F08JXFP (PZSTEIN)

NAG Parallel Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check for implementation-dependent details. You are advised to enclose any calls to NAG Parallel Library routines between calls to Z01AAFP and Z01ABFP.

1 Description

F08JXFP (PZSTEIN) computes the eigenvectors of an n by n real symmetric tridiagonal matrix T using inverse iteration. This routine is identical to F08JKFP (PDSTEIN) except that the eigenvectors are stored in a complex array instead in a real array. F08JXFP (PZSTEIN) is mainly intended to be used in the solution of the dense complex Hermitian eigenvalue problem. Note that the eigenvectors of a real symmetric tridiagonal matrix are real.

The computed eigenvectors correspond to m user specified eigenvalues. The computed eigenvectors are are in a (complex) submatrix Z_s of a larger m_Z by n_Z (complex) matrix Z, i.e.,

$$Z_s \equiv Z(i_Z : i_Z + n - 1, j_Z : j_Z + m - 1).$$

Note: if $i_Z = j_Z = 1$ and $m = m_Z$, $n = n_Z$, then $Z_s \equiv Z$.

The columns of Z_s give the computed eigenvectors which correspond to the eigenvalues in the ascending order.

This routine is designed to be used in particular after specified eigenvalues have been computed by F08JJFP (PDSTEBZ) with ORDER = 'B', but may also be used when the eigenvalues have been computed by some other means.

If the matrix T has been formed by reduction of a full complex Hermitian matrix A_s to tridiagonal form then eigenvectors of T may be transformed to eigenvectors of A_s by a call to F08FUFP (PZUNMTR).

F08JXFP (PZSTEIN) does not orthogonalize the computed eigenvectors that are resident on different logical processors. The extent of orthogonalization is controlled by the input parameter LWORK which defines the size of the allocated workspace. Eigenvectors that are to be orthogonalized are computed by the same logical processor. If insufficient workspace is allocated, the expected orthogonalization may not be done. If the computed eigenvectors are not orthogonal to the user desired accuracy then the workspace specified by LWORK should be increased in the next run.

2 Specification

```
 SUBROUTINE F08JXFP(N, D, E, M, W, IBLOCK, ISPLIT, ORFAC, Z, IZ,
1                   JZ, IDESCZ, WORK, LWORK, IWORK, LIWORK, JFAIL,
2                   ICLUSTR, GAP, INFO)
 ENTRY      PZSTEIN(N, D, E, M, W, IBLOCK, ISPLIT, ORFAC, Z, IZ,
1                   JZ, IDESCZ, WORK, LWORK, IWORK, LIWORK, JFAIL,
2                   ICLUSTR, GAP, INFO)
 COMPLEX*16         Z(*)
 DOUBLE PRECISION   D(*), E(*), W(*), ORFAC(*), WORK(*), GAP(*)
 INTEGER            N, M, IBLOCK(*), ISPLIT(*), IZ, JZ, IDESCZ(*),
1                   LWORK, IWORK(*), LIWORK, JFAIL(*), ICLUSTR(*),
2                   INFO
```

The ENTRY statement enables the routine to be called by its ScaLAPACK name.

3 Usage

3.1 Definitions

The following definitions are used in describing the data distribution within this document:

