NAG C Library Function Document

nag_real_symm_banded_sparse_eigensystem_sol (f12fgc)

1 Purpose

nag_real_symm_banded_sparse_eigensystem_sol (f12fgc) is the main solver function in a suite of functions consisting of nag_real_symm_banded_sparse_eigensystem_sol (f12fgc), nag_real_symm_sparse_eigensystem_option (f12fdc) and nag_real_symm_banded_sparse_eigensystem_init (f12ffc), that must be called following an initial call to nag_real_symm_banded_sparse_eigensystem_init (f12ffc) and following any calls to nag_real_symm_sparse_eigensystem_option (f12fdc).

nag_real_symm_banded_sparse_eigensystem_sol (f12fgc) returns approximations to selected eigenvalues, and (optionally) the corresponding eigenvectors, of a standard or generalized eigenvalue problem defined by real banded symmetric matrices. The banded matrix must be stored using the LAPACK storage format for real banded nonsymmetric matrices.

2 Specification

```
#include <nag.h>
#include <nagf12.h>
```

```
void nag_real_symm_banded_sparse_eigensystem_sol (Integer kl, Integer ku,
     const double ab[], const double mb[], double sigma, Integer *nconv,
     double d[], double z[], double resid[], double v[], double comm[],
     Integer icomm[], NagError *fail)
```

3 Description

The suite of functions is designed to calculate some of the eigenvalues, λ, (and optionally the corresponding eigenvectors, x) of a standard eigenvalue problem $Ax = \lambda x$, or of a generalized eigenvalue problem $Ax = \lambda Bx$ of order n, where n is large and the coefficient matrices A and B are banded, real and symmetric.

Following a call to the initialization function nag_real_symm_banded_sparse_eigensystem_init (f12ffc), nag_real_symm_banded_sparse_eigensystem_sol (f12fgc) returns the converged approximations to eigenvalues and (optionally) the corresponding approximate eigenvectors and/or an orthonormal basis for the associated approximate invariant subspace. The eigenvalues (and eigenvectors) are selected from those of a standard or generalized eigenvalue problem defined by real banded symmetric matrices. There is negligible additional computational cost to obtain eