BLZPACK User's Guide*

Osni Marques†

November 21, 2015

Abstract

This document describes how to use BLZPACK (Block Lanczos Package), an implementation of the block Lanczos algorithm intended for the solution of eigenvalue problems involving real, sparse, symmetric matrices. BLZPACK works in an interactive way, so as the matrices of the target problem do not need to be passed as arguments for the interface subprogram. The user has therefore freedom to exploit the characteristics of a particular application, including different storage formats. This document describes the list of arguments of the BLZPACK interface, error and warning messages, and gives general information and examples of use.

*This work was partly supported by the Director, Office of Computational and Technology Research, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy under contract No. DE-AC03-76SF00098.

†Scalable Solvers Group, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, USA. E-mail: oamarques@lbl.gov.

Contents

1 Overview

BLZPACK (for Block Lanczos Package) is a standard Fortran implementation of the block Lanczos algorithm, which employs a combination of a modified partial reorthogonalization and a selective orthogonalization strategies to preserve the orthogonality of the basis of vectors generated by the Lanczos algorithm. The package is intended for the computation of pairs of eigenvalues λ and eigenvectors x of the standard eigenvalue problem

$$Ax = \lambda x$$

or the generalized eigenvalue problem

$$Ax = \lambda Bx$$

where A and B are real sparse symmetric matrices. In the latter case, it is assumed that there exists a positive definite linear combination of A and B to guarantee that all eigenvalues are real. Details of the algorithm can be found in [2, 3, 4, 5, 6, 7] and the current implementation supersedes the information provided in the user's guide appended to [5].

BLZPACK can be used in either sequential or parallel mode, by means of either MPI or PVM communication interfaces[1]. Let n be the dimension of A and B, and npe the number of processes, then the numerical operations can be distributed so as each process computes with n_i rows of the eigenvalue problem and $n = \sum_i^{npe} n_i$. Figure 1 illustrates the distribution of the eigenvalue problem on 5 processes.

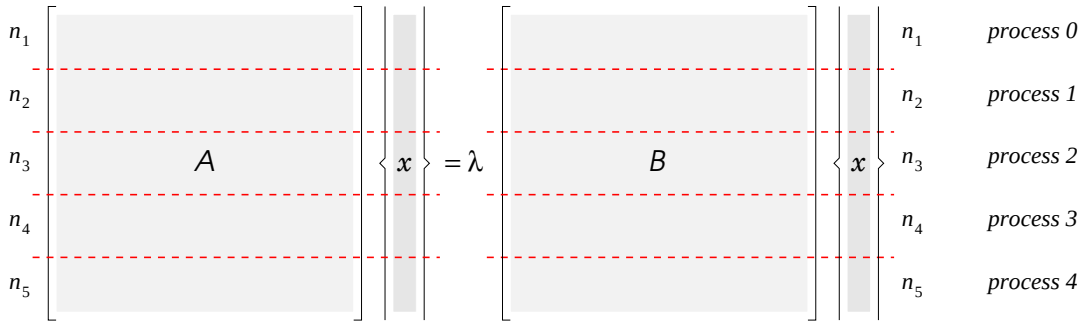


Figure 1: Distribution of the eigenvalue problem, *npe=5*.

The use of BLZPACK is explained in Section 2 of this document, error and warning messages are listed in Section 3, general information is given in Section 4, the algorithm implemented in BLZPACK is briefly described in Section 5, guidelines for the preparation of a driver for BLZPACK are given in Section 6, and examples of use are given in Section 7.

A Unix script file included in the BLZPACK distribution allows the automatic generation of a (sequential or parallel) library for BLZPACK on many existing platforms. The distribution contains also further documentation and a collection of driver models.

[1]The PVM interface is no longer supported and is given only as reference.

2 How to use the package

The user interface for BLZPACK is the subroutine xBLZDRV, which defines control parameters, specifies data to be supplied by the user at execution time, and computes eigenvalues and eigenvectors. The algorithm implemented in BLZPACK requires calculations involving the matrices A and B, and sets of vectors until convergence for the required eigenvalues and eigenvectors is reached. However, each time such calculations have to be performed the control is returned to the user (reverse communication strategy). Therefore, A and B do not have to be passed as arguments to the interface[2].

2.1 Argument list

The single precision version,

```
CALL SBLZDRV (ISTOR,RSTOR,SIGMA,NNEIG,U,V,LFLAG,NVOPU,EIG,X)
```

The double precision version,

```
CALL DBLZDRV (ISTOR,RSTOR,SIGMA,NNEIG,U,V,LFLAG,NVOPU,EIG,X)
```

ISTOR is an INTEGER array of dimension 20+LISTOR (defined below) holding variables that must be preserved between calls to D/SSBLZDRV. The first 15 entries of ISTOR must be set by the user as follows:

ISTOR(1)=NI, where NI is the number of active rows of U, V and X on the current process, such that $\mathrm{N} = \sum_{i=1}^{\mathrm{NPE}} \mathrm{NI}_i$, where N is the dimension of the matrix A (and B), NPE is the number of processes, and $\mathrm{NI}_i > 0$. See Section 2.2.1 for more information.

ISTOR(2)=LNI, where LNI is the first dimension of the arrays U, V and X. LNI must be equal to or greater than NI.

ISTOR(3)=NREIG, where NREIG is the number of required eigenvalues and eigenvectors.

ISTOR(4)=LEIG, where LEIG is the first dimension of the array EIG and the second dimension of the array X. LEIG should be greater than NREIG because the number of *converged* eigenpairs may be greater than the number of *required* eigenpairs. It is recommended that LEIG≥min(NREIG+10,NREIG×2).

[2]The name of the interfaces have changed with respect to earlier versions of the package; the list of arguments remains the same.

ISTOR(5)=NVBSET, where NVBSET is the number of vectors in a block. If NVBSET is set to 0, BLZPACK will set the number of vectors in a block based on the dimension of RSTOR. See Section 2.2.2 for more information.

ISTOR(6)=NSTEPS, where NSTEPS is the maximum number of steps to be performed per run. If NSTEPS is set to 0, BLZPACK will set the maximum number of steps based on the dimension of RSTOR. See Section 2.2.3 for more information.

ISTOR(7)=NSTART, where NSTART is the number of starting vectors given as input in the array V. If NSTART is set to 0, BLZPACK will generate starting vectors with random entries. See Section 2.2.4 for more information.

ISTOR(8)=NGEIG, where NGEIG is the number of eigenpairs given as input in the arrays EIG and X. See Section 2.2.5 for more information.

ISTOR(9)=GNRZD, where GNRZD defines the problem to be solved: 0 for a standard eigenvalue problem, 1 or 2 for a generalized eigenvalue problem. See Section 2.2.6 for more information.

ISTOR(10)=SLICE, where SLICE sets the spectrum slicing strategy: 0 sets the slicing off, 1 sets the slicing on. SLICE is ignored if GNRZD=0. See Section 2.2.7 for more information.

ISTOR(11)=PURIFY, where PURIFY sets the purification of eigenvectors: 0 sets the purification off, 1 sets the purification on. PURIFY is ignored if GNRZD=0. See Section 2.2.8 for more information.

ISTOR(12)=LPRNT, where LPRNT sets the level of information to be printed:

0, nothing is printed;
1, prints header, errors and warnings;
2, the same of 1 plus eigenvalues;
3, the same of 2 plus algorithm statistics;
4, the same of 3 plus a run-by-run history;
5, the same of 4 plus eigenvectors (sequential mode only);
6, the same of 5 plus a step-by-step history.

ISTOR(13)=LFILE, where LFILE defines the file unit where the level of information specified by LPRNT will be printed. If set to 0 or 6, the output is usually printed on the screen. If set to an integer k between 1 and 99, the output is usually printed in a file *fort.k* (unless the file is named with an OPEN statement in the calling program).

ISTOR(14)=LCOMM, where LCOMM must be a nonnegative integer. In the MPI version, LCOMM must be set to the *communicator* associated with the group of processes running BLZPACK. In the PVM version, LCOMM must be set to an integer k between 0 and 9, in which case the PVM group name associated to BLZPACK has to be defined as *blzpack.pvm.k*. LCOMM is ignored in the sequential mode. See Section 7 for more information.

ISTOR(15)=LISTOR, where 20+LISTOR is the dimension of the array ISTOR.

$$\text{LISTOR} \geq 150 + k_1 \times 12,$$

where

$$k_1 = \begin{cases} \text{NVBSET}\times\text{NSTEPS if NVBSET}\times\text{NSTEPS}>0, \\ \min(\text{N},180) \text{ otherwise (default).} \end{cases}$$

If LISTOR=0 a workspace query is assumed: BLZPACK computes the minimum LISTOR needed for execution, overwrites ISTOR(15) with this value and exits with LFLAG=0. See Section 2.2.9 for more information.

RSTOR is a REAL (DOUBLE PRECISION in the D version) array of dimension 10+LRSTOR (defined below) holding variables that must be preserved between calls to xBLZDRV. The first 4 entries of RSTOR must be set by the user as follows:

RSTOR(1)=EIGL, where EIGL is the lower (left) limit for the eigenvalue interval. EIGL is ignored if GNRZD=0. See Section 2.2.10 for more information.

RSTOR(2)=EIGR, where EIGR is the upper (right) limit for the eigenvalue interval. EIGR is ignored if GNRZD=0. See Section 2.2.10 for more information.

RSTOR(3)=THRSH, where THRSH is the threshold for convergence. If THRSH is set to 0, BLZPACK will assign a value to the threshold based on the eigenvalue approximations computed at each step. See Section 2.2.11 for more information.

RSTOR(4)=LRSTOR, where 10+LRSTOR is the dimension of the array RSTOR. In *sequential mode*,

$$\text{LRSTOR} \geq \begin{cases} \text{NI}\times k_2 \times 5 + k_3 \text{ if GNRZD=0,} \\ \text{NI}\times(k_2 \times 6 + 1) + k_3 \text{ otherwise,} \end{cases}$$

and in *parallel mode*,

$$\text{LRSTOR} \geq \begin{cases} \text{NI}\times(k_2 \times 4 + k_1) + k_3 \text{ if GNRZD=0,} \\ \text{NI}\times(k_2 \times 4 + k_1 \times 2 + k_4) + k_3 \text{ otherwise,} \end{cases}$$

where

$$\begin{aligned} k_2 &= \begin{cases} \text{NVBSET if NVBSET>0,} \\ \text{3 otherwise (default),} \end{cases} \\ k_3 &= 500 + k_1 \times (13 + k_1 \times 2 + k_2 + \max(18, k_2+2)) + \\ &\quad k_2 \times k_2 \times 3 + k_4 \times 2, \\ k_4 &= \min(\text{LEIG,N}). \end{aligned}$$

If LRSTOR=0 a workspace query is assumed: BLZPACK computes the minimum LRSTOR required for execution, stores this value in RSTOR(4) and exits with LFLAG=0. See Section 2.2.9 for more information about how LRSTOR can affect the computational performance.

SIGMA (σ) is a REAL (DOUBLE PRECISION in the D version) variable which BLZPACK sets to the translation of origin to be applied to the eigenvalue spectrum. On exit with LFLAG=3, the user must compute the factorization $A - \sigma B = LDL^T$ and call xBLZDRV again. SIGMA is not used if GNRZD=0, but must be preserved between calls to xBLZDRV if GNRZD$\neq$0. See Sections 2.2.6, 2.2.7 and 2.2.10 for more information.

NNEIG is an INTEGER variable which must be set by the user to the number of eigenvalues less than σ of the matrix $A-\sigma B$. This information can be recovered from the inertia of D after the factorization $A - \sigma B = LDL^T$. NNEIG is not used if GNRZD=0. See Sections 2.2.6 and 2.2.10 for more information.

U is a REAL (DOUBLE PRECISION in the D version) two-dimensional array with dimensions (LNI,NCU) which need not be set by the user and must be preserved between calls to xBLZDRV. NCU is not required by xBLZDRV, but must be set in the calling program to 3 or NVBSET, if NVBSET>0.

V is a REAL (DOUBLE PRECISION in the D version) two-dimensional array with dimensions (LNI,NCV) which need not be set by the user and must be preserved between calls to xBLZDRV. NCV is not required by xBLZDRV, but must be set in the calling program to 3 or NVBSET, if NVBSET>0. On exit with LFLAG=4, the user must specify the starting vectors for the Lanczos algorithm in the leading NI×NVOPU entries of V and call xBLZDRV again.

LFLAG is an INTEGER variable used to control execution as follows:

LFLAG<0, an error has been detected, the computation is interrupted and the control is returned to the user. The corresponding error code is stored in ISTOR(16).

LFLAG=0, must be used on the first call to xBLZDRV, after the entries of ISTOR and RSTOR have been set by the user. The package keeps an internal lock which is open only with LFLAG=0. Upon completion LFLAG is reset to 0. A warning code, if any, is stored in ISTOR(17).

LFLAG=1, given the leading NI×NVOPU entries of U, the user must compute the corresponding piece of AU if GNRZD=0 or $(A-\sigma B)^{-1}$U if GNRZD>0, assign the result to the leading NI×NVOPU entries of V, and call xBLZDRV again.

LFLAG=2, given the leading NI×NVOPU entries of U, the user must compute the corresponding piece of BU, assign the result to the leading NI×NVOPU entries of V, and call xBLZDRV again.

LFLAG=3, given SIGMA, the user must compute $A-\sigma B = LDL^T$ in order to replace operations with an inverse matrix by solutions of systems of linear equations, recover NNEIG from the inertia of D, and call xBLZDRV again.

LFLAG=4, given NSTART>0, the user must specify the starting vectors for the Lanczos algorithm in the leading NI×NVOPU entries of V and call xBLZDRV again.

LFLAG=5, sets for finalization and computation of eigenvector approximations. See Section 2.2.12 for more information.

NVOPU is an INTEGER variable which BLZPACK sets to the number of columns of U or V to be considered in the operations involving A and B. NVOPU may vary for LFLAG=1, 2 or 4.

EIG is a REAL (DOUBLE PRECISION in the D version) two-dimensional array with dimensions (LEIG,2) which need not be set by the user. On exit with LFLAG=0, EIG holds the NTEIG eigenvalue approximations in the first NTEIG positions of the first column, and the corresponding estimated residuals in the first NTEIG positions of the second column. Therefore, all converged approximations are stored (in ascending order) in EIG, with the corresponding eigenvectors stored in X. See Section 2.2.5 for more information.

X is a REAL (DOUBLE PRECISION in the D version) two-dimensional array with dimensions (LNI,LEIG) which need not be set by the user. On exit with LFLAG=0, X holds NI components of each eigenvector approximation (according to the ordering of EIG) in the leading NI×NTEIG entries. See Sections 2.2.1 and 2.2.5 for more information.

2.2 Argument list: details

2.2.1 Active rows of U, V and X

In sequential mode, ISTOR(1)=NI=N. In parallel mode, each process contains a set of rows of the eigenvalue problem, as illustrated in Figure 1. Then, the partitioning of the arrays U, V and X must be as follows:

process	ISTOR(1)	row indices of U, V and X	leading dimension of U, V and X
0	NI_0	1 to NI_0	LNI_0 ($\geq NI_0$)
1	NI_1	NI_0+1 to NI_0+NI_1	LNI_1 ($\geq NI_1$)
$\vdots$	$\vdots$	$\vdots$	$\vdots$
NPE-1	NI_{NPE-1}	$(\sum_{i=1}^{NPE-2} NI_i)$+1 to N	LNI_{NPE-1} ($\geq NI_{NPE-1}$)

2.2.2 Number of vectors in a block

The block Lanczos algorithm iterates with a matrix R of dimension N×NVB, $N = \sum_{i=1}^{NPE} NI_i$, NVB $\ll$ N, where NVB is the *block size*. When NVB=1 a formulation similar to the single-vector Lanczos algorithm is obtained. An implementation by blocks provides for better data management on some computer architectures and may yield better convergence rates when there are multiple eigenvalues. However, the definition of a fixed value for NVB that would lead to similar performances for all applications is not feasible. It is often suggested that NVB should be set as follows:

- $\leq$ NREIG (the number of required eigenpairs);
- 6, for structural engineering problems with rigid body motions (there are similar situations in chemistry applications, for example);
- k, where k is the largest multiplicity of any sought eigenvalue (which is normally unknown in advance).
- the cost of applying A and B to NVB vectors should be less than or equal to 1.5 the cost of applying A and B to one vector.

If NVBSET>0, BLZPACK uses NVB=NVBSET, otherwise it assigns to NVB a value based on the dimension of RSTOR (3 being the default). The user should then let BLZPACK set NVB if no information about the problem being solved is available.