m_p	–	the number of rows in the Library Grid.
n_p	–	the number of columns in the Library Grid.
p	–	$m_p \times n_p$, the total number of processors in the Library Grid.
p_r	–	the row grid coordinate of the calling processor.
p_c	–	the column grid coordinate of the calling processor.
m_X	–	the number of rows of a matrix X.
n_X	–	the number of columns of a matrix X.
M_b^X	–	the blocking factor for the distribution of the rows of a matrix X.
N_b^X	–	the blocking factor for the distribution of the columns of a matrix X.
s_r^X	–	the row coordinate of the processor that possesses the first row of a distributed matrix X.
s_c^X	–	the column coordinate of the processor that possesses the first column of a distributed matrix X.
numroc($\hat{\ell}$,L_b^X,p,s^X,ℓ_p)	–	a function which gives the **num**ber of **r**ows **o**r **c**olumns of a distributed matrix X owned by the processor with the row or column coordinate p (p_r or p_c), where $\hat{\ell}$ is the total number of rows or columns of the matrix, L_b^X is the blocking factor used (M_b^X or N_b^X), s^X is the row or column coordinate (s_r^X or s_c^X) of the processor that possesses the first row or column of the distributed matrix and ℓ_p is either m_p or n_p. The Library provides the function Z01CAFP (NUMROC) for the evaluation of this function.
$\lfloor x \rfloor$	–	the floor function of x which gives the largest integer which is not greater than x.
$\lceil x \rceil$	–	the ceiling function of x, which gives the smallest integer which is not less than x.

3.2 Global and Local Arguments

The following global **input** arguments must have the same value on entry to the routine on each processor and the global **output** arguments will have the same value on exit from the routine on each processor:

Global input arguments:	N, D, E, M, IBLOCK, ISPLIT, ORFAC, IZ, JZ, W, the array elements IDESCZ(1) and IDESCZ(3:8)
Global output arguments:	JFAIL, ICLUSTR, GAP, W, IWORK, INFO

The remaining arguments are local.

3.3 Distribution Strategy

Each logical processor must contain the vectors d and e which are the diagonal and off-diagonal entries of the tridiagonal matrix T, respectively. Furthermore, the array W containing the user specified eigenvalues (corresponding to the required eigenvectors) must be supplied to each logical processor.

On exit, the matrix Z is partitioned into M_b by N_b rectangular blocks and stored in an array Z in a two-dimensional cyclic block distribution.

3.4 Related Routines

The Library provides many support routines for the generation/distribution and input/output of data. The following routines may be used in conjunction with F08JXFP (PZSTEIN):

Real matrix output:	X04BFFP
Complex matrix generation:	F01ZVFP
Real vector gather:	F01ZPFP
Complex matrix gather:	F01WGFP

4 Arguments

1: N — INTEGER *Global Input*

On entry: n, the order of the symmetric tridiagonal matrix T.

Constraint: N ≥ 0.

2: D($*$) — DOUBLE PRECISION array *Global Input*

Note: the dimension of the array D must be at least max(1,N).

On entry: d, the vector containing the diagonal elements of the symmetric tridiagonal matrix T.

3: E($*$) — DOUBLE PRECISION array *Global Input*

Note: the dimension of the array E must be at least max(1,N−1).

On entry: e, the vector containing the off-diagonal elements of the symmetric tridiagonal matrix T.

4: M — INTEGER *Global Input*

On entry: m, the requested number of eigenvectors to be computed.

Constraint: $0 \leq$ M $\leq$ N.

5: W($*$) — DOUBLE PRECISION array *Global Input/Global Output*

Note: the dimension of the array W must be at least max(1,N).

On entry: the first m elements of W must contain all the eigenvalues for which eigenvectors are to be computed. The eigenvalues should be grouped by split-off block and ordered from smallest to largest within the block.

Usually, the output array W from routine F08JJFP (PDSTEBZ) with ORDER='B' is expected here. To obtain computed eigenvectors with high orthogonality, the eigenvalues should be computed to the highest possible accuracy. This can be achieved by setting the parameter ABSTOL to the underflow threshold in the prior call to the routine F08JJFP (PDSTEBZ). The underflow threshold is given by the function X02AKF.

On exit: the first m elements contain the input eigenvalues in ascending order.

6: IBLOCK($*$) — INTEGER array *Global Input*

Note: the dimension of the array IBLOCK must be at least max(1,N).

On entry: IBLOCK(i) contains the block number of the eigenvalues stored in W(i), for $i = 1, \ldots, m$.

Usually, the output array IBLOCK from routine F08JJFP (PDSTEBZ) is expected here.

