Input.
ICDEST	_XXSD2D/BS2D: Column index of destination node. Input.
IRSRC	_XXRV2D/BR2D: Row index of source node. Output.
ICSRC	_XXRV2D/BR2D: Column index of source node. Output.
MSGID	Identifier of message being send/received. Input.

4.2 Global Operators

The calling sequences for the proposed global operators primitives are:

```
_GMAX2D ( SCOPE, TOP, M, N, A, LDA, IRA, ICA, LDIA, IRDEST, ICDEST )
_GMIN2D ( SCOPE, TOP, M, N, A, LDA, IRA, ICA, LDIA, IRDEST, ICDEST )
_GSUM2D ( SCOPE, TOP, M, N, A, LDA,               IRDEST, ICDEST )
```

Parameters:

SCOPE	Scope of operation. Limited to 'ROW', 'COLUMN', or 'ALL'. Input.
TOP	Network topology to be emmulated during communication. Input.
M	Row dimension of matrix being compared/summed. Input.
N	Column dimension of matrix being compared/summed. Input.
A	Input: Matrix of values being compared/summed (elementwise). Output: Matrix of results.
LDA	Leading dimension of matrix. Input.
IRA	Integer array indicating the row index of node that provided maximum/minimum. Output.
ICA	Integer array indicating the column index of node that provided maximum/minimum. Output.
LDIA	Leading dimension of integer arrays. Input.
IRDEST	Row index of node on which result it to be accumulated. On all other nodes X and IX are overwritten with intermediate results. IRDEST=-1 indicates the result is to be left on all nodes. Input.
ICDEST	Column index of node on which result it to be accumulated. On all other nodes X and IX are overwritten with intermediate results. Input.

5 Communicating in 2D

The proposed calling sequences now include parameters that indicate the scope of the operation. In other words, a subset of nodes of the mesh can be targeted, depending on the value of SCOPE.

Value	Meaning
'C'	Only nodes in same column are involved
'R'	Only nodes in same row are involved
'A'	All nodes are involved

In addition, we allow a parameter TOP to be passed to the communication primitives and global operators. This allows the user some control over how the communication is to be handled.

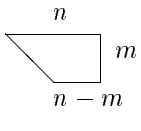
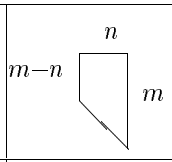
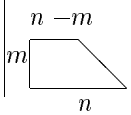
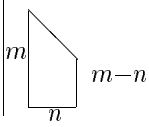
6 Trapezoidal Matrices

In addition to being able to send general (rectangular) arrays, the BLACS allow the user to specify that only a trapezoidal portion of the array is to be referenced. The arguments that specify these options are character arguments with the names UPLO and DIAG.

UPLO is used by the trapezoidal matrix routines to specify whether the upper or lower trapezoid is being referenced as follows:

Value	Meaning
'U'	Upper trapezoid
'L'	Lower trapezoid

The shape of the trapezoid to be sent is determined by M and N:

UPLO	M ≤ N	M > N
'U'		
'L'		

DIAG is used by the trapezoidal matrix routines to specify whether or not the matrix has ones on the diagonal, as follows:

Value	Meaning
'U'	Unit trapezoidal
'N'	Non-unit trapezoidal

When DIAG is supplied as 'U' the diagonal elements are not referenced.

Thus, UPLO and DIAG have the same values and similar meanings as for the Level 2 and 3 BLAS.

7 Specification of the BLACS

Type and dimension for variables occurring in the subroutine specifications are as follows:

```
INTEGER      M, N, LDA, LDIA, IRDEST, ICDEST, IRSRC, ICSRC
INTEGER      IRA(LDIA,*), ICA(LDIA,*), MSGID
CHARACTER*1 UPLO, DIAG, SCOPE, TOP
```

For routines whose first letter is an S:

```
REAL         A(LDA,*)
```

For routines whose first letter is a D

```
DOUBLE PRECISION  A(LDA,*)
```

For routines whose first letter is a C:

```
COMPLEX     A(LDA,*)
```

For routines whose first letter is Z:

```
COMPLEX*16  A(LDA,*)
```

or

```
DOUBLE COMPLEX    A(LDA,*)
```

8 Discussion

In the design of all levels of BLAS, one of the main concerns is to keep both the calling sequences simple and the range of options limited, while at the same time maintaining sufficient functionality. This clearly implies a compromise, and a good decision is vital if the BLACS are to be accepted as a useful standard. In this section, we discuss the reasoning behind some of the decisions which we have made and indicate some issues that have not yet been resolved.