2.2.3 Maximum number of steps per run

The number of steps required for convergence of NREIG eigenpairs depends on the starting vectors and on the eigenvalue distribution of the problem being solved. A value for NSTEPS between NREIG×3 and NREIG×6 should be a good guess. If NSTEPS=0, BLZPACK assigns a value to the maximum number of steps based on the dimension of RSTOR (the default leads to the computation of a basis with 180 vectors). The user should then let BLZPACK set the maximum number of steps if no information about the problem being solved is available.

It can happen that the maximum number of steps specified by the user or set by BLZPACK is not enough for the convergence of the required eigenpairs. In this case, a restart will be automatically performed and BLZPACK will take as initial vectors a linear combination of some vectors that did not converge to eigenvectors in the previous run, or the entries given in V, if NSTART>0. More sophisticated restart strategies can be implemented by the user, externally to BLZPACK.

Each time the maximum number of steps is reached and a restart is automatically performed a "new run" is initiated. This means that the solution of the eigenvalue problem by BLZPACK may require several runs.

2.2.4 Starting vectors given as input

The user can specify the starting vectors for the Lanczos algorithm by setting NSTART>0. Then, on exit with LFLAG=4, the user must provide linearly independent starting vectors for the algorithm in the NI×NVOPU leading entries of V (see Section 2.2.1) and call xBLZDRV again. This option can be also used, for instance, when estimates for a few eigenvectors, or linear combinations of some eigenvectors, are available. However, if the estimates match the eigenvectors in many digits, an ill-conditioned subspace may be computed. See Section 4 for more information on ill-conditioned subspaces.

2.2.5 Eigenpairs given as input

If a set of approximate eigenpairs $(\hat{\lambda}_i, \hat{x}_i)$, $i = 1, 2, \ldots$ NGEIG, is available when calling xBLZDRV with LFLAG=0, $(\hat{\lambda}_i, \hat{x}_i)$ may be given as input in the arrays EIG and X. Then, BLZPACK will search for eigenvectors that are orthogonal to the eigenvectors in X. This option may be used in case of restart or to extend the range of eigenvalues previously computed (it is a form of "locking" known eigenvalues and eigenvectors).

The eigenvalues $\hat{\lambda}_i$ and associated residual norms $\|A\hat{x}_i - \hat{\lambda}_i\hat{x}_i\|$ ($\|A\hat{x}_i - \hat{\lambda}_i B\hat{x}_i\|$ in the generalized case) must be stored in the columns of EIG as:

and the corresponding eigenvectors in the columns of X as $\hat{x}_1, \hat{x}_2, \cdots \hat{x}_{\text{NCEIG}}$ (see Section 2.2.1).

On exit with LFLAG=0, the number of computed eigenvalues, NTEIG, includes NGEIG. The

row	1st column	2nd column
1	$\hat{\lambda}_1$	$\|A\hat{x}_1 - \hat{\lambda}_1\hat{x}_1\|$
2	$\hat{\lambda}_2$	$\|A\hat{x}_2 - \hat{\lambda}_2\hat{x}_2\|$
$\vdots$	$\vdots$	$\vdots$
NCEIG	$\hat{\lambda}_{\text{NCEIG}}$	$\|A\hat{x}_{\text{NCEIG}} - \hat{\lambda}_{\text{NCEIG}}\hat{x}_{\text{NCEIG}}\|$

required eigenvalues and associated residuals are therefore stored from row NGEIG+1 to row NTEIG in EIG, and the eigenvectors from column NGEIG+1 to column NTEIG in X. It should be emphasized that if the residuals $\|A\hat{x}_i - \hat{\lambda}_i\hat{x}_i\|$ ($\|A\hat{x}_i - \hat{\lambda}_i B\hat{x}_i\|$ in the generalized case) are not small relatively to $\|A\|$, in other words, the approximations for the NGEIG eigenpairs are not accurate, numerical problems may occur. The user should be then careful when specifying NGEIG and the corresponding entries in the arrays EIG and X.

Section 7 gives an example in which the largest eigenvalue and corresponding eigenvector of a problem $Ax = \lambda x$ are used to "deflate" the spectrum and extend the range of the eigenvalues computed.

2.2.6 Problem types

BLZPACK can be used to solve the standard eigenvalue problem $Ax = \lambda x$ (GNRZD=0), the generalized eigenvalue problem $Ax = \lambda Bx$ (GNRZD=1), or the *buckling problem* $Kx = \lambda K_\delta x$ (GNRZD=2). The latter problem arises in structural engineering: K is a symmetric positive semidefinite stiffness matrix and K_δ is a symmetric geometric stiffness matrix. In this case, $A = K$ and $B = K_\delta$, but the eigenvalue problem is handled in a special way (see details below).

In the standard problem case, BLZPACK returns the eigenvalues of largest magnitude. The problem $Ax = \lambda x$ can be transformed to $A^{-1}x = \frac{1}{\lambda}x$ and in this case BLZPACK returns the largest $\frac{1}{\lambda}$. See examples in Section 7.

In the generalized problem cases, BLZPACK returns the eigenvalues of smallest magnitude or as specified by the computational interval (see Section 2.2.10). The problem $Ax = \lambda Bx$ is internally transformed by BLZPACK to $B(A - \sigma B)^{-1}Bx = \theta Bx$, where $\theta = \frac{1}{\lambda - \sigma}$ and σ is an appropriate translation of origin for the eigenvalue spectrum. See Figure 3.a. The buckling problem $Kx = \lambda K_\delta x$ is internally transformed by BLZPACK to $K(K - \sigma K_\delta)^{-1}Kx = \theta Kx$, where $\theta = \frac{\lambda}{\lambda - \sigma}$ and $\sigma \neq 0$ (if $\sigma = 0$ all eigenvalues are mapped into 1). See Figure 3.b.

An explicit inversion of A or $A - \sigma B$ does not need to be performed, since systems of linear equations can be solved instead. In other words, one can factor $A - \sigma B$ as LDL^T (or $K - \sigma K_\delta$ as LDL^T), where L is a lower unit triangular matrix and D is a direct sum of 1×1 and 2×2 pivot blocks. It turns out that the inertias of $A - \sigma B$ and D are the same. The inertia of a matrix is given by a triple of nonnegative integers: the number of eigenvalues less, equal, and greater than zero. Therefore, the number of eigenvalues less than σ (the argument NNEIG of xBLZDRV) can be recovered from D.

Table 1: Problem types.

problem	GNRZD	computations	output
$Ax = \lambda x$	0	LFLAG=1: V=AU	largest λ, $x^T x = 1$
$A^{-1}x = \frac{1}{\lambda}x$	0	LFLAG=1: V=A^{-1}U	largest $\frac{1}{\lambda}$, $x^T x = 1$
$Ax = \lambda Bx$	1	LFLAG=1: (LDL^T)V=U LFLAG=2: V=BU LFLAG=3: $A - \sigma B = LDL^T$	smallest λ or in interval $x^T Bx = 1$
$Kx = \lambda K_\delta x$	2	LFLAG=1: (LDL^T)V=U LFLAG=2: V=KU LFLAG=3: $K - \sigma K_\delta = LDL^T$	smallest λ or in interval $x^T Kx = 1$

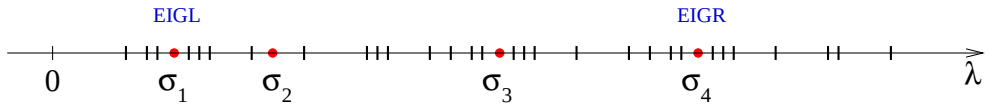


Figure 2: Spectrum slicing strategy, subintervals $[\sigma_1, \sigma_2]$, $[\sigma_2, \sigma_3]$ and $[\sigma_3, \sigma_4]$.

The problem types that can be solved by BLZPACK are summarized in Table 1, as well as the corresponding output and the computations that the user has to perform for different values of LFLAG.

2.2.7 Spectrum slicing

A spectrum slicing or dynamic shift is useful when many eigenpairs are sought or the eigenvalue distribution is clustered. This option may require factorizations of matrices $A - \sigma B$ for a set of scalars σ, as shown in Figure 2. The spectrum slicing provides for reliable checks of the computed eigenvalues. The eigenvalue problem is solved in subintervals, based on the information that the Lanczos algorithm obtains for the eigenvalue distribution.

The strategy implemented for the dynamic shift may fail in the case of pathological eigenvalue distributions, as for example big clusters of eigenvalues. However, if LPRNT is set accordingly (level 4), BLZPACK returns information on the subintervals examined which can help the user to decide how to proceed. Possible alternatives would be to restart the algorithm by modifying the computational interval or the block size, for instance.

2.2.8 Purification of eigenvectors

When the matrix B is positive semidefinite, the generalized problem $Ax = \lambda Bx$ has eigenvalues equal to $-\infty$ or $+\infty$. Components in the null space of B may then appear in the computed

eigenvectors. However, the computed eigenvectors can be purged from those components by an inexpensive procedure which can be activated by PURIFY. The user should thus set PURIFY on in case of doubt about the properties of B (it should be noted, however, that all eigenvalues are guaranteed to be real only if a positive definite linear combination of A and B exists).

2.2.9 Workspace and temporary files

The workspace available in the arrays ISTOR and RSTOR (LISTOR and LRSTOR, respectively) must be defined in ISTOR(15) and RSTOR(4), as indicated in Section 2.1. If LFLAG=0 and ISTOR(15)=0 or RSTOR(4)=0 a workspace query is assumed: BLZPACK calculates the minimum sizes required for the workspaces, overwrites ISTOR(15) and RSTOR(4) with those values, and exits with LFLAG=0. In this case, the input data is also checked for consistency.

In sequential mode, temporary files may be created depending on the workspace available in RSTOR (the value set to LRSTOR), potentially affecting the computational performance of BLZPACK. These files are labeled *blzpack.u1.BQ*, *blzpack.u2.BX* and *blzpack.u3.Q*, where *u1*, *u2* and *u3* indicate the corresponding Fortran file units. *No file will be created when the workspace for RSTOR is defined as if in parallel mode.*

2.2.10 Computational interval

The computational interval [EIGL,EIGR] for BLZPACK may be specified as follows:

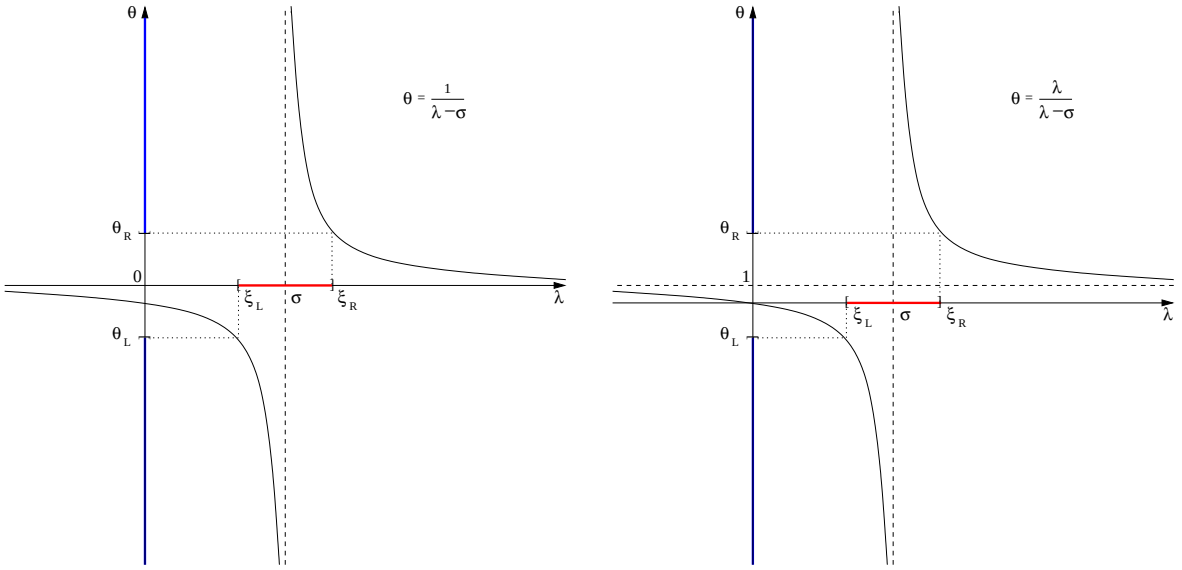
EIGL>EIGR, run from EIGL to EIGR (EIGL is the upper bound), BLZPACK seeks for NREIG eigenpairs to the left of EIGL. The computation finishes if more than NREIG eigenpairs exist in the interval or if an eigenvalue less than EIGR is found.

EIGL<EIGR, run from EIGL to EIGR (EIGL is the lower bound), BLZPACK seeks for NREIG eigenpairs to the right of EIGL. The computation finishes if more than NREIG eigenpairs exist in the interval or if an eigenvalue greater than EIGR is found.

EIGL=EIGR, run around EIGL/EIGR, BLZPACK seeks for the NREIG eigenpairs closest to EIGL/EIGR.

The computational interval [EIGL,EIGR] is taken into account only for generalized problems and in such cases the eigenvalue spectrum is transformed internally by BLZPACK as shown in Figure 3. In the buckling problem case, BLZPACK will change the interval limits slightly whenever EIGL=0 or EIGR=0. For an interval defined by lower and upper limits ξ_L and ξ_R in the λ spectrum, BLZPACK will seek for eigenvalues θ such that $\theta \leq \theta_L$ or $\theta \geq \theta_R$ (note that for a negative value of σ, the function depicted in Figure 3.b will be *inverted*). The original spectrum is then recovered at the end of the run.

In order to obtain reliable results when a interval is specified, the user should set SLICE on. BLZPACK will then apply SIGMA to at least both ends of the interval. The exact number



a) $Ax = \lambda Bx$, $\lambda = \sigma + \dfrac{1}{\theta}$. b) $Kx = \lambda K_\delta x$, $\lambda = \dfrac{\sigma\theta}{\theta - 1}$.

Figure 3: The transformation of the spectrum.

of eigenvalues lying in the interval is then given by the difference of the two corresponding NNEIG's. When EIGL=EIGR, BLZPACK will move SIGMA outwards the starting point and perform checks using pairs of SIGMA's and NNEIG's as well. See Figure 2.

The user should avoid defining the limits of the computational interval too close to known eigenvalues, providing some information on the eigenvalue distribution is available. When σ is applied too close to an eigenvalue the operator $A - \sigma B$ may result quasi-singular, leading to numerical difficulties if the quasi-singularity is not detected during the factorization $A - \sigma B = LDL^T$. This rule might be relaxed, however, when big clusters of eigenvalues are detected.

2.2.11 Threshold for convergence

An approximate eigenpair $(\hat{\lambda}_i, \hat{x}_i)$ must satisfy $\|A\hat{x}_i - \hat{\lambda}_i\hat{x}_i\| \leq |\lambda_i| tol$ (or $\|A\hat{x}_i - \hat{\lambda}_i B\hat{x}_i\| \leq |\lambda_i| tol$ in the generalized case) to be labeled converged. The threshold tol is set to THRSH if THRSH>0, otherwise $tol = \sqrt{\varepsilon}$, where ε is the machine precision ($\approx 2.2204 \times 10^{-16}$ in double precision arithmetic).

The residuals are estimated at a very low cost by BLZPACK: λ_i is estimated by means of the eigenvalue distribution computed by BLZPACK and $\hat{x}_i$ does not need to be explicitly computed. A cutoff value is used for the residuals in order to avoid underflow and they are stored in the second column of EIG for each converged eigenpair.

2.2.12 Forcing finalization

The eigenvalue approximations $\hat{\lambda}_i$ and the corresponding estimated residuals $\|A\hat{x}_i - \hat{\lambda}_i\hat{x}_i\|$ (or $\|A\hat{x}_i - \hat{\lambda}_i B\hat{x}_i\|$ in the generalized case) computed by BLZPACK are stored in a two-dimensional array with dimensions (JT,2), which starts at the address RSTOR(IRITZ), as:

row	1st column	2nd column
1	$\hat{\lambda}_1$	$\|A\hat{x}_1 - \hat{\lambda}_1\hat{x}_1\|$
2	$\hat{\lambda}_2$	$\|A\hat{x}_2 - \hat{\lambda}_2\hat{x}_2\|$
$\vdots$	$\vdots$	$\vdots$
JT	$\hat{\lambda}_{\mathrm{JT}}$	$\|A\hat{x}_{\mathrm{JT}} - \hat{\lambda}_{\mathrm{JT}}\hat{x}_{\mathrm{JT}}\|$

JT and IRITZ can be retrieved by the user by means of BLZPACK auxiliary function calls. Section 4 shows how one can accomplish that.