7: ISPLIT($*$) — INTEGER array *Global Input*

Note: the dimension of the array ISPLIT must be at least max(1,N).

On entry: the leading NSPLIT (as defined in F08JJFP (PDSTEBZ)) elements of ISPLIT contain the points at which the matrix T splits up into sub-matrices (matrix blocks) as follows. The first sub-matrix consists of rows/columns 1 to ISPLIT(1), the second sub-matrix consists of rows/columns ISPLIT(1) + 1 to ISPLIT(2), and the last sub-matrix consists of rows/columns ISPLIT(NSPLIT−1) + 1 to ISPLIT(NSPLIT) (= n). NSPLIT is the number of diagonal blocks which constitute the tridiagonal matrix T.

Usually, the output array ISPLIT from routine F08JJFP (PDSTEBZ) is expected here.

8: ORFAC — INTEGER *Global Input*

On entry: specifies which eigenvectors should be orthogonalized. Eigenvectors that correspond to eigenvalues which are within ORFAC$*\|T\|$ of each other are to be orthogonalized. However, if the workspace is insufficient, this tolerance may be decreased until all eigenvectors to be orthogonalized can be stored on one logical processor (see also LWORK). No orthogonalization will be done if ORFAC = 0. A default value of 10^{-3} is used if ORFAC is negative.

9: Z(∗) — COMPLEX*16 array *Local Output*

Note: array Z is formally defined as a vector. However, you may find it more convenient to consider Z as a two-dimensional array of dimension (IDESCZ(9),γ), where $\gamma \geq m/n_p + N_b$.

On exit: the locally held parts of the array Z pertaining to the eigenvector matrix Z_s whose columns give the eigenvectors. The computed eigenvectors corresponds to the user specified eigenvalues in the ascending order. The array Z conforms to the two-dimensional cyclic block distribution.

Any eigenvector which fails to converge is set to the final iterate after the maximum number of iterations allowed by this routine.

10: IZ — INTEGER *Global Input*

On entry: i_Z, the row index of matrix Z that identifies the first row of the submatrix Z_s.

Constraints:

$1 \leq$ IZ $\leq$ IDESCZ(3)+ 1 − N.

11: JZ — INTEGER *Global Input*

On entry: j_Z, the column index of matrix Z that identifies the first column of the submatrix Z_s.

Constraints:

$1 \leq$ JZ $\leq$ IDESCZ(4) + 1− M.

12: IDESCZ(∗) — INTEGER array *Local Input*

Note: the dimension of the array IDESCZ must be at least 9.

Distribution: the array elements IDESCZ(1) and IDESCZ(3),. . .,IDESCZ(8) must be global to the processor grid and the elements IDESCZ(2) andIDESCZ(9) are local to each processor.

On entry: the description array for the matrix Z. This array must contain details of the distribution of the matrix Z and the logical processor grid.

IDESCZ(1), the descriptor type. For this routine, which uses a cyclic two-dimensional block distribution, IDESCZ(1) = 1;
IDESCZ(2), the Library context, usually returned by a call to the Library Grid initialisation routine Z01AAFP;
IDESCZ(3), the number of rows, m_Z, of the matrix Z;
IDESCZ(4), the number of columns, n_Z, of the matrix Z;
IDESCZ(5), the blocking factor, M_b^Z, used to distribute the rows of the matrix Z;
IDESCZ(6), the blocking factor, N_b^Z, used to distribute the columns of the matrix Z;
IDESCZ(7), the processor row index, s_r^Z, over which the first row of the matrix Z is distributed;
IDESCZ(8), the processor column index, s_c^Z, over which the first column of the matrix Z is distributed;
IDESCZ(9), the leading dimension of the conceptual two-dimensional array Z.

Constraints:

IDESCZ(5) $\geq$ 1; IDESCZ(6) $\geq$ 1;
0 $\leq$ IDESCZ(7) $\leq m_p-1$; 0 $\leq$ IDESCZ(8) $\leq n_p-1$;
IDESCZ(9) $\geq$ max(1, numroc(IDESCZ(3),IDESCZ(5), p_r, IDESCZ(7), m_p)).