- **Parameters: Input vs. Output**
  The reader may have noted that parameters that specify the shape of data being communicated are output parameters for the receive functions. This allows data with structure unknown to the receiving node(s) to be sent. Possible applications include parallel implementations of the Bunch-Kaufman algorithm for factoring indefinite matrices. On the other hand, this decision requires a header to be sent with the data that indicates to the receiving node the shape of the data structure. Moreover, given the current calling sequences, it is difficult to determine when a buffer into which data is to be received will overflow, before it is too late.
- **`TOP` Parameter**
  Removing this parameter would improve simplicity of the BLACS, at the possible expense of performance.
- **Error Handling**
  When a BLACS routine detects an error, the standard BLAS error handling routine `XERBLA` will be called.
- **Untyped Communication**
  In our effort to implement the LU factorization using 2D BLACS, a major source of potential programming difficulty lies with choosing the message identifier. When we were limiting ourselves to one dimensional mappings of matrices to nodes, such identifiers could typically be conveniently chosen to have some relation to the iteration index. However, when two dimensional data mappings are used, the number of communications increase dramatically. Moreover, often such communication occur in several levels of subroutine calls. Keeping the message identifiers from interfering with each other has become a major concern.

  We propose to extend the BLACS to include a set of routines that are untyped, requiring no message identifier. Receiving a message would be accomplished by specifying the source.

Order of such messages would be strictly enforced. Moreover, typed and untyped messages could easily coexist, so BLACS routines that require message identifiers could continue to be used.

9 Conclusion

We do not claim to provide the definitive answer to everyone's communication needs. Indeed, our insistence on simplicity precludes that. We do believe that a set of standard communication calls for linear algebra applications provides a means for achieving portability and readability of code and a general framework in which to program distributed memory MIMD architectures.

We welcome input from the user community.

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK: A portable linear algebra library for high-performance computers*, in Proceeding puting '90, IEEE Computer Society Press, Los Alamitos, California, 1990, pp. 2–11.

[2] E Anderson, A Benzon, J. Dongarra, S Meliton, S Ostrouchov, B Tourancheau, and R. van de Geijn, *LAPACK for distributed memory architectures: progress report*, to appear in the proceedings of the Fifth SIAM Conference on Parallel Processing for Scientific Computing, Houstin, March 25-27, 1991.

[3] E Anderson, A Benzon, J. Dongarra, S Meliton, S Ostrouchov, B Tourancheau, and R. van de Geijn, *Basic Linear Algebra Communication Subprograms*. Sixth Distributed Memory Computing Conference Proceedings, IEEE Computer Society Press, 1991.

[4] J. Dongarra and S. 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] J. Dongarra and R. van de Geijn, *Reduction to condensed form for the eigenvalue problem on distributed memory architectures*, LAPACK Working Note 30, technical repor University of Tennessee, 1991.

[6] J. J. Dongarra, J. Du Croz, S. Hammarling, and I. Duff, *A set of level 3 basic linear algebra subprograms*, ACM Trans. Math. Soft., 16 (1990), pp. 1–17.

[7] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson, *An extended set of FORTRAN basic linear algebra subprograms*, ACM Trans. Math. Soft., 14 (1988), pp. 1–17.

[8] G. H. Golub and C. F. V. Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, second ed., 1989.

[9] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh, *Basic linear algebra subprograms for Fortran usage*, ACM Trans. Math. Soft., 5 (1979), pp. 308–323.

[10] R. van de Geijn, *Massively parallel LINPACK benchmark on the Intel Touchstone De and iPSC/860 systems*, Technical Report CS-91-28, University of Texas at Austin, Aug. 1991.

[11] R. van de Geijn, *On global combine operations*, LAPACK Working Note 29, Technical Report CS-91-129, University of Tennessee, 1991.

[12] R. van de Geijn, *Efficient global combine operations*, Sixth Distributed Memory Computing Conference Proceedings, IEEE Computer Society Press, 1991, pp. 291–294.