Good approximate eigenvalues are marked with a positive sign for the residual, zero residuals indicate that the corresponding eigenvalue has no meaning. The user can thus examine the convergence history of the algorithm step by step. An early finalization may be forced by setting LFLAG=5 and calling xBLZDRV again. Then, BLZPACK will compute the eigenvector approximations associated with the residuals with positive sign and exit. The user may change the signs of the residuals to induce the computational of specific vectors.

3 Error and warning messages

Upon completion, the value of ISTOR(16) corresponds to a code used for error messages, as indicated in Table 2, while the value of ISTOR(17) corresponds to a code used for warning messages, as indicated in Table 3.

For more than one error or warning message, each bit of the value indicates whether a message was set on. For example, codes 4 and 6 would be represented by the sum $2^4 + 2^6 = 80$.

4 General information

Invariant and ill-conditioned subspaces. The block Lanczos algorithm uses an iteration matrix whose norm may drop to a value close to zero after some steps, which indicates that an invariant or ill-conditioned subspace has been computed. In the first case, all solutions computed by the algorithm are good approximations to solutions of the original problem. This situation is reached when NVB$\times$JL is equal to N, for example. On the other hand, the iteration matrix may result rank deficient if SIGMA is applied too close to an eigenvalue, the starting vectors for the algorithm are linearly dependent or, for instance, after 3 steps of the algorithm

Table 2: Table of errors.

code	meaning	diagnostic
0	normal execution	
1	illegal data, LFLAG	LFLAG<0 or LFLAG>5
2	illegal data, dimension of (U), (V), (X)	ISTOR(1)≤0
3	illegal data, leading dimension of (U), (V), (X)	ISTOR(2)≤0 or ISTOR(2)<ISTOR(1)
4	illegal data, leading dimension of (EIG)	ISTOR(3)≤0
5	illegal data, number of required eigenvalues	ISTOR(4)≤0 or ISTOR(4)<ISTOR(3)
6	illegal data, Lanczos algorithm block size	ISTOR(5)<0 or ISTOR(5)>N
7	illegal data, maximum number of steps	ISTOR(6)<0 or ISTOR(6)>N
8	illegal data, number of starting vectors	ISTOR(7)<0 or ISTOR(7)>N
9	illegal data, number of eigenpairs provided	ISTOR(8)<0 or ISTOR(8)>N
10	illegal data, problem type flag	ISTOR(9)<0 or ISTOR(9)>2
11	illegal data, spectrum slicing flag	ISTOR(10)<0 or ISTOR(10)>1
12	illegal data, eigenvectors purification flag	ISTOR(11)<0 or ISTOR(11)>1
13	illegal data, level of output	ISTOR(12)<0
14	illegal data, output file unit	ISTOR(13)<0
15	illegal data, LCOMM (MPI or PVM)	ISTOR(14)<0
16	illegal data, dimension of ISTOR	ISTOR(15)<0
17	illegal data, convergence threshold	RSTOR(3)<0
18	illegal data, dimension of RSTOR	RSTOR(4)<0
19	illegal data on at least one PE	
20	ISTOR(3:14) must be equal on all PEs	
21	RSTOR(1:3) must be equal on all PEs	
22	not enough space in ISTOR to start eigensolution	ISTOR(14) is too small
23	not enough space in RSTOR to start eigensolution	RSTOR(4) is too small
24	illegal data, number of negative eigenvalues	NNEIG<0
25	illegal data, entries of V	$\Sigma v_{ij} = 0$ or Σv_{ij} =NaN (IEEE)
26	illegal data, entries of X	$\Sigma x_{ij} = 0$ or Σx_{ij} =NaN (IEEE)
27	failure in computational subinterval	big cluster of eigenvalues
28	file I/O error, blzpack.*u1*.BQ	
29	file I/O error, blzpack.*u2*.BX	
30	file I/O error, blzpack.*u3*.Q	
31	file I/O error, blzpack.*u4*.X	
32	parallel interface error	

Table 3: Table of warnings.

code	meaning	diagnostic
0	normal execution	
1	no eigenvalue in computational interval	no eigenvalue in [EIGL,EIGR]
2	SIGMA is too close to an eigenvalue	matrix $A - \sigma B$ was assumed singular
3	invariant or ill-conditioned starting block	block of vectors is zero or rank deficient
4	invariant or ill-conditioned subspace	block of vectors is zero or rank deficient
5	reduced eigenproblem can not be solved	
6	three runs without any new information	no converged eigenvalue after three runs
7	maximum number of runs has been reached	
8	algorithm unable to deal with subinterval	big cluster of eigenvalues
9	not enough space to continue eigensolution	LEIG not big enough
10	execution interrupted due to a big cluster	
11	no eigenpair has been computed	NTEIG=0
12	no eigenpair has been checked	SLICE=1 and only one σ applied
13	less than NREIG eigenpairs have been accepted	NTEIG<NREIG
14	less than NREIG eigenpairs have been checked	SLICE=1 and not enough σ's applied
15	matrix B seems to be indefinite	
16	orthogonality check not performed	
17	LFLAG=5 (early termination)	

with NVB=2 for a problem of dimension 5. In addition, if the matrix B in the generalized eigenvalue problem is diagonal with k non null entries, $k < n$, k is the maximum size of the basis of vectors that the Lanczos algorithm can generate. However, it may happen that the number of converged eigenpairs is less than the number of required eigenpairs when an invariant or ill-conditioned subspace is found. Then, one possible remedy is to run the algorithm again (setting LFLAG=0) using a different block size, computational interval or starting vectors.

Converged eigenpairs. All converged eigenpairs are stored in the arrays EIG and X in ascending order. It is recommended that LEIG≥min(NREIG+10,NREIG×2) because BLZPACK may compute more than NREIG eigenpairs. If there is not enough space for the eigenpairs, a warning message is issued. On the other hand, only NREIG eigenpairs or those in the computational interval are printed if LPRNT is set accordingly. In the parallel mode, the eigenvectors are not printed, which means that LPRNT=5 has not effect in this case.

The output. The level of information provided by BLZPACK is set by LPRNT, as described in Section 2.1. The higher the value of LPRNT (up to 6), the more the information. In particular, the estimated residuals $\|A\hat{x}_i - \hat{\lambda}_i\hat{x}_i\|$ (or $\|A\hat{x}_i - \lambda_i B\hat{x}_i\|$ in the generalized case) printed with positive signs correspond to converged solutions. Additional information includes the number of runs, origin translations, converged solutions, partial reorthogonalizations and selective orthogonalizations, matrix vector operations performed with A and B, CPU time (in seconds) spent in different parts of the algorithm, spectral slicing parameters, and the eigenvalue approximations computed at each step.

Retrieving internal variables. All relevant internal variables used by BLZPACK are stored in the arrays ISTOR and RSTOR. Such variables are listed and documented in the file *blz_address.h* in the directory *blzpack/src* of the BLZPACK distribution, and can be retrieved by means of the BLZPACK auxiliary functions `IRVSTOR` and `DRVSTOR` or `SRVSTOR`,

```
integer_variable = IRVSTOR(ISTOR,string1)
double_precision_variable = DRVSTOR(RSTOR,string2)
single_precision_variable = SRVSTOR(RSTOR,string2)
```

where "string1" and "string2" are strings defining variable names (listed in *addresses.h*). As an example, the following code fragment prints the eigenvalue approximations and corresponding residual norms:

```
      CALL BLZPACK (ISTOR,RSTOR,SIGMA,NNEIG,U,V,LFLAG,NVOPU,EIG,X)
C
C     code fragment for printing the eigenvalue approximations
C
C     JL    : current step
C     JT    : dimension of the projection
C     NRITZ : number of eigenvalue approximations
C     IRITZ : address of the array storing the eigenvalues
C     INORM : address of the array storing the estimated residual norms
C
      JL = IRVSTOR(ISTOR,'JL')
      JT = IRVSTOR(ISTOR,'JT')
      NRITZ = IRVSTOR(ISTOR,'NRITZ')
      IRITZ = IRVSTOR(ISTOR,'IRITZ')
      IRNORM = IRITZ + JT
C
      IF ( NRITZ .GT. 0 ) WRITE (*,'(A,I4,/,(I4,1P,2E14.4))')
     &                    'Eigenvalues and residuals at step:',JL,
     &                    (I+1,RSTOR(IRITZ+I),RSTOR(IRNORM+I),
     &                    I=0,NRITZ-1)
```

BLAS kernels. The following BLAS kernels (or their single precision versions) are called by BLZPACK: IDAMAX, DASUM, DAXPY, DCOPY, DDOT, DGEMM, DGEMV, DGER, DSCAL, DSWAP and DTRMM. These kernels are required when linking BLZPACK to a calling program. A file containing the BLAS is provided in the directory *blzpack/etc* of the BLZPACK distribution, in the case that there is no BLAS library to be linked to.

Intrinsic functions. BLZPACK calls intrinsic functions to generate random numbers, to perform bit tests and to determine the CPU time used. These functions are stored in the directory *blzpack/sys* of the BLZPACK distribution. A Unix script file provided with the package generates a library on various existing computers. If the script does not work properly on a particular computer, the user should perform modifications in the script accordingly.

LAPACK subroutines. The following LAPACK subroutines (or their single precision versions) are used upon request: DSBTRD and DSTEGR. They are used in the solution of the reduced eigenvalue problems generated by the Lanczos algorithm and can be selected when the BLZPACK library is created. See the README file in the directory *blzpack/sys* of the BLZPACK distribution and also [1].

5 Method

The algorithm implemented in BLZPACK is based on the block Lanczos algorithm, in combination with a selective orthogonalization and a modified partial reorthogonalization strategies to preserve the orthogonality of the basis of vectors generated by the algorithm. The computational scheme is summarized in Table 4 for both the standard eigenvalue problem (s.e.p.) $Ax = \lambda x$ and the generalized eigenvalue problem (g.e.p.) $Ax = \lambda Bx$.

In practical cases, the operations involving A_σ^{-1} in step (2.01) are replaced by solutions of systems of linear equations. The basis of Lanczos vectors is defined as $\mathcal{Q}_j = \left[\begin{array}{cccc} Q_1 & Q_2 & ... & Q_j \end{array} \right]$, and T_j is a block tridiagonal matrix with A_j in the main diagonal and B_j and B_j^T in the lower and upper diagonals, respectively. If NSTEPS is not enough for the convergence of the required eigenpairs, a restart is usually performed. Then, the modified partial reorthogonalization in step (2.06) is complemented with an orthogonalization of Q_j and Q_{j+1} against specific vectors $\hat{x}_k$ previously computed (selective orthogonalization). In step (2.10), $s_j^{(k)}$ contains the last NVB components of the eigenvector s_k, and usually $tol = |\hat{\lambda}_k|\sqrt{\varepsilon}$, where $\varepsilon \approx 2.2204 \times 10^{-16}$ (in double precision arithmetic).

6 Writing a driver for xBLZDRV

The flowchart depicted in Figure 4 gives guidelines on how to prepare a driver for BLZPACK. One should proceed as follows with the different values of LFLAG:

LFLAG=1. Use $op(A) = A$ or $op(A) = A^{-1}$ for $Ax = \lambda x$. Use $op(A) = A_\sigma^{-1}$ for $Ax = \lambda Bx$, where $A_\sigma = A - \sigma B$. In practical cases, the inversion is replaced by the solution of a system of linear equations, i.e., one should solve AV=U or A_σV=U.

LFLAG=2. Use $op(B) = B$. This flag should be taken into account only if the problem has been defined as generalized.

LFLAG=3. Given σ, compute $A_\sigma = A - \sigma B$ and prepare for calculations with LFLAG=1, i.e., factor $A_\sigma = LDL^T$ for the solution of systems of linear equations. This flag should be taken into account only if the problem has been defined as generalized.

LFLAG=4. Set the starting vectors for the Lanczos algorithm. This flag should be taken into account only if ISTOR(7)>0.

LFLAG=5. This flag forces finalization at the user's discretion. If desired, the user can select vectors to be computed (see Section 2.2.12).

The entries in U and V are meaningless when LFLAG=0 or 3.

7 Examples of use

In this section, we give six simple driver models for BLZPACK to simulate the flowchart depicted in Figure 4:

- Driver model DRVSP1. This driver model is aimed at the standard eigenvalue problem $Ax = \lambda x$ and uses the BLAS kernel DSYMM to perform matrix vector multiplications.

Table 4: The Block Lanczos technique used by BLZPACK.

1. *Initialization*:

set NVB, the number of vectors in a block
set NSTEPS, the maximum number of steps per run
set σ and compute $A_\sigma = A - \sigma B$ for the g.e.p.
set $Q_0 = 0$
set $R_0 \neq 0$ and factorize R_0: $R_0 = Q_1 \mathsf{B}_1$ $\begin{cases} Q_1^T Q_1 = I & \text{for the s.e.p.} \\ Q_1^T B Q_1 = I & \text{for the g.e.p.} \end{cases}$

2. *Lanczos steps*:

for j=1,2,...NSTEPS

2.01) compute $R_j = \begin{cases} AQ_j & \text{for the s.e.p.} \\ A_\sigma^{-1} B Q_j & \text{for the g.e.p.} \end{cases}$
2.02) $R_j \leftarrow R_j - Q_{j-1}\mathsf{B}_j^T$
2.03) $\mathsf{A}_j \leftarrow \begin{cases} Q_j^T R_j & \text{for the s.e.p.} \\ Q_j^T B R_j & \text{for the g.e.p.} \end{cases}$
2.04) $R_j \leftarrow R_j - Q_j \mathsf{A}_j$
2.05) factorize R_j: $R_j = Q_{j+1}\mathsf{B}_{j+1}$ $\begin{cases} Q_{j+1}^T Q_{j+1} = I & \text{for the s.e.p.} \\ Q_{j+1}^T B Q_{j+1} = I & \text{for the g.e.p.} \end{cases}$
2.06) if required orthogonalize Q_j and Q_{j+1} against the vectors in $\mathcal{Q}_{j-1}$
2.07) insert Q_j into $\mathcal{Q}_j$ and A_j, B_j into T_j
2.08) solve the reduced problem $T_j s_k = s_k \theta_k$, $k = 1, 2, \ldots$NVB$\times j$
2.09) $\hat{\lambda}_k = \begin{cases} \sigma & \text{for the s.e.p.} \\ \sigma + \dfrac{1}{\theta_k} & \text{for the g.e.p.} \end{cases}$
2.10) convergence test:
check the number of eigenpairs for which
$tol \geq \|\beta_{j+1} s_j^{(k)}\| \approx \begin{cases} \|A\hat{x}_k - \hat{\lambda}_k \hat{x}_k\| & \text{for the s.e.p.} \\ \|A\hat{x}_k - \hat{\lambda}_k B\hat{x}_k\| & \text{for the g.e.p.} \end{cases}$
and exit if enough of them have converged

end for

3. *Compute the eigenvector approximations (converged eigenpairs)*:

$\hat{x}_k = \mathcal{Q}_j s_k$

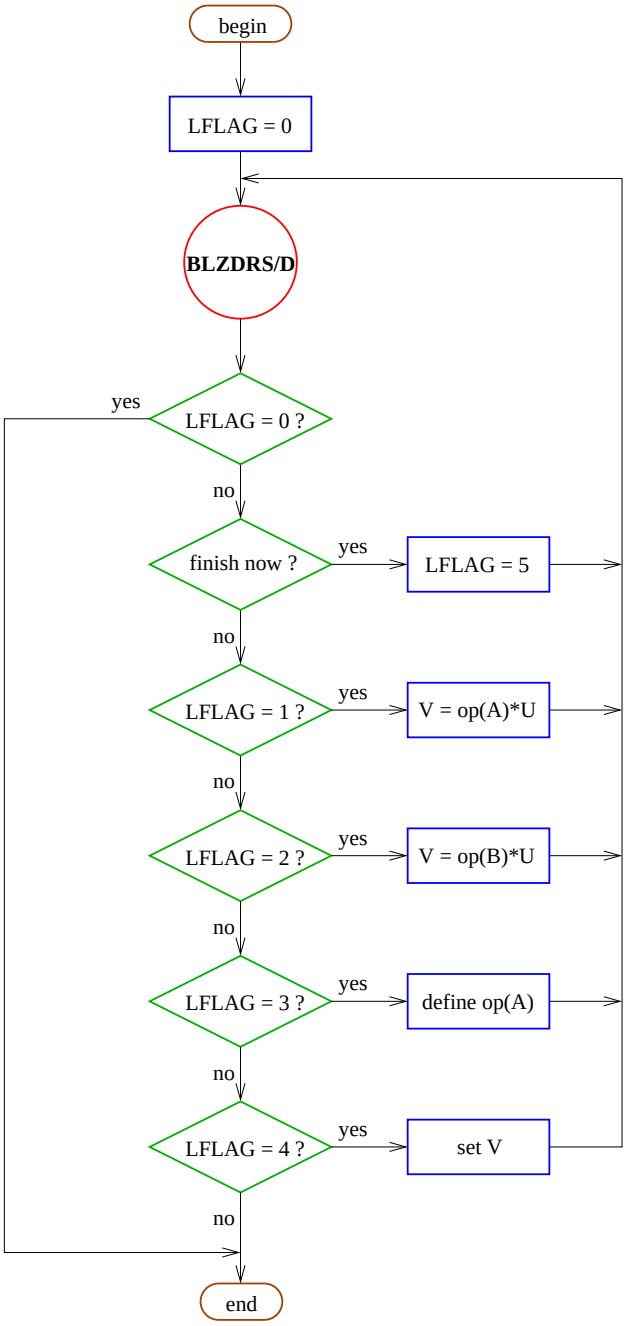


Figure 4: Flowchart for a program unit calling xBLZDRV.