13: WORK(∗) — DOUBLE PRECISION array *Local Workspace/Global Output*

Note: the dimension of WORK must be at least max(1,LWORK). The minimum value of LWORK required to successfully call this routine can be obtained by setting LWORK = −1. The required size is returned in array element WORK(1).

On exit: WORK(1) contains the minimum dimension of the array WORK required to successfully complete the task.

14: LWORK — INTEGER *Local Input*

On entry: either -1 (see WORK) or the dimension of the array WORK required to successfully complete the task. If LWORK is set to -1 on entry this routine simply performs some initial error checking and then, if these checks are successful, calculates the minimum size of LWORK required.

Constraints:

either LWORK $= -1$,
or LWORK $\geq \max(5n, m_0\ n_0) + \lceil m/p \rceil n$.

15: IWORK($*$) — INTEGER array *Local Workspace/Global Output*

Note: the minimum value of LIWORK required to successfully call this routine can be obtained by setting LIWORK $= -1$. The required size is returned in array element IWORK(1).

On exit: IWORK(1) contains the minimum dimension of array IWORK required to successfully complete the task.

16: LIWORK — INTEGER *Local Input*

On entry: either -1 (see IWORK) or the dimension of the array WORK required to successfully complete the task. If LIWORK is set to -1 on entry this routine simply performs some initial error checking and then, if these checks are successful, calculates the minimum size of LIWORK required.

Constraints:

either LIWORK $= -1$,
or LIWORK $\geq 3n+p+1$.

17: JFAIL($*$) — INTEGER array *Global Output*

Note: the dimension of the array JFAIL must be at least max(1,M).

On exit: on successful exit, the first M elements of JFAIL are zero.

However, if mod(INFO,M+1) > 0 then one or more eigenvectors failed to converge after the maximum number of iterations allowed by this routine. For $i = 1, \ldots,$mod(INFO,M+1), the eigenvector corresponding to the eigenvalue W(JFAIL(i)) failed to converge where W refers to the array of eigenvalues on exit.

18: ICLUSTR($*$) — INTEGER array *Global Output*

Note: the dimension of the array ICLUSTR must be at least max(1,2p).

On exit: on successful exit, ICLUSTR does not contain any useful information.

However, if mod(INFO,M+1) > 0 then this array contains indices of eigenvectors corresponding to a cluster of eigenvalues that could not be orthogonalized due to insufficient workspace. Specifically, the eigenvectors corresponding to clusters of eigenvalues indexed from ICLUSTR($2i-1$) to ICLUSTR($2i$), $i = 1, \ldots,$INFO/(M+1) could not be orthogonalized due to lack of workspace. Hence the eigenvectors corresponding to these clusters may not be orthogonal. ICLUSTR is zero terminated, that is ICLUSTR($2k$) $\neq 0$ and ICLUSTR($2k+1$) $= 0$, if and only if k is the number of clusters.

19: GAP($*$) — DOUBLE PRECISION array *Global Output*

Note: the dimension of the array GAP must be at least max(1,p).

On exit: on successful exit (INFO=0), GAP does not contain any useful information.

However, if INFO/(M+1) > 0, this array contains the gap between eigenvalues whose eigenvectors could not be orthogonalized. The INFO/M output values in this array correspond to the INFO/(M+1) clusters indicated by the array ICLUSTR.

20: INFO — INTEGER *Global Output*

The NAG Parallel Library provides a mechanism, via the routine Z02EAFP, to reduce the amount of parameter validation performed by this routine. For a full description refer to the Z02 Chapter Introduction.

On exit: INFO = 0 (or −9999 if reduced error checking is enabled) unless the routine detects an error (see Section 5).

5 Errors and Warnings

If INFO < 0 an explanatory message is output and control returned to the calling program.