- Driver model DRVSP2. This driver model is aimed at the standard eigenvalue problem $Ax = \lambda x$ and uses a sparse matrix vector multiplication. Two eigenvectors are given as input.
- Driver model DRVGP1. This driver model is aimed at the generalized eigenvalue problem $Ax = \lambda Bx$. For the sake of simplicity, B is set to the identity matrix.
- Driver model DRVGP4. This driver model is aimed at the generalized eigenvalue problem $Ax = \lambda Bx$. The eigenvalue problem is solved in buckling mode.
- Driver model DRVMPI. Parallel version of DRVSP1, MPI interface.
- Driver model DRVPVM. Parallel version of DRVSP1, PVM interface.

For the first four models we list the parameters read by the driver and also give the resulting output file. Drivers DRVGP1 and DRVGP4 use the subroutine MA47 from the Harwell Subroutine Library (*http://www.hsl.rl.ac.uk*) to perform a factorization of the matrix $A - \sigma B$ and solution of systems of equations. Alternatively, MA47 can be replaced by MA27 or any other equivalent routine. Note, however, that such routines are not included in the BLZPACK distribution.

The same symmetric matrix, of dimension 10, is used for all drivers. Since the test matrix is small, a basis size comparable to the dimension of the matrix may be needed depending on the block size used and on the number of eigenpairs required.

The upper triangle of the test matrix A is stored in coordinate format (row, column, value) in the file *A.dat*:

```
 1    1    5.000000000000000e+00
 2    2    4.000000000000000e+00
 1    3    1.500000000000000e+01
 2    3    8.000000000000000e+00
 3    3    6.400000000000000e+01
 2    4    1.200000000000000e+01
 3    4    3.000000000000000e+01
 4    4    5.000000000000000e+01
 3    5    9.000000000000000e+00
 4    5    2.200000000000000e+01
 5    5    3.600000000000000e+01
 1    6    3.000000000000000e+01
 2    6    2.000000000000000e+01
 3    6    1.420000000000000e+02
 4    6    9.000000000000000e+01
 5    6    5.000000000000000e+01
 6    6    3.510000000000000e+02
 5    7    3.000000000000000e+00
 6    7    8.000000000000000e+00
 7    7    1.500000000000000e+01
 5    8    4.000000000000000e+00
 6    8    1.100000000000000e+01
 7    8    2.200000000000000e+01
 8    8    3.600000000000000e+01
 3    9    2.100000000000000e+01
 4    9    5.400000000000000e+01
 5    9    9.200000000000000e+01
 6    9    1.340000000000000e+02
 7    9    2.900000000000000e+01
 8    9    5.000000000000000e+01
 9    9    2.940000000000000e+02
 8   10    9.000000000000000e+00
 9   10    2.600000000000000e+01
10   10    4.800000000000000e+01
```

The eigenvalues of the matrix and the corresponding eigenvectors of $Ax = \lambda x$ are:

```
x =

  Columns 1 through 5

   9.2653e-01   2.2281e-01  -2.6021e-02   6.7915e-02  -9.1419e-02
  -1.3982e-01   9.2065e-01   1.9415e-02  -1.6040e-01  -7.5761e-02
  -2.0911e-01   1.0679e-01  -3.4743e-01   7.4332e-01  -2.3846e-01
   2.6908e-01  -2.2677e-01  -7.0728e-02   1.7785e-01  -4.1721e-02
   1.0119e-02   1.7275e-01   3.1015e-01   4.0424e-01   6.9780e-01
  -4.9844e-02  -9.0482e-02   1.7780e-01  -3.8055e-01   1.2994e-01
   4.9624e-02   9.9661e-03  -6.4574e-01  -1.1179e-01   5.3548e-01
   1.3164e-02  -3.2313e-02   5.4755e-01   2.4486e-01  -5.7324e-02
  -2.2950e-02   2.6451e-02  -1.6888e-01  -6.8899e-02  -3.1405e-01
   9.9630e-03  -8.2702e-03  -1.1255e-02  -8.7865e-03   1.8664e-01

  Columns 6 through 10

   2.3804e-01   6.2671e-02   9.0144e-02  -8.5078e-02   4.7853e-02
  -3.0595e-01  -5.1185e-02  -1.7667e-02  -5.6587e-02   3.5710e-02
   1.4111e-01   3.1772e-02   1.4357e-01  -3.2572e-01   2.5913e-01
  -7.9364e-01  -2.8155e-01  -2.7939e-01  -5.6472e-02   2.1773e-01
   1.7392e-01  -1.2300e-01  -3.2156e-01   2.0047e-01   1.8590e-01
   9.1669e-02   6.0313e-02   6.8177e-02  -4.9224e-01   7.3282e-01
  -2.0424e-01   4.2250e-01   2.2156e-01   9.9358e-02   4.5182e-02
  -3.2121e-01   5.1944e-01   4.7528e-01   1.7727e-01   7.3665e-02
   1.2683e-01  -1.2242e-02  -6.6631e-02   7.3775e-01   5.4948e-01
  -1.8034e-02  -6.6778e-01   7.1093e-01   1.1051e-01   3.0040e-02


eig =

   6.8426e-04
   6.5869e-03
   2.8600e-01
   1.0713e+00
   1.4875e+00
   2.5446e+01
   4.1476e+01
   5.1580e+01
   2.3600e+02
   5.4565e+02
```

The upper triangle of the test matrix B is stored in coordinate format (row, column, value) in the file *B2.dat*:

```
 1    1    5.000000000000000e+00
 2    2    5.000000000000000e+00
 1    3    1.000000000000000e+00
 2    3    1.000000000000000e+00
 3    3    5.000000000000000e+00
 2    4    1.000000000000000e+00
 3    4    1.000000000000000e+00
 4    4    5.000000000000000e+00
 3    5    1.000000000000000e+00
 4    5    1.000000000000000e+00
 5    5    5.000000000000000e+00
 1    6    1.000000000000000e+00
 2    6    1.000000000000000e+00
 3    6    1.000000000000000e+00
 4    6    1.000000000000000e+00
 5    6    1.000000000000000e+00
 6    6    5.000000000000000e+00
 5    7    1.000000000000000e+00
 6    7    1.000000000000000e+00
 7    7    5.000000000000000e+00
 5    8    1.000000000000000e+00
 6    8    1.000000000000000e+00
 7    8    1.000000000000000e+00
 8    8    5.000000000000000e+00
 3    9    1.000000000000000e+00
 4    9    1.000000000000000e+00
 5    9    1.000000000000000e+00
 6    9    1.000000000000000e+00
 7    9    1.000000000000000e+00
 8    9    1.000000000000000e+00
 9    9    5.000000000000000e+00
 8   10    1.000000000000000e+00
 9   10    1.000000000000000e+00
10   10    5.000000000000000e+00
```

The eigenvalues of the matrix and the corresponding eigenvectors of $Ax = \lambda Bx$ (solved in buckling mode) are:

```
x =

  Columns 1 through 5

   3.5438e+01  -2.5103e+00  -2.0764e-02  -3.1979e-02   4.4245e-02
  -5.2703e+00  -1.1377e+01  -1.3268e-01  -2.4094e-01  -2.0414e-02
  -7.9829e+00  -1.3814e+00   6.5046e-01   6.6562e-01   3.3370e-01
   1.0268e+01   2.8591e+00   1.4481e-01   1.6671e-01   7.1408e-02
   4.0060e-01  -2.1217e+00  -6.1000e-01   4.8470e-01  -4.7222e-01
  -1.9140e+00   1.1085e+00  -3.2122e-01  -3.2611e-01  -1.7011e-01
   1.8990e+00  -1.1902e-01   1.1953e+00  -9.8936e-03  -4.7910e-01
   4.9968e-01   4.0858e-01  -1.0208e+00   2.1833e-01   1.0979e-01
  -8.7473e-01  -3.3476e-01   3.1877e-01  -1.1905e-01   2.3047e-01
   3.8013e-01   1.0474e-01   1.7809e-02   2.4609e-02  -1.4810e-01

  Columns 6 through 10

   3.1971e-02   2.4584e-04  -4.1247e-03   6.2310e-03   6.9070e-03
  -3.1244e-02   5.2514e-05   3.1142e-03   8.5694e-03   7.9613e-03
   3.7543e-02   1.4833e-04  -3.6371e-02  -2.4359e-02  -6.7201e-03
  -1.7783e-01  -9.9367e-04   3.1048e-02  -5.3438e-03   3.5841e-04
   4.5168e-02   9.6162e-03   7.3230e-02   1.4291e-02   1.4089e-03
   1.7243e-02   4.7478e-04   5.5651e-04  -2.5939e-02  -4.3783e-02
  -2.7519e-02  -2.1743e-02  -5.5717e-02  -4.3750e-03   9.2582e-03
  -4.3534e-02  -4.6547e-02  -1.0371e-01   2.1095e-03   7.9815e-03
   1.7702e-02  -3.5734e-03  -5.7321e-03   5.6394e-02  -1.8460e-02


eig =

   1.5581e-04
   1.4599e-03
   7.2754e-02
   2.2968e-01
   3.2065e-01
   5.7785e+00
   9.5544e+00
   9.7363e+00
   5.8110e+01
   8.6408e+01
```

drvsp1: Fortran source file (1 of 3)

```
      PROGRAM DRVSP1
C     ==============
C
C*********************************************************************
C*                                                                   *
C*    DRVSP1 is a driver model for the standard eigenvalue problem   *
C*                                                                   *
C*           (A)*(x)-eig*(x)=(0)                                     *
C*                                                                   *
C*    where (A) is a symmetric matrix of dimension N.                *
C*                                                                   *
C*    Data for DRVSP1 are read from the standard input (see below).  *
C*    The upper triangle of (A) is read from the file MATRXA (in     *
C*    coordinate format).                                            *
C*                                                                   *
C*    Examples: files 'drvsp1.dat' and 'A.dat'                       *
C*    BLAS kernel used: DSYMM                                        *
C*                                                                   *
C*********************************************************************
C
C.... parameters ..................................................
C
      INTEGER          LEIG
      PARAMETER        (LEIG   =     30)
      INTEGER          LISTOR
      PARAMETER        (LISTOR =  10000)
      INTEGER          LN
      PARAMETER        (LN     =    100)
      INTEGER          LRSTOR
      PARAMETER        (LRSTOR =  10000)
      INTEGER          NCUV
      PARAMETER        (NCUV   =      3)
      DOUBLE PRECISION ONE
      PARAMETER        (ONE    =  1.0D0)
      DOUBLE PRECISION ZERO
      PARAMETER        (ZERO   =  0.0D0)
C
C.... work variables ..............................................
C
      INTEGER          I,J,K
C
C.... DBLZDRV variables ...........................................
C
      INTEGER          ISTOR(LISTOR),LFLAG,NNEIG,NVOPU
      DOUBLE PRECISION EIG(LEIG,2),RSTOR(LRSTOR),SIGMA,
     &                 U(LN,NCUV),V(LN,NCUV),X(LN,LEIG)
C
C.... matrix (A) ..................................................
C
      INTEGER          N
      DOUBLE PRECISION A(LN,LN)
      CHARACTER        MATRXA*16
C
C=====================================================================
C
C.... initialize (A) ..............................................
C
      DO 10 I = 1,LN
         DO 20 J = 1,LN
            A(J,I) = ZERO
   20    CONTINUE
```

drvsp1: Fortran source file (2 of 3)

```
   10 CONTINUE
C
C.... read data ...................................................
C
      READ (*,ERR=1,FMT=*) ISTOR( 3)
      READ (*,ERR=1,FMT=*) ISTOR( 5)
      READ (*,ERR=1,FMT=*) ISTOR( 6)
      READ (*,ERR=1,FMT=*) ISTOR( 7)
      READ (*,ERR=1,FMT=*) ISTOR( 8)
      READ (*,ERR=1,FMT=*) ISTOR( 9)
      READ (*,ERR=1,FMT=*) ISTOR(10)
      READ (*,ERR=1,FMT=*) ISTOR(11)
      READ (*,ERR=1,FMT=*) ISTOR(12)
      READ (*,ERR=1,FMT=*) ISTOR(13)
      READ (*,ERR=1,FMT=*) RSTOR( 1)
      READ (*,ERR=1,FMT=*) RSTOR( 2)
      READ (*,ERR=1,FMT=*) RSTOR( 3)
      READ (*,ERR=1,FMT=*) MATRXA
C
C.... read (A) ....................................................
C
      N = 0
      OPEN (UNIT=10,ERR=2,STATUS='OLD',FILE=MATRXA)
      DO 30 I = 1,LN*LN
         READ (UNIT=10,ERR=2,END=40,FMT=*) J,K,A(J,K)
         N = MAX(J,K,N)
   30 CONTINUE
   40 CLOSE (UNIT=10,ERR=2)
C
C.... check the dimensions ........................................
C
      IF ( N .GT. LN ) STOP '* Error: N > LN *'
      IF ( ISTOR(5) .GT. NCUV ) STOP '* Error: NVB > NCUV *'
C
      ISTOR( 1) = N
      ISTOR( 2) = LN
      ISTOR( 4) = LEIG
      ISTOR(14) = 0
      ISTOR(15) = LISTOR
      RSTOR( 4) = LRSTOR
C
C.... reverse communication strategy ..............................
C
      LFLAG = 0
C
   50 CONTINUE
C
C     ********************************************************
      CALL DBLZDRV (ISTOR,RSTOR,SIGMA,NNEIG,U,V,LFLAG,NVOPU,EIG,X)
C     ********************************************************
C
      IF      ( LFLAG .LT. 0 ) THEN
C
C............ early  finalization ..................................
C
              WRITE (*,'(/A)') 'execution finished: abnormal exit'
C
      ELSE IF ( LFLAG .EQ. 0 ) THEN
C
C............ normal finalization ..................................
C
```

drvsp1: Fortran source file (3 of 3)

```
               WRITE (*,'(/A)') 'execution finished: standard exit'
C
      ELSE IF ( LFLAG .EQ. 1 ) THEN
C
C..... given (U), compute (V)=(A)*(U) ......................
C
               CALL DSYMM ('L','U',N,NVOPU,ONE,A,LN,U,LN,ZERO,V,LN)
C
               GO TO 50
C
      ELSE
C
C..... other flags should not be used here .................
C
               STOP '* Error: LFLAG does not apply in this case *'
C
      END IF
C
      STOP
    1 STOP '* IO error: standard input *'
    2 STOP '* IO error: file MATRXA *'
C
C**** end of DRVSP1 ************************************************
C
      END
```

drvsp1: input file (1 of 1)

```
         2     : istor( 3) = number of required eigenpairs
         2     : istor( 5) = number of vectors in a block
         0     : istor( 6) = maximum number of steps per run
         0     : istor( 7) = number of starting vectors as input
         0     : istor( 8) = number of eigenpairs given as input
         0     : istor( 9) = problem type
         0     : istor(10) = spectrum slicing
         0     : istor(11) = solutions refinement
         5     : istor(12) = level of printing
         6     : istor(13) = file unit for output
 0.00E+00      : rstor( 1) = interval inferior limit (EIGL)
 0.00E+00      : rstor( 2) = interval superior limit (EIGR)
 0.00E+00      : rstor( 3) = threshold for convergence
  'A.dat'      : matrix file
```

drvsp1: output file (1 of 2)

```
*********************************************
*                                           *
*                  BLZPACK                  *
*                ===========                *
*                                           *
*     Block Lanczos Algorithm Eigensolver   *
*          (real symmetric matrices)        *
*                                           *
*             - release 1999.11 -           *
*                                           *
*********************************************

dimension of the problem ........ =      10
number of required eigenpairs ... =       2
number of starting eigenpairs ... =       0
number of vectors in a block .... =       2
maximum number of eigenpairs .... =      10
maximum number of steps ......... =       5
lower limit for eigenvalues ..... = -4.1138E+62
upper limit for eigenvalues ..... = +4.1138E+62
starting point .................. = +0.0000E+00

---------------------------------------------------------------------------
run:  1
---------------------------------------------------------------------------

steps performed     :    5
eigenvalues accepted:   10

eigenvalue approximations and estimated residuals:
+6.8426E-04 (+2.22E-16) +6.5869E-03 (+2.22E-16) +2.8600E-01 (+2.22E-16)
+1.0713E+00 (+2.22E-16) +1.4875E+00 (+2.22E-16) +2.5446E+01 (+2.22E-16)
+4.1476E+01 (+2.22E-16) +5.1580E+01 (+2.22E-16) +2.3600E+02 (+2.22E-16)
+5.4565E+02 (+2.22E-16)

***************************************************************************

statistics
-------------------------------------------
number of runs .................. =        1
number of origin translations ... =        0
number of converged eigenpairs .. =       10
number of products for p.o. ..... =        2
number of products for s.o. ..... =        0
number of op(A)*vector .......... =       10
number of op(B)*vector .......... =        0
-------------------------------------------

time table
-------------------------------------------
op(A)*vectors .......... (  2.1%) = 1.19E-04
op(B)*vectors .......... (  0.0%) = 0.00E+00
factorizations ......... (  0.0%) = 0.00E+00
vectors generation ..... ( 29.6%) = 1.70E-03
reorthogonalizations ... ( 15.3%) = 8.74E-04
reduced problem ........ ( 15.2%) = 8.71E-04
Ritz vectors ........... (  2.0%) = 1.13E-04
total time ............. = 5.73E-03
-------------------------------------------

BLZPACK exit ************************************************************
```

drvsp1: output file (2 of 2)

```
eigenvalues
===================================
vector        value       residual
-----------------------------------
     1      +2.3600E+02  2.2204E-16
     2      +5.4565E+02  2.2204E-16
===================================

vector    1       value +2.3600E+02
===========================================================================
-8.5078E-02 -5.6587E-02 -3.2572E-01 -5.6472E-02 +2.0047E-01 -4.9224E-01
+9.9358E-02 +1.7727E-01 +7.3775E-01 +1.1051E-01

vector    2       value +5.4565E+02
===========================================================================
+4.7853E-02 +3.5710E-02 +2.5913E-01 +2.1773E-01 +1.8590E-01 +7.3282E-01
+4.5182E-02 +7.3665E-02 +5.4948E-01 +3.0040E-02

***************************************************************************

execution finished: standard exit
```

drvsp2: Fortran source file (1 of 3)

```
      PROGRAM DRVSP2
C     ==============
C
C*********************************************************************
C*                                                                   *
C*    DRVSP2 is a driver model for the standard eigenvalue problem   *
C*                                                                   *
C*           (A)*(x)-eig*(x)=(0)                                     *
C*                                                                   *
C*    where (A) is a symmetric matrix of dimension N. DRVSP2 uses    *
C*    eigenvalues/eigenvectors of (A) as input.                      *
C*                                                                   *
C*    Data for DRVSP2 are read from the standard input (see below).  *
C*    The upper triangle of (A) is read from the file MATRXA (in     *
C*    coordinate format) and the eigenvalues/eigenvectors of (A)     *
C*    are read from the file EIGENA.                                 *
C*                                                                   *
C*    Examples: files 'drvsp2.dat', 'A.dat' and 'A.eig.dat'          *
C*                                                                   *
C*********************************************************************
C
C.... parameters ..................................................
C
      INTEGER          LEIG
      PARAMETER        (LEIG   =     30)
      INTEGER          LISTOR
      PARAMETER        (LISTOR =  10000)
      INTEGER          LN
      PARAMETER        (LN     =    100)
      INTEGER          LRSTOR
      PARAMETER        (LRSTOR =  10000)
      INTEGER          MAXA
      PARAMETER        (MAXA   =  10000)
      INTEGER          NCUV
      PARAMETER        (NCUV   =      3)
      DOUBLE PRECISION ZERO
      PARAMETER        (ZERO   =  0.0D0)
C
C.... work variables ..............................................
C
      INTEGER          I,J,K,NGEIG
      DOUBLE PRECISION AJK
C
C.... DBLZDRV variables ...........................................
C
      INTEGER          ISTOR(LISTOR),LFLAG,NNEIG,NVOPU
      DOUBLE PRECISION EIG(LEIG,2),RSTOR(LRSTOR),SIGMA,
     &                 U(LN,NCUV),V(LN,NCUV),X(LN,LEIG)
C
C.... matrix (A) ..................................................
C
      INTEGER          IRN(MAXA),JCN(MAXA),N,NE
      DOUBLE PRECISION A(MAXA)
      CHARACTER        EIGENA*16,MATRXA*16
C
C=====================================================================
C
C.... read data ...................................................
C
      READ (*,ERR=1,FMT=*) ISTOR( 3)
      READ (*,ERR=1,FMT=*) ISTOR( 5)
```

drvsp2: Fortran source file (2 of 3)

```
      READ (*,ERR=1,FMT=*) ISTOR( 6)
      READ (*,ERR=1,FMT=*) ISTOR( 7)
      READ (*,ERR=1,FMT=*) ISTOR( 8)
      READ (*,ERR=1,FMT=*) ISTOR( 9)
      READ (*,ERR=1,FMT=*) ISTOR(10)
      READ (*,ERR=1,FMT=*) ISTOR(11)
      READ (*,ERR=1,FMT=*) ISTOR(12)
      READ (*,ERR=1,FMT=*) ISTOR(13)
      READ (*,ERR=1,FMT=*) RSTOR( 1)
      READ (*,ERR=1,FMT=*) RSTOR( 2)
      READ (*,ERR=1,FMT=*) RSTOR( 3)
      READ (*,ERR=1,FMT=*) MATRXA
      READ (*,ERR=1,FMT=*) EIGENA
C
C.... read (A) ....................................................
C
      N = 0
      NE = 0
      OPEN (UNIT=10,ERR=2,STATUS='OLD',FILE=MATRXA)
      DO 10 I = 1,LN*LN
         READ (UNIT=10,ERR=2,END=20,FMT=*) J,K,AJK
         NE = NE + 1
         IF ( NE .GT. MAXA ) STOP '* Error: NE > MAXA *'
         N = MAX(J,K,N)
         IRN(NE) = J
         JCN(NE) = K
         A(NE) = AJK
   10 CONTINUE
   20 CLOSE (UNIT=10,ERR=2)
C
C.... read eigenvalues and eigenvectors of (A) ....................
C
      NGEIG = ISTOR(8)
C
      OPEN (UNIT=10,ERR=3,STATUS='OLD',FILE=EIGENA)
      DO 30 I = 1,NGEIG
         READ (UNIT=10,ERR=3,FMT=*) EIG(I,1),EIG(I,2)
         READ (UNIT=10,ERR=3,FMT=*) (X(J,I),J=1,N)
   30 CONTINUE
      CLOSE (UNIT=10,ERR=3)
C
C.... check the dimensions ........................................
C
      IF ( N .GT. LN ) STOP '* Error: N > LN *'
      IF ( ISTOR(5) .GT. NCUV ) STOP '* Error: NVB > NCUV *'
C
      ISTOR( 1) = N
      ISTOR( 2) = LN
      ISTOR( 4) = LEIG
      ISTOR(14) = 0
      ISTOR(15) = LISTOR
      RSTOR( 4) = LRSTOR
C
C.... reverse communication strategy ..............................
C
      LFLAG = 0
C
   40 CONTINUE
C
C     ********************************************************
      CALL DBLZDRV (ISTOR,RSTOR,SIGMA,NNEIG,U,V,LFLAG,NVOPU,EIG,X)
```

drvsp2: Fortran source file (3 of 3)

```
C     ********************************************************
C
      IF      ( LFLAG .LT. 0 ) THEN
C
C............ early  finalization ....................................
C
              WRITE (*,'(/A)') 'execution finished: abnormal exit'
C
      ELSE IF ( LFLAG .EQ. 0 ) THEN
C
C............ normal finalization ....................................
C
              WRITE (*,'(/A)') 'execution finished: standard exit'
C
      ELSE IF ( LFLAG .EQ. 1 ) THEN
C
C............ given (U), compute (V)=(A)*(U) ........................
C
              DO 70 I = 1,NVOPU
                 DO 50 J = 1,N
                    V(J,I) = ZERO
   50            CONTINUE
                 DO 60 J = 1,NE
                    IF ( IRN(J) .EQ. JCN(J) ) THEN
                       V(IRN(J),I) = V(IRN(J),I) + A(J)*U(JCN(J),I)
                    ELSE
                       V(IRN(J),I) = V(IRN(J),I) + A(J)*U(JCN(J),I)
                       V(JCN(J),I) = V(JCN(J),I) + A(J)*U(IRN(J),I)
                    END IF
   60            CONTINUE
   70         CONTINUE
C
              GO TO 40
C
      ELSE
C
C............ other flags should not be used here ......................
C
              STOP '* Error: LFLAG does not apply in this case *'
C
      END IF
C
      STOP
    1 STOP '* IO error: standard input *'
    2 STOP '* IO error: file MATRXA *'
    3 STOP '* IO error: file EIGENA *'
C
C**** end of DRVSP2 ***************************************************
C
      END
```

drvsp2: input file (1 of 1):

```
         2     : istor( 3) = number of required eigenpairs
         2     : istor( 5) = number of vectors in a block
         0     : istor( 6) = maximum number of steps per run
         0     : istor( 7) = number of starting vectors as input
         1     : istor( 8) = number of eigenpairs given as input
         0     : istor( 9) = problem type
         0     : istor(10) = spectrum slicing
         0     : istor(11) = solutions refinement
         5     : istor(12) = level of printing
         6     : istor(13) = file unit for output
  0.00E+00     : rstor( 1) = interval inferior limit (EIGL)
  0.00E+00     : rstor( 2) = interval superior limit (EIGR)
  0.00E+00     : rstor( 3) = threshold for convergence
    'A.dat'    : matrix file
  'A.eig.1'    : eigenvalues and eigenvectors of the matrix
```

A.eig.dat: input file (1 of 1):

```
  5.4564533582057970e+02   3.8104831792003235e-16     (eig_10,x_10)
  4.7852821884759929e-02
  3.5709952158645537e-02
  2.5912760294987597e-01
  2.1773174414782256e-01
  1.8590489207706720e-01
  7.3281636378667658e-01
  4.5182413852188956e-02
  7.3665045613005270e-02
  5.4948139937132801e-01
  3.0040474044674737e-02
  2.3600107432808124e+02   3.3241347894312373e-16     (eig_9,x_9)
 -8.5078085576099929e-02
 -5.6587266855303825e-02
 -3.2572247553385364e-01
 -5.6472128462583815e-02
  2.0046878654538500e-01
 -4.9224306789492428e-01
  9.9357910636162713e-02
  1.7727466998262675e-01
  7.3774829094796246e-01
  1.1051494077226823e-01
  5.1579911489517983e+01   7.1050273616605535e-17     (eig_8,x_8)
  9.0143625539717415e-02
 -1.7667262491263883e-02
  1.4357140160984186e-01
 -2.7939322625274399e-01
 -3.2156411733617646e-01
  6.8177035827888713e-02
  2.2155557618221625e-01
  4.7527538595852742e-01
 -6.6631002032673942e-02
  7.1093166080480452e-01
```

drvsp2: output file (1 of 2):

```
*********************************************
*                                           *
*                  BLZPACK                  *
*                ===========                *
*                                           *
*     Block Lanczos Algorithm Eigensolver   *
*          (real symmetric matrices)        *
*                                           *
*             - release 1999.11 -           *
*                                           *
*********************************************

dimension of the problem ........ =      10
number of required eigenpairs ... =       2
number of starting eigenpairs ... =       1
number of vectors in a block .... =       2
maximum number of eigenpairs .... =      10
maximum number of steps ......... =       5
lower limit for eigenvalues ..... = -4.1138E+62
upper limit for eigenvalues ..... = +4.1138E+62
starting point .................. = +0.0000E+00

---------------------------------------------------------------------------
run:  1
---------------------------------------------------------------------------

steps performed     :    5
eigenvalues accepted:    9

eigenvalue approximations and estimated residuals:
+0.0000E+00 (-4.11E+62) +6.8426E-04 (+2.22E-16) +6.5869E-03 (+2.22E-16)
+2.8600E-01 (+2.22E-16) +1.0713E+00 (+2.22E-16) +1.4875E+00 (+2.22E-16)
+2.5446E+01 (+2.22E-16) +4.1476E+01 (+2.22E-16) +5.1580E+01 (+2.22E-16)
+2.3600E+02 (+2.22E-16)

***************************************************************************

statistics
-------------------------------------------
number of runs .................. =        1
number of origin translations ... =        0
number of converged eigenpairs .. =        9
number of products for p.o. ..... =        2
number of products for s.o. ..... =        4
number of op(A)*vector .......... =       10
number of op(B)*vector .......... =        0
-------------------------------------------

time table
-------------------------------------------
op(A)*vectors .......... (  2.0%) = 1.15E-04
op(B)*vectors .......... (  0.0%) = 0.00E+00
factorizations ......... (  0.0%) = 0.00E+00
vectors generation ..... ( 29.5%) = 1.70E-03
reorthogonalizations ... ( 17.5%) = 1.01E-03
reduced problem ........ ( 13.6%) = 7.83E-04
Ritz vectors ........... (  1.8%) = 1.02E-04
total time ..................... = 5.77E-03
-------------------------------------------

BLZPACK exit ************************************************************
```

drvsp2: output file (2 of 2):

```
eigenvalues
===================================
vector        value       residual
-----------------------------------
     1      +5.1580E+01  2.2204E-16
     2      +2.3600E+02  2.2204E-16
===================================

vector    1       value +5.1580E+01
=====================================================================
+9.0144E-02 -1.7667E-02 +1.4357E-01 -2.7939E-01 -3.2156E-01 +6.8177E-02
+2.2156E-01 +4.7528E-01 -6.6631E-02 +7.1093E-01

vector    2       value +2.3600E+02
=====================================================================
-8.5078E-02 -5.6587E-02 -3.2572E-01 -5.6472E-02 +2.0047E-01 -4.9224E-01
+9.9358E-02 +1.7727E-01 +7.3775E-01 +1.1051E-01

***************************************************************************

execution finished: standard exit
```

drvgp1: Fortran source file (1 of 4):

```
      PROGRAM DRVGP1
C     ==============
C
C***********************************************************************
C*                                                                     *
C*    DRVGP1 is a driver model for the generalized eigenvalue problem  *
C*                                                                     *
C*           (A)*(x)-eig*(B)*(x)=(0)                                   *
C*                                                                     *
C*    where (A) is a symmetric matrix of dimension N and (B) is set    *
C*    to the identity matrix (for demonstration purposes only).        *
C*                                                                     *
C*    Data for DRVGP1 are read from the standard input (see below).    *
C*    The upper triangle of (A) is read from the file MATRXA (in       *
C*    coordinate format).                                              *
C*                                                                     *
C*    Examples: files 'drvgp1.dat' and 'A.dat'                         *
C*    Factorization and solver used: MA47AD, MA47BD, MA47CD, MA47ID    *
C*    BLAS kernel used: DCOPY                                          *
C*                                                                     *
C***********************************************************************
C
C.... parameters ......................................................
C
      INTEGER          LEIG
      PARAMETER        (LEIG   =     30)
      INTEGER          LISTOR
      PARAMETER        (LISTOR =  10000)
      INTEGER          LN
      PARAMETER        (LN     =    100)
      INTEGER          LRSTOR
      PARAMETER        (LRSTOR =  10000)
      INTEGER          MAXA
      PARAMETER        (MAXA   =  10000)
      INTEGER          MAXIW1
      PARAMETER        (MAXIW1 =  10000)
      INTEGER          MAXNE
      PARAMETER        (MAXNE  =   1000)
      INTEGER          NCUV
      PARAMETER        (NCUV   =      3)
      DOUBLE PRECISION ONE
      PARAMETER        (ONE    =  1.0D0)
C
C.... work variables ..................................................
C
      INTEGER          I,J,K
      DOUBLE PRECISION AJK
C
C.... DBLZDRV variables ...............................................
C
      INTEGER          ISTOR(LISTOR),LFLAG,NNEIG,NVOPU
      DOUBLE PRECISION EIG(LEIG,2),RSTOR(LRSTOR),SIGMA,
     &                 U(LN,NCUV),V(LN,NCUV),X(LN,LEIG)
C
C.... matrices (A) and (B) ............................................
C
      INTEGER          ICNTL(7),INFO(24),IRN(MAXNE),IW1(MAXIW1),
     &                 IW2(LN*2+2),JCN(MAXNE),KEEP(MAXNE+LN*5+2),N,NE
      DOUBLE PRECISION A(MAXA),B(LN),C(MAXA),CNTL(2),RINFO(4),W(LN)
      CHARACTER        MATRXA*16
C
```