INFO = $-i$

On entry, one of the arguments was invalid:

if the kth argument is a scalar INFO = $-k$;
if the kth argument is an array and its jth element is invalid, INFO = $-(100\times k + j)$.

This error occurred either because a global argument did not have the same value on all logical processors, or because its value on one or more processors was incorrect.

mod(INFO,M+1) > 0

One or more eigenvectors failed to converge after the maximum number of iterations allowed by this routine. For $i = 1,\ldots,$mod(INFO,M+1), the eigenvector corresponding to the eigenvalue W(JFAIL(i)) failed to converge where W refers to the array of eigenvalues on exit.

INFO/(M+1) > 0

The array ICLUSTR contains indices of eigenvectors corresponding to a cluster of eigenvalues that could not be orthogonalized due to insufficient workspace. Specifically, the eigenvectors corresponding to clusters of eigenvalues indexed from ICLUSTR($2i-1$) to ICLUSTR($2i$), $i = 1,\ldots,$INFO/(M+1) could not be orthogonalized due to lack of workspace. Hence the eigenvectors corresponding to these clusters may not be orthogonal. ICLUSTR is a zero terminated, that is ICLUSTR($2k$) $\neq 0$ and ICLUSTR($2k+1$) = 0, if and only if k is the number of clusters.

The array GAP contains the gap between eigenvalues whose eigenvectors could not be orthogonalized. The INFO/M output values in this array correspond to the INFO/(M+1) clusters indicated by the array ICLUSTR.

6 Further Comments

Often, F08JXFP (PZSTEIN) is the penultimate routine to be called in the solution of the Hermitian eigenvalue problem associated with a dense Hermitian matrix. The vectors d and e are obtained via F08FSFP (PZHETRD) which gives the tridiagonal form T of the dense Hermitian matrix A_s. The vectors d and e are then gathered to each and every processor using F01ZPFP. The eigenvalues of T (equivalently, the eigenvalues of A_s), which are required by this eigenvector routine F08JXFP (PZSTEIN), are usually computed via F08JJFP (PDSTEBZ). Once the eigenvectors of T are computed, they can be transformed to eigenvectors of A using F08FUFP (PZUNMTR).

6.1 Algorithmic Detail

This eigenvector routine executes the ScaLAPACK routine DSTEIN2 which is a modified version of the LAPACK routine DSTEIN. See Blackford *et al.* [4].

6.2 Parallelism Detail

The algorithm is embarrassingly parallel and the inter-process communication is minimal. However, in the presence of large clusters of eigenvalues for which eigenvectors have been requested, the performance (scalability) of this algorithm could degrade as the sizes of the clusters increase.

6.3 Accuracy

Each computed eigenvector z_i is the exact eigenvector of a nearby matrix $A_s + E_i$, such that $\| E_i \| = O(\epsilon)\| A_s \|$, where ϵ is the ***machine precision***. Hence the residual is small:

$$\| A_s z_i - \lambda_i z_i \| = O(\epsilon)\| A_s \|.$$

However a set of eigenvectors computed by this routine may not be orthogonal to so high a degree of accuracy as those computed by QR/QL algorithms.

7 References

[1] Golub G H and van Loan C F (1996) *Matrix Computations* Johns Hopkins University Press (3rd Edition), Baltimore

[2] Kahan W (1966) Accurate eigenvalues of a symmetric tridiagonal matrix *Report CS41* Stanford University

[3] Jessup E and Ipsen I C F (1992) Improving the accuracy of inverse iteration *SIAM J. Sci. Statist. Comput.* **13** 550–572

[4] Blackford L S, Choi J, Cleary A, D'Azevedo E, Demmel J, Dhillon I, Dongarra J, Hammarling S, Henry G, Petitet A, Stanley K, Walker D and Whaley R C (1997) ScaLAPACK Users' Guide *SIAM* 3600 University City Science Center, Philadelpia, PA 19104-2688, USA. URL: http://www.netlib.org/scalapack/slug/scalapack_slug.html

8 Example

See Section 8 of the document for F08FUFP.