drvgp1: Fortran source file (2 of 4):

```
C=======================================================================
C
C.... read data .......................................................
C
      READ (*,ERR=1,FMT=*) ISTOR( 3)
      READ (*,ERR=1,FMT=*) ISTOR( 5)
      READ (*,ERR=1,FMT=*) ISTOR( 6)
      READ (*,ERR=1,FMT=*) ISTOR( 7)
      READ (*,ERR=1,FMT=*) ISTOR( 8)
      READ (*,ERR=1,FMT=*) ISTOR( 9)
      READ (*,ERR=1,FMT=*) ISTOR(10)
      READ (*,ERR=1,FMT=*) ISTOR(11)
      READ (*,ERR=1,FMT=*) ISTOR(12)
      READ (*,ERR=1,FMT=*) ISTOR(13)
      READ (*,ERR=1,FMT=*) RSTOR( 1)
      READ (*,ERR=1,FMT=*) RSTOR( 2)
      READ (*,ERR=1,FMT=*) RSTOR( 3)
      READ (*,ERR=1,FMT=*) MATRXA
C
C.... read (A) ........................................................
C
      N = 0
      NE = 0
      OPEN (UNIT=10,ERR=2,STATUS='OLD',FILE=MATRXA)
      DO 10 I = 1,LN*LN
         READ (UNIT=10,ERR=2,END=20,FMT=*) J,K,AJK
         NE = NE + 1
         IF ( NE .GT. MAXA ) STOP '* Error: NE > MAXA *'
         IF ( NE .GT. MAXNE ) STOP '* Error: NE > MAXNE *'
         N = MAX(J,K,N)
         IRN(NE) = J
         JCN(NE) = K
         A(NE) = AJK
   10 CONTINUE
   20 CLOSE (UNIT=10,ERR=2)
C
C.... check the dimensions ............................................
C
      IF ( N .GT. LN ) STOP '* Error: N > LN *'
      IF ( ISTOR(5) .GT. NCUV ) STOP '* Error: NVB > NCUV *'
C
      ISTOR( 1) = N
      ISTOR( 2) = LN
      ISTOR( 4) = LEIG
      ISTOR(14) = 0
      ISTOR(15) = LISTOR
      RSTOR( 4) = LRSTOR
C
C.... generate an identity (B) ........................................
C
      DO 30 I = 1,N
         B(I) = ONE
   30 CONTINUE
C
C.... set default parameters for MA47 and analyse sparsity pattern .....
C
      CALL MA47ID (CNTL,ICNTL)
C
      CALL MA47AD (N,NE,IRN,JCN,IW1,MAXIW1,KEEP,ICNTL,RINFO,INFO)
C
      IF ( INFO(1).NE.0 ) STOP '* Error: MA47AD, INFO(1)>0 *'
```

drvgp1: Fortran source file (3 of 4):

```
C
C.... reverse communication strategy ..................................
C
      LFLAG = 0
C
   40 CONTINUE
C
C     ********************************************************
      CALL DBLZDRV (ISTOR,RSTOR,SIGMA,NNEIG,U,V,LFLAG,NVOPU,EIG,X)
C     ********************************************************
C
      IF      ( LFLAG .LT. 0 ) THEN
C
C..... early  finalization ....................................
C
              WRITE (*,'(/A)') 'execution finished: abnormal exit'
C
      ELSE IF ( LFLAG .EQ. 0 ) THEN
C
C..... normal finalization ....................................
C
              WRITE (*,'(/A)') 'execution finished: standard exit'
C
      ELSE IF ( LFLAG .EQ. 1 ) THEN
C
C..... given (U), solve (C)*(V)=(U) ...........................
C
              DO 50 I = 1,NVOPU
                 CALL DCOPY  (N,U(1,I),1,V(1,I),1)
                 CALL MA47CD (N,C,MAXA,IW1,MAXIW1,W,V(1,I),IW2,ICNTL)
   50         CONTINUE
C
              GO TO 40
C
      ELSE IF ( LFLAG .EQ. 2 ) THEN
C
C..... given (U), compute (V)=(B)*(U) .........................
C
              DO 70 I = 1,NVOPU
                 DO 60 J = 1,N
                    V(J,I) = B(J)*U(J,I)
   60            CONTINUE
   70         CONTINUE
C
              GO TO 40
C
      ELSE IF ( LFLAG .EQ. 3 ) THEN
C
C..... given SIGMA, form (C)=(A)-SIGMA*(B) ....................
C
              DO 80 I = 1,NE
                 C(I) = A(I)
                 IF ( IRN(I).EQ.JCN(I) ) C(I) = C(I) - B(IRN(I))*SIGMA
   80         CONTINUE
C
C..... factor (C)=(L)*(D)*(L') ................................
C
              CALL MA47BD (N,NE,JCN,C,MAXA,IW1,MAXIW1,KEEP,
     &                     CNTL,ICNTL,IW2,RINFO,INFO)
C
              IF ( INFO( 1).NE.0 ) STOP '* Error: MA47BD, INFO(1)>0 *'
```

drvgp1: Fortran source file (4 of 4):

```
              IF ( INFO(24).GT.0 ) STOP '* Error: MA47BD, INFO(24)>0 *'
C
              NNEIG = INFO(23)
C
              GO TO 40
C
      END IF
C
      STOP
    1 STOP '* IO error: standard input *'
    2 STOP '* IO error: file MATRXA *'
C
C**** end of DRVGP1 ***************************************************
C
      END
```

drvgp1: input file (1 of 1):

```
         2     : istor( 3) = number of required eigenpairs
         2     : istor( 5) = number of vectors in a block
         0     : istor( 6) = maximum number of steps per run
         0     : istor( 7) = number of starting vectors as input
         0     : istor( 8) = number of eigenpairs given as input
         1     : istor( 9) = problem type
         0     : istor(10) = spectrum slicing
         0     : istor(11) = solutions refinement
         5     : istor(12) = level of printing
         6     : istor(13) = file unit for output
  0.00E+00     : rstor( 1) = interval inferior limit (EIGL)
  0.00E+00     : rstor( 2) = interval superior limit (EIGR)
  0.00E+00     : rstor( 3) = threshold for convergence
   'A.dat'     : matrix file
```

drvgp1: output file (1 of 2):

```
*******************************************
*                                         *
*                  BLZPACK                *
*                ===========              *
*                                         *
*     Block Lanczos Algorithm Eigensolver *
*          (real symmetric matrices)      *
*                                         *
*             - release 1999.11 -         *
*                                         *
*******************************************

dimension of the problem ........ =      10
number of required eigenpairs ... =       2
number of starting eigenpairs ... =       0
number of vectors in a block .... =       2
maximum number of eigenpairs .... =      10
maximum number of steps ......... =       5
lower limit for eigenvalues ..... = -4.1138E+62
upper limit for eigenvalues ..... = +4.1138E+62
starting point .................. = +0.0000E+00

======================================================================
factorization:  1     SIGMA: +0.0000E+00     eigenvalues required:    2
======================================================================

eigenvalues smaller than SIGMA:       0
lower boundary for eigenvalues: +0.0000E+00
upper boundary for eigenvalues: +4.1138E+62

----------------------------------------------------------------------
run:  1  (origin in EIGL=EIGR)
----------------------------------------------------------------------

steps performed     :    5
eigenvalues accepted:   10

eigenvalue approximations and estimated residuals:
+6.8426E-04 (+2.22E-16) +6.5869E-03 (+2.22E-16) +2.8600E-01 (+2.22E-16)
+1.0713E+00 (+2.22E-16) +1.4875E+00 (+2.22E-16) +2.5446E+01 (+2.22E-16)
+4.1476E+01 (+2.22E-16) +5.1580E+01 (+2.22E-16) +2.3600E+02 (+2.22E-16)
+5.4565E+02 (+2.22E-16)

**********************************************************************

statistics
--------------------------------------------
number of runs .................. =        1
number of origin translations ... =        1
number of converged eigenpairs .. =       10
number of products for p.o. ..... =        2
number of products for s.o. ..... =        0
number of op(A)*vector .......... =       10
number of op(B)*vector .......... =       10
--------------------------------------------

time table
--------------------------------------------
op(A)*vectors ..... (  4.0%) = 2.61E-04
op(B)*vectors ..... (  1.1%) = 7.23E-05
factorizations ..... (  2.7%) = 1.73E-04
```

drvgp1: output file (2 of 2):

```
vectors generation ..... ( 26.2%) = 1.71E-03
reorthogonalizations ... ( 15.0%) = 9.76E-04
reduced problem ........ ( 12.0%) = 7.85E-04
Ritz vectors ........... (  1.7%) = 1.08E-04
total time ..................... = 6.51E-03
-----------------------------------------

BLZPACK exit ************************************************************

eigenvalues
===================================
vector        value       residual
-----------------------------------
     1      +6.8426E-04  2.2204E-16
     2      +6.5869E-03  2.2204E-16
===================================

vector    1       value +6.8426E-04
======================================================================
-9.2653E-01 +1.3982E-01 +2.0911E-01 -2.6908E-01 -1.0119E-02 +4.9844E-02
-4.9624E-02 -1.3164E-02 +2.2950E-02 -9.9630E-03

vector    2       value +6.5869E-03
======================================================================
-2.2281E-01 -9.2065E-01 -1.0679E-01 +2.2677E-01 -1.7275E-01 +9.0482E-02
-9.9661E-03 +3.2313E-02 -2.6451E-02 +8.2702E-03

**********************************************************************

execution finished: standard exit
```

drvgp4: Fortran source file (1 of 4):

```
      PROGRAM DRVGP4
C     ==============
C
C***********************************************************************
C*                                                                     *
C*    DRVGP4 is a driver model for the generalized eigenvalue problem  *
C*                                                                     *
C*           (A)*(x)-eig*(B)*(x)=(0)                                   *
C*                                                                     *
C*    where (A) is a symmetric matrix of dimension N and (B) has the   *
C*    same pattern of (A). The problem is solved in buckling mode.     *
C*                                                                     *
C*    Data for DRVGP4 are read from the standard input (see below).    *
C*    The upper triangle of (A) is read from the file MATRXA (in       *
C*    coordinate format) and the upper triangle of (B) is read         *
C*    from the file MATRXB (in coordinate format).                     *
C*                                                                     *
C*    Examples: files 'drvgp4.dat', 'A.dat' and 'B2.dat'               *
C*    Factorization and solver used: MA47AD, MA47BD, MA47CD, MA47ID    *
C*    BLAS kernels used: DAXPY,DCOPY                                   *
C*                                                                     *
C***********************************************************************
C
C.... parameters ....................................................
C
      INTEGER          LEIG
      PARAMETER        (LEIG   =     30)
      INTEGER          LISTOR
      PARAMETER        (LISTOR =  10000)
      INTEGER          LN
      PARAMETER        (LN     =    110)
      INTEGER          LRSTOR
      PARAMETER        (LRSTOR =  10000)
      INTEGER          MAXA
      PARAMETER        (MAXA   =  10000)
      INTEGER          MAXIW1
      PARAMETER        (MAXIW1 =  10000)
      INTEGER          MAXNE
      PARAMETER        (MAXNE  =   1000)
      INTEGER          NCUV
      PARAMETER        (NCUV   =      3)
      DOUBLE PRECISION ZERO
      PARAMETER        (ZERO   =  0.0D0)
C
C.... work variables ................................................
C
      INTEGER          I,J,K,L
      DOUBLE PRECISION AJK,BJK
C
C.... DBLZDRV variables .............................................
C
      INTEGER          ISTOR(LISTOR),LFLAG,NNEIG,NVOPU
      DOUBLE PRECISION EIG(LEIG,2),RSTOR(LRSTOR),SIGMA,
     &                 U(LN,NCUV),V(LN,NCUV),X(LN,LEIG)
C
C.... matrices (A) and (B) ..........................................
C
      INTEGER          ICNTL(7),INFO(24),IRN(MAXNE),IW1(MAXIW1),
     &                 IW2(LN*2+2),JCN(MAXNE),KEEP(MAXNE+LN*5+2),
     &                 N,NE,NNEIGB
      DOUBLE PRECISION A(MAXA),B(MAXA),C(MAXA),CNTL(2),RINFO(4),W(LN)
```

drvgp4: Fortran source file (2 of 4):

```
      CHARACTER        MATRXA*16,MATRXB*16
C
C=======================================================================
C
C.... read data .....................................................
C
      READ (*,ERR=1,FMT=*) ISTOR( 3)
      READ (*,ERR=1,FMT=*) ISTOR( 5)
      READ (*,ERR=1,FMT=*) ISTOR( 6)
      READ (*,ERR=1,FMT=*) ISTOR( 7)
      READ (*,ERR=1,FMT=*) ISTOR( 8)
      READ (*,ERR=1,FMT=*) ISTOR( 9)
      READ (*,ERR=1,FMT=*) ISTOR(10)
      READ (*,ERR=1,FMT=*) ISTOR(11)
      READ (*,ERR=1,FMT=*) ISTOR(12)
      READ (*,ERR=1,FMT=*) ISTOR(13)
      READ (*,ERR=1,FMT=*) RSTOR( 1)
      READ (*,ERR=1,FMT=*) RSTOR( 2)
      READ (*,ERR=1,FMT=*) RSTOR( 3)
      READ (*,ERR=1,FMT=*) MATRXA
      READ (*,ERR=1,FMT=*) MATRXB
C
C.... read (A) ......................................................
C
      N = 0
      NE = 0
      OPEN (UNIT=10,ERR=2,STATUS='OLD',FILE=MATRXA)
      DO 10 I = 1,LN*LN
         READ (UNIT=10,ERR=2,END=20,FMT=*) J,K,AJK
         NE = NE + 1
         IF ( NE .GT. MAXA ) STOP '* Error: NE > MAXA *'
         IF ( NE .GT. MAXNE ) STOP '* Error: NE > MAXNE *'
         N = MAX(J,K,N)
         IRN(NE) = J
         JCN(NE) = K
         A(NE) = AJK
         B(I) = ZERO
   10 CONTINUE
   20 CLOSE (UNIT=10,ERR=2)
C
C.... check the dimensions ..........................................
C
      IF ( N .GT. LN ) STOP '* Error: N > LN *'
      IF ( ISTOR(5) .GT. NCUV ) STOP '* Error: NVB > NCUV *'
C
      ISTOR( 1) = N
      ISTOR( 2) = LN
      ISTOR( 4) = LEIG
      ISTOR(14) = 0
      ISTOR(15) = LISTOR
      RSTOR( 4) = LRSTOR
C
C.... read (B), same pattern of (A) .................................
C
      L = 1
      OPEN (UNIT=10,ERR=3,STATUS='OLD',FILE=MATRXB)
   30 CONTINUE
      READ (UNIT=10,ERR=3,END=50,FMT=*) J,K,BJK
      DO 40 I = L,NE
         IF ( J.EQ.IRN(I) .AND. K.EQ.JCN(I) ) THEN
            B(I) = BJK
```

drvgp4: Fortran source file (3 of 4):

```
            L = I + 1
            GO TO 30
          END IF
   40 CONTINUE
      STOP '* Error: Patterns of (A) and (B) differ *'
   50 CLOSE (UNIT=10,ERR=3)
C
C.... set default parameters for MA47 and analyse sparsity pattern .....
C
      CALL MA47ID (CNTL,ICNTL)
C
      CALL MA47AD (N,NE,IRN,JCN,IW1,MAXIW1,KEEP,ICNTL,RINFO,INFO)
C
      IF ( INFO(1).NE.0 ) STOP '* Error: MA47AD, INFO(1)>0 *'
C
C.... compute the number of negative eigenvalues of (B) ................
C
      CALL DCOPY  (NE,B,1,C,1)
C
C.... factor (C)=(L)*(D)*(L') ..........................................
C
      CALL MA47BD (N,NE,JCN,C,MAXA,IW1,MAXIW1,KEEP,
     &             CNTL,ICNTL,IW2,RINFO,INFO)
C
      IF ( INFO( 1).NE.0 ) STOP '* Error: MA47BD, INFO(1)>0 *'
      IF ( INFO(24).GT.0 ) STOP '* Error: MA47BD, INFO(24)>0 *'
C
      NNEIGB = INFO(23)
C
C.... reverse communication strategy ...................................
C
      LFLAG = 0
C
   60 CONTINUE
C
C     ************************************************************
      CALL DBLZDRV (ISTOR,RSTOR,SIGMA,NNEIG,U,V,LFLAG,NVOPU,EIG,X)
C     ************************************************************
C
      IF      ( LFLAG .LT. 0 ) THEN
C
C..... early  finalization .....................................
C
               WRITE (*,'(/A)') 'execution finished: abnormal exit'
C
      ELSE IF ( LFLAG .EQ. 0 ) THEN
C
C..... normal finalization .....................................
C
               WRITE (*,'(/A)') 'execution finished: standard exit'
C
      ELSE IF ( LFLAG .EQ. 1 ) THEN
C
C..... given (U), solve (C)*(V)=(U) ............................
C
               DO 70 I = 1,NVOPU
                  CALL DCOPY  (N,U(1,I),1,V(1,I),1)
                  CALL MA47CD (N,C,MAXA,IW1,MAXIW1,W,V(1,I),IW2,ICNTL)
   70          CONTINUE
C
               GO TO 60
```

drvgp4: Fortran source file (4 of 4):

```
C
      ELSE IF ( LFLAG .EQ. 2 ) THEN
C
C..... given (U), compute (V)=(A)*(U) ..........................
C
               DO 100 I = 1,NVOPU
                  DO 80 J = 1,N
                     V(J,I) = ZERO
   80             CONTINUE
                  DO 90 J = 1,NE
                     IF ( IRN(J) .EQ. JCN(J) ) THEN
                        V(IRN(J),I) = V(IRN(J),I) + A(J)*U(JCN(J),I)
                     ELSE
                        V(IRN(J),I) = V(IRN(J),I) + A(J)*U(JCN(J),I)
                        V(JCN(J),I) = V(JCN(J),I) + A(J)*U(IRN(J),I)
                     END IF
   90             CONTINUE
  100          CONTINUE
C
               GO TO 60
C
      ELSE IF ( LFLAG .EQ. 3 ) THEN
C
C..... given SIGMA, form (C)=(A)-SIGMA*(B) .....................
C
               CALL DCOPY (NE,A,1,C,1)
               CALL DAXPY (NE,-SIGMA,B,1,C,1)
C
C..... factor (C)=(L)*(D)*(L') .................................
C
               CALL MA47BD (N,NE,JCN,C,MAXA,IW1,MAXIW1,KEEP,
     &                      CNTL,ICNTL,IW2,RINFO,INFO)
C
               IF ( INFO( 1).NE.0 ) STOP '* Error: MA47BD, INFO(1)>0 *'
               IF ( INFO(24).GT.0 ) STOP '* Error: MA47BD, INFO(24)>0 *'
C
               IF      ( SIGMA .LT. ZERO ) THEN
                       NNEIG = NNEIGB - INFO(23)
               ELSE IF ( SIGMA .GT. ZERO ) THEN
                       NNEIG = NNEIGB + INFO(23)
               ELSE
                       NNEIG = NNEIGB
               END IF
C
               GO TO 60
C
      END IF
C
      STOP
    1 STOP '* IO error: standard input *'
    2 STOP '* IO error: file MATRXA *'
    3 STOP '* IO error: file MATRXB *'
C
C**** end of DRVGP4 ************************************************
C
      END
```

drvgp4: input file (1 of 1):

```
         2     : istor( 3) = number of required eigenpairs
         2     : istor( 5) = number of vectors in a block
         0     : istor( 6) = maximum number of steps per run
         0     : istor( 7) = number of starting vectors as input
         0     : istor( 8) = number of eigenpairs given as input
         2     : istor( 9) = problem type
         0     : istor(10) = spectrum slicing
         0     : istor(11) = solutions refinement
         5     : istor(12) = level of printing
         6     : istor(13) = file unit for output
  0.00E+00     : rstor( 1) = interval inferior limit (EIGL)
  0.00E+00     : rstor( 2) = interval superior limit (EIGR)
  0.00E+00     : rstor( 3) = threshold for convergence
   'A.dat'     : matrix file
  'B2.dat'     : matrix file
```

drvgp4: output file (1 of 2):

```
*********************************************
*                                           *
*                  BLZPACK                  *
*                ===========                *
*                                           *
*     Block Lanczos Algorithm Eigensolver   *
*          (real symmetric matrices)        *
*                                           *
*             - release 1999.11 -           *
*                                           *
*********************************************

dimension of the problem ........ =      10
number of required eigenpairs ... =       2
number of starting eigenpairs ... =       0
number of vectors in a block .... =       2
maximum number of eigenpairs .... =      10
maximum number of steps ......... =       5
lower limit for eigenvalues ..... = -4.1138E+62
upper limit for eigenvalues ..... = +4.1138E+62
starting point .................. = -1.0000E-01

======================================================================
factorization:  1     SIGMA: -1.0000E-01     eigenvalues required:    2
======================================================================

eigenvalues smaller than SIGMA:       0
lower boundary for eigenvalues: -1.0000E-01
upper boundary for eigenvalues: +4.1138E+62

----------------------------------------------------------------------
run:  1  (origin in EIGL=EIGR)
----------------------------------------------------------------------

steps performed     :    5
eigenvalues accepted:   10

eigenvalue approximations and estimated residuals:
+1.5581E-04 (+2.22E-16) +1.4599E-03 (+2.22E-16) +7.2754E-02 (+2.22E-16)
+2.2968E-01 (+2.22E-16) +3.2065E-01 (+2.22E-16) +5.7785E+00 (+2.22E-16)
+9.5544E+00 (+2.22E-16) +9.7363E+00 (+2.22E-16) +5.8110E+01 (+2.22E-16)
+8.6408E+01 (+2.22E-16)

**********************************************************************

statistics
--------------------------------------------
number of runs .................. =        1
number of origin translations ... =        1
number of converged eigenpairs .. =       10
number of products for p.o. ..... =        1
number of products for s.o. ..... =        0
number of op(A)*vector .......... =       10
number of op(B)*vector .......... =       10
--------------------------------------------

time table
--------------------------------------------
op(A)*vectors ..... (  4.0%) = 2.52E-04
op(B)*vectors ..... (  2.3%) = 1.46E-04
factorizations ..... (  2.8%) = 1.76E-04
```

drvgp4: output file (2 of 2):

```
vectors generation ..... ( 27.6%) = 1.74E-03
reorthogonalizations ... ( 12.8%) = 8.08E-04
reduced problem ........ ( 10.3%) = 6.48E-04
Ritz vectors ........... (  1.9%) = 1.20E-04
total time ..................... = 6.31E-03
-----------------------------------------

BLZPACK exit ************************************************************

eigenvalues
===================================
vector        value       residual
-----------------------------------
     1      +1.5581E-04  2.2204E-16
     2      +1.4599E-03  2.2204E-16
===================================

vector    1       value +1.5581E-04
======================================================================
+3.5438E+01 -5.2703E+00 -7.9829E+00 +1.0268E+01 +4.0060E-01 -1.9140E+00
+1.8990E+00 +4.9968E-01 -8.7473E-01 +3.8013E-01

vector    2       value +1.4599E-03
======================================================================
-2.5103E+00 -1.1377E+01 -1.3814E+00 +2.8591E+00 -2.1217E+00 +1.1085E+00
-1.1902E-01 +4.0858E-01 -3.3476E-01 +1.0474E-01

**********************************************************************

execution finished: standard exit
```

drvmpi: Fortran source file (1 of 5):

```
      PROGRAM DRVMPI
C     =============
C
C**********************************************************************
C*                                                                    *
C*    DRVMPI is a driver model for the standard eigenvalue problem    *
C*                                                                    *
C*           (A)*(x)-eig*(x)=(0)                                      *
C*                                                                    *
C*    where (A) is a symmetric matrix of dimension N. DRVMPI is the   *
C*    parallel, MPI based, version of DRVSP1.                         *
C*                                                                    *
C*    Data for DRVMPI are read from 'drvsp1.dat' (see below). The     *
C*    upper triangle of (A) is read from the file MATRXA (in          *
C*    coordinate format).                                             *
C*                                                                    *
C*    Examples: files 'drvsp1.dat' and 'A.dat'                        *
C*    BLAS kernel used: DGEMM                                         *
C*                                                                    *
C**********************************************************************
C
C.... parameters ......................................................
C
      INTEGER          LEIG
      PARAMETER        (LEIG   =     30)
      INTEGER          LISTOR
      PARAMETER        (LISTOR =  10000)
      INTEGER          LN
      PARAMETER        (LN     =    100)
      INTEGER          LRSTOR
      PARAMETER        (LRSTOR =  10000)
      INTEGER          NCUV
      PARAMETER        (NCUV   =      3)
      INTEGER          NNZMAX
      PARAMETER        (NNZMAX =   4000)
      DOUBLE PRECISION ONE
      PARAMETER        (ONE    =  1.0D0)
      DOUBLE PRECISION ZERO
      PARAMETER        (ZERO   =  0.0D0)
C
C.... work variables ..................................................
C
      INTEGER          I,J,K
C
C.... BLZDRS variables ................................................
C
      INTEGER          ISTOR(LISTOR),LFLAG,NNEIG,NVOPU
      DOUBLE PRECISION EIG(LEIG,2),RSTOR(LRSTOR),SIGMA,T(LN*NCUV),
     &                 U(LN*NCUV),V(LN*NCUV),X(LN*LEIG)
C
C.... matrix (A) ......................................................
C
      INTEGER          ICOL(NNZMAX),ICOLL,ICOLR,IROW(NNZMAX),
     &                 N,NCOLL,NCOLP,NNZ,RMNDR
      DOUBLE PRECISION A(LN,LN),S(NNZMAX)
      CHARACTER        MATRXA*16
C
C.... MPI .............................................................
C
      INTEGER          II,INFO,JJ,MYPE,NI,NJ,NPE,ROOT,STATUS(10)
C
```

drvmpi: Fortran source file (2 of 5):

```
      INCLUDE          'mpif.h'
C
C=====================================================================
C
      CALL MPI_INIT      (INFO)
      CALL MPI_COMM_SIZE (MPI_COMM_WORLD,NPE ,INFO)
      CALL MPI_COMM_RANK (MPI_COMM_WORLD,MYPE,INFO)
C
      IF ( MYPE .EQ. 0 ) THEN
C
C....... read data ....................................................
C
         OPEN  (UNIT=10,ERR=1,STATUS='OLD',FILE='drvsp1.dat')
         READ  (UNIT=10,ERR=1,FMT=*) ISTOR( 3)
         READ  (UNIT=10,ERR=1,FMT=*) ISTOR( 5)
         READ  (UNIT=10,ERR=1,FMT=*) ISTOR( 6)
         READ  (UNIT=10,ERR=1,FMT=*) ISTOR( 7)
         READ  (UNIT=10,ERR=1,FMT=*) ISTOR( 8)
         READ  (UNIT=10,ERR=1,FMT=*) ISTOR( 9)
         READ  (UNIT=10,ERR=1,FMT=*) ISTOR(10)
         READ  (UNIT=10,ERR=1,FMT=*) ISTOR(11)
         READ  (UNIT=10,ERR=1,FMT=*) ISTOR(12)
         READ  (UNIT=10,ERR=1,FMT=*) ISTOR(13)
         READ  (UNIT=10,ERR=1,FMT=*) RSTOR( 1)
         READ  (UNIT=10,ERR=1,FMT=*) RSTOR( 2)
         READ  (UNIT=10,ERR=1,FMT=*) RSTOR( 3)
         READ  (UNIT=10,ERR=1,FMT=*) MATRXA
         CLOSE (UNIT=10,ERR=1)
C
C....... read (A) .....................................................
C
         N = 0
         NNZ = 0
         OPEN  (UNIT=11,ERR=2,STATUS='OLD',FILE=MATRXA)
         DO 10 I = 1,LN*LN
            READ (UNIT=11,ERR=2,END=20,FMT=*) IROW(I),ICOL(I),S(I)
            N = MAX(N,IROW(I),ICOL(I))
            NNZ = NNZ + 1
   10    CONTINUE
   20    CLOSE (UNIT=11,ERR=2)
C
         NCOLL = N/NPE
         ICOLL = N + 1
         RMNDR = MOD(N,NPE)
C
      END IF
C
C.... broadcast ISTOR and RSTOR .......................................
C
      CALL MPI_BCAST (ISTOR,13,MPI_INTEGER         ,0,
     &                MPI_COMM_WORLD,INFO)
      CALL MPI_BCAST (RSTOR,3 ,MPI_DOUBLE_PRECISION,0,
     &                MPI_COMM_WORLD,INFO)
C
C.... make sure each process contains the right piece of (A) ..........
C
      DO 60 I = 1,NPE
C
         IF      ( MYPE .EQ. 0 ) THEN
C
C                PE 0 distributes (IROW,ICOL,S) among the processes
```

drvmpi: Fortran source file (3 of 5):

```
C
                NCOLP = NCOLL
                IF ( RMNDR .GT. NPE-I ) NCOLP = NCOLP + 1
                ICOLR = ICOLL - 1
                ICOLL = ICOLR - NCOLP + 1
                DO 40 J = 1,LN
                   DO 30 K = 1,LN
                      A(K,J) = ZERO
   30              CONTINUE
   40           CONTINUE
                DO 50 J = 1,NNZ
                   IF ( ICOLL.LE.IROW(J) .AND. IROW(J).LE.ICOLR )
     &                A(ICOL(J),IROW(J)-ICOLL+1) = S(J)
                   IF ( ICOLL.LE.ICOL(J) .AND. ICOL(J).LE.ICOLR )
     &                A(IROW(J),ICOL(J)-ICOLL+1) = S(J)
   50           CONTINUE
                IF ( I .EQ. NPE ) THEN
                   II = 1
                   NI = NCOLP
                ELSE
                   CALL MPI_SEND (ICOLL,1       ,MPI_INTEGER         ,
     &                            NPE-I,I,MPI_COMM_WORLD,INFO)
                   CALL MPI_SEND (NCOLP,1       ,MPI_INTEGER         ,
     &                            NPE-I,I,MPI_COMM_WORLD,INFO)
                   CALL MPI_SEND (A    ,LN*NCOLP,MPI_DOUBLE_PRECISION,
     &                            NPE-I,I,MPI_COMM_WORLD,INFO)
                END IF
C
         ELSE IF ( MYPE .EQ. NPE-I ) THEN
C
C                PE NPE-I receives data from PE 0
C
                CALL MPI_RECV (II     ,1    ,MPI_INTEGER         ,0,I,
     &                         MPI_COMM_WORLD,STATUS,INFO)
                CALL MPI_RECV (NI     ,1    ,MPI_INTEGER         ,0,I,
     &                         MPI_COMM_WORLD,STATUS,INFO)
                CALL MPI_RECV (A      ,LN*NI,MPI_DOUBLE_PRECISION,0,I,
     &                         MPI_COMM_WORLD,STATUS,INFO)
C
         END IF
C
   60 CONTINUE
C
      ISTOR( 1) = NI
      ISTOR( 2) = NI
      ISTOR( 4) = LEIG
      ISTOR(14) = MPI_COMM_WORLD
      ISTOR(15) = LISTOR
      RSTOR( 4) = LRSTOR
C
C.... check the dimensions of (U) and (V) ..............................
C
      IF ( ISTOR(1) .GT. LN ) STOP '* Error: ISTOR(1) > LN *'
      IF ( ISTOR(5) .GT. NCUV ) STOP '* Error: ISTOR(5) > NCUV *'
C
C.... reverse communication strategy ...................................
C
      LFLAG = 0
C
   70 CONTINUE
C
```

drvmpi: Fortran source file (4 of 5):

```
C     ********************************************************
      CALL DBLZDRV (ISTOR,RSTOR,SIGMA,NNEIG,U,V,LFLAG,NVOPU,EIG,X)
C     ********************************************************
C
      IF      ( LFLAG .LT. 0 ) THEN
C
C..... early  finalization .....................................
C
                WRITE (*,'(/A)') 'execution finished: abnormal exit'
C
      ELSE IF ( LFLAG .EQ. 0 ) THEN
C
C..... normal finalization .....................................
C
                WRITE (*,'(/A)') 'execution finished: standard exit'
C
      ELSE IF ( LFLAG .EQ. 1 ) THEN
C
C..... given (U), compute (V)=(A)*(U) ..........................
C
                DO 80 I = 1,NPE
C
                   ROOT = I - 1
C
C..... each process deals with a pice of (A), (U) and (V) ....
C
                   IF ( NPE .EQ. 1 ) THEN
C
                      JJ = 1
                      NJ = NI
C
                   ELSE
C
C..... process I tells the others what it needs ...........
C
C                     NI rows of (U) and (V): A(II:II+NI-1,:)
C
                      JJ = II
                      NJ = NI
                      CALL MPI_BCAST (JJ,1,MPI_INTEGER,ROOT,
     &                                MPI_COMM_WORLD,INFO)
                      CALL MPI_BCAST (NJ,1,MPI_INTEGER,ROOT,
     &                                MPI_COMM_WORLD,INFO)
C
                   END IF
C
C..... each process computes (T)=A(JJ:JJ+NJ-1,:)*U ...........
C
                   CALL DGEMM ('N','N',NJ,NVOPU,NI,ONE,A(JJ,1),LN,
     &                         U,NI,ZERO,T,NJ)
C
C..... process I gets the sum of (T) and puts it in (V) ......
C
                   CALL MPI_REDUCE (T,V,NJ*NVOPU,MPI_DOUBLE_PRECISION,
     &                              MPI_SUM,ROOT,MPI_COMM_WORLD,INFO)
C
   80           CONTINUE
C
                GO TO 70
C
      ELSE
```

drvmpi: Fortran source file (5 of 5):

```
C
C..... other flags should not be used here ......................
C
              STOP '* Error: LFLAG does not apply in this case *'
C
      END IF
C
      CALL MPI_BARRIER  (MPI_COMM_WORLD,INFO)
      CALL MPI_FINALIZE (INFO)
C
      STOP
    1 STOP '* IO error: file drvsp1.dat *'
    2 STOP '* IO error: file MATRXA *'
C
C**** end of DRVMPI ************************************************
C
      END
```

drvpvm: Fortran source file (1 of 6):

```
      PROGRAM DRVPVM
C     ==============
C
C***********************************************************************
C*                                                                     *
C*    DRVPVM is a driver model for the standard eigenvalue problem     *
C*                                                                     *
C*           (A)*(x)-eig*(x)=(0)                                       *
C*                                                                     *
C*    where (A) is a symmetric matrix of dimension N. DRVPVM is the    *
C*    parallel, PVM based, version of DRVSP1.                          *
C*                                                                     *
C*    Data for DRVPVM are read from 'drvsp1.dat' (see below). The      *
C*    upper triangle of (A) is read from the file MATRXA (in           *
C*    coordinate format).                                              *
C*                                                                     *
C*    Examples: files 'drvsp1.dat' and 'A.dat'                         *
C*    BLAS kernel used: DGEMM                                          *
C*                                                                     *
C***********************************************************************
C
C.... parameters ......................................................
C
      INTEGER          LEIG
      PARAMETER        (LEIG   =     10)
      INTEGER          LISTOR
      PARAMETER        (LISTOR =  10000)
      INTEGER          LN
      PARAMETER        (LN     =    100)
      INTEGER          LRSTOR
      PARAMETER        (LRSTOR =  10000)
      INTEGER          MAXPE
      PARAMETER        (MAXPE  =     32)
      INTEGER          NCUV
      PARAMETER        (NCUV   =      3)
      INTEGER          NNZMAX
      PARAMETER        (NNZMAX =   5000)
      DOUBLE PRECISION ONE
      PARAMETER        (ONE    =  1.0D0)
      DOUBLE PRECISION ZERO
      PARAMETER        (ZERO   =  0.0D0)
C
C.... work variables ..................................................
C
      INTEGER          I,J,K
C
C.... DBLZDRV variables ...............................................
C
      INTEGER          ISTOR(LISTOR),LFLAG,NNEIG,NVOPU
      DOUBLE PRECISION EIG(LEIG,2),RSTOR(LRSTOR),SIGMA,T(LN*NCUV),
     &                 U(LN*NCUV),V(LN*NCUV),X(LN*LEIG)
C
C.... matrix (A) ......................................................
C
      INTEGER          ICOL(NNZMAX),ICOLL,ICOLR,IROW(NNZMAX),
     &                 N,NCOLL,NCOLP,NNZ,RMNDR
      DOUBLE PRECISION A(LN,LN),S(NNZMAX)
      CHARACTER        MATRXA*16
C
C.... PVM .............................................................
C
```

drvpvm: Fortran source file (2 of 6):

```
      INTEGER          II,INFO,JJ,MYINST,MYTID,NI,NJ,
     &                 NPE,ROOT,TID(MAXPE),TIDI
      CHARACTER        GROUP*13
C
      INCLUDE          'fpvm3.h'
C
      EXTERNAL         PVMSUM
C
C=======================================================================
C
      GROUP = 'blzpack.pvm.0'
C
      CALL PVMFMYTID     (MYTID)
      CALL PVMFJOINGROUP (GROUP,MYINST)
C
      IF ( MYINST .EQ. 0 ) THEN
C
         WRITE (*,*) 'How many copies of drvpvm.x?'
         READ  (*,*) NPE
C
C        CALL PVMFCATCHOUT (1)
C
C....... read data ....................................................
C
         OPEN  (UNIT=10,ERR=1,STATUS='OLD',FILE='drvsp1.dat')
         READ  (UNIT=10,ERR=1,FMT=*) ISTOR( 3)
         READ  (UNIT=10,ERR=1,FMT=*) ISTOR( 5)
         READ  (UNIT=10,ERR=1,FMT=*) ISTOR( 6)
         READ  (UNIT=10,ERR=1,FMT=*) ISTOR( 7)
         READ  (UNIT=10,ERR=1,FMT=*) ISTOR( 8)
         READ  (UNIT=10,ERR=1,FMT=*) ISTOR( 9)
         READ  (UNIT=10,ERR=1,FMT=*) ISTOR(10)
         READ  (UNIT=10,ERR=1,FMT=*) ISTOR(11)
         READ  (UNIT=10,ERR=1,FMT=*) ISTOR(12)
         READ  (UNIT=10,ERR=1,FMT=*) ISTOR(13)
         READ  (UNIT=10,ERR=1,FMT=*) RSTOR( 1)
         READ  (UNIT=10,ERR=1,FMT=*) RSTOR( 2)
         READ  (UNIT=10,ERR=1,FMT=*) RSTOR( 3)
         READ  (UNIT=10,ERR=1,FMT=*) MATRXA
         CLOSE (UNIT=10,ERR=1)
C
C....... read (A) .....................................................
C
         N = 0
         NNZ = 0
         OPEN  (UNIT=11,ERR=2,STATUS='OLD',FILE=MATRXA)
         DO 10 I = 1,LN*LN
            READ (UNIT=11,ERR=2,END=20,FMT=*) IROW(I),ICOL(I),S(I)
            N = MAX(N,IROW(I),ICOL(I))
            NNZ = NNZ + 1
   10    CONTINUE
   20    CLOSE (UNIT=11,ERR=2)
C
         NCOLL = N/NPE
         ICOLL = N + 1
         RMNDR = MOD(N,NPE)
C
C....... initiate NPE copies of drvpvm.x ..............................
C
         IF ( NPE .LE. 0 ) STOP '* Error: NPE < 0 *'
         IF ( NPE .GT. N ) STOP '* Error: NPE > N *'
```

drvpvm: Fortran source file (3 of 6):

```
C
         TID(1) = MYTID
C
         IF ( NPE .GT. 1 ) THEN
            CALL PVMFSPAWN ('drvpvm.x',0,'*',NPE-1,TID(2),INFO)
            IF ( INFO .NE. NPE-1 ) THEN
               CALL PVMFEXIT (INFO)
               STOP '* Error: PVMFSPAWN failed to spawn drvpvm.x *'
            END IF
         END IF
C
C....... send N, NPE and TID to other processes ......................
C
         DO 30 I = 1,NPE-1
            CALL PVMFINITSEND (PVMDEFAULT,INFO)
            CALL PVMFPACK     (INTEGER4,NPE,1  ,1,INFO)
            CALL PVMFPACK     (INTEGER4,TID,NPE,1,INFO)
            CALL PVMFSEND     (TID(I+1),1,INFO)
   30    CONTINUE
C
      ELSE
C
C....... receive N, NPE and TID from root ............................
C
         CALL PVMFRECV   (-1,1,INFO)
         CALL PVMFUNPACK (INTEGER4,NPE,1  ,1,INFO)
         CALL PVMFUNPACK (INTEGER4,TID,NPE,1,INFO)
C
      END IF
C
      CALL PVMFFREEZEGROUP (GROUP,NPE,INFO)
C
C.... make sure each process contains the right piece of (A) ...........
C
      DO 70 I = 1,NPE
C
         IF      ( MYINST .EQ. 0 ) THEN
C
C                PE 0 distributes (IROW,ICOL,S) among the processes
C
                 NCOLP = NCOLL
                 IF ( RMNDR .GT. NPE-I ) NCOLP = NCOLP + 1
                 ICOLR = ICOLL - 1
                 ICOLL = ICOLR - NCOLP + 1
                 DO 50 J = 1,LN
                    DO 40 K = 1,LN
                       A(K,J) = ZERO
   40               CONTINUE
   50            CONTINUE
                 DO 60 J = 1,NNZ
                    IF ( ICOLL.LE.IROW(J) .AND. IROW(J).LE.ICOLR )
     &                 A(ICOL(J),IROW(J)-ICOLL+1) = S(J)
                    IF ( ICOLL.LE.ICOL(J) .AND. ICOL(J).LE.ICOLR )
     &                 A(IROW(J),ICOL(J)-ICOLL+1) = S(J)
   60            CONTINUE
                 IF ( I .EQ. NPE ) THEN
                    II = 1
                    NI = NCOLP
                 ELSE
                    CALL PVMFGETTID   (GROUP,NPE-I,TIDI)
                    CALL PVMFINITSEND (PVMDEFAULT,INFO)
```

drvpvm: Fortran source file (4 of 6):

```
                    CALL PVMFPACK     (INTEGER4,ICOLL ,1       ,1,INFO)
                    CALL PVMFPACK     (INTEGER4,NCOLP ,1       ,1,INFO)
                    CALL PVMFPACK     (INTEGER4,ISTOR ,13      ,1,INFO)
                    CALL PVMFPACK     (REAL8   ,RSTOR ,3       ,1,INFO)
                    CALL PVMFPACK     (REAL8   ,A     ,LN*NCOLP,1,INFO)
                    CALL PVMFSEND     (TIDI,I,INFO)
                 END IF
C
         ELSE IF ( MYINST .EQ. NPE-I ) THEN
C
C                PE NPE-I receives data from PE 0
C
                 CALL PVMFRECV   (-1,I,INFO)
                 CALL PVMFUNPACK (INTEGER4,II   ,1    ,1,INFO)
                 CALL PVMFUNPACK (INTEGER4,NI   ,1    ,1,INFO)
                 CALL PVMFUNPACK (INTEGER4,ISTOR,13   ,1,INFO)
                 CALL PVMFUNPACK (REAL8   ,RSTOR,3    ,1,INFO)
                 CALL PVMFUNPACK (REAL8   ,A    ,LN*NI,1,INFO)
C
         END IF
C
   70 CONTINUE
C
      ISTOR( 1) = NI
      ISTOR( 2) = NI
      ISTOR( 4) = LEIG
      ISTOR(14) = 0
      ISTOR(15) = LISTOR
      RSTOR( 4) = LRSTOR
C
C.... check the dimensions of (U) and (V) ............................
C
      IF ( ISTOR(1) .GT. LN ) STOP '* Error: ISTOR(1) > LN *'
      IF ( ISTOR(5) .GT. NCUV ) STOP '* Error: ISTOR(5) > NCUV *'
C
      CALL PVMFBARRIER (GROUP,NPE,INFO)
C
C.... reverse communication strategy .................................
C
      LFLAG = 0
C
   80 CONTINUE
C
C     ********************************************************
      CALL DBLZDRV (ISTOR,RSTOR,SIGMA,NNEIG,U,V,LFLAG,NVOPU,EIG,X)
C     ********************************************************
C
      IF      ( LFLAG .LT. 0 ) THEN
C
C............ early  finalization ....................................
C
              WRITE (*,'(/A)') 'execution finished: abnormal exit'
C
      ELSE IF ( LFLAG .EQ. 0 ) THEN
C
C............ normal finalization ....................................
C
              WRITE (*,'(/A)') 'execution finished: standard exit'
C
      ELSE IF ( LFLAG .EQ. 1 ) THEN
C
```

drvpvm: Fortran source file (5 of 6):

```
C..... given (U), compute (V)=(A)*(U) ........................
C
              DO 90 I = 1,NPE
C
C.............. each process deals with a pice of (A), (U) and (V) ....
C
                 IF      ( NPE .EQ. 1 ) THEN
C
                         JJ = 1
                         NJ = NI
C
                 ELSE IF ( TID(I) .EQ. MYTID ) THEN
C
C...................... process I tells the others what it needs ......
C
C                        NI rows of (U) and (V): A(II:II+NI-1,:)
C
                         JJ = II
                         NJ = NI
                         CALL PVMFINITSEND (PVMDEFAULT,INFO)
                         CALL PVMFPACK     (INTEGER4,JJ,1,1,INFO)
                         CALL PVMFPACK     (INTEGER4,NJ,1,1,INFO)
                         CALL PVMFBCAST    (GROUP,I,INFO)
C
                 ELSE
C
C...................... request from process I .......................
C
                         CALL PVMFRECV     (TID(I),I,INFO)
                         CALL PVMFUNPACK   (INTEGER4,JJ,1,1,INFO)
                         CALL PVMFUNPACK   (INTEGER4,NJ,1,1,INFO)
C
                 END IF
C
C.............. each process computes (T)=A(JJ:JJ+NJ-1,:)*U ...........
C
                 CALL DGEMM ('N','N',NJ,NVOPU,NI,ONE,A(JJ,1),LN,
     &                       U,NI,ZERO,T,NJ)
C
C.............. process I gets the sum of (T) and puts it in (V) ......
C
C
                 IF ( NPE .GT. 1 ) THEN
                    CALL PVMFBARRIER (GROUP,NPE,INFO)
                    CALL PVMFGETINST (GROUP,TID(I),ROOT)
                    CALL PVMFREDUCE  (PVMSUM,T,NJ*NVOPU,REAL8,1,
     &                                GROUP,ROOT,INFO)
                 END IF
C
                 IF ( TID(I) .EQ. MYTID ) CALL DCOPY (NJ*NVOPU,T,1,V,1)
C
   90         CONTINUE

              GO TO 80
C
      ELSE
C
C..... other flags should not be used here ......................
C
              STOP '* Error: LFLAG does not apply in this case *'
C
```

drvpvm: Fortran source file (6 of 6):

```
      END IF
C
      CALL PVMFBARRIER (GROUP,NPE,INFO)
      CALL PVMFLVGROUP (GROUP,INFO)
      CALL PVMFEXIT    (INFO)
C
      STOP
    1 STOP '* IO error: file drvsp1.dat *'
    2 STOP '* IO error: file MATRXA *'
C
C**** end of DRVPVM ***************************************************
C
      END
```

References

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. W. Demmel, J. J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. C. Sorensen. *LAPACK User's Guide*. SIAM, Philadelphia, USA. Third edition, 1999.

[2] J. W. Demmel. *Applied Numerical Linear Algebra*. SIAM, Philadelphia, USA, 1997.

[3] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, USA, third edition, 1996.

[4] R. G. Grimes, J. G. Lewis, and H. D. Simon. A Shifted Block Lanczos Algorithm for Solving Sparse Symmetric Eigenvalue Problems. *SIAM J. Matrix Anal. Appl.*, 15:228–272, 1994.

[5] O. A. Marques. BLZPACK: Description and User's Guide. Technical Report TR/PA/95/30, CERFACS, Toulouse, France, 1995.

[6] B. Nour-Omid. The Lanczos Algorithm for Solution of Large Generalized Eigenproblem. In T. J. R. Hughes, editor, *The Finite Element Method*, pages 582–630, Englewood Cliffs, USA, 1987. Prentice Hall International Editions.

[7] B. N. Parlett. *The Symmetric Eigenvalue Problem*. SIAM (Classics in Applied Mathematics), Philadelphia, USA, 1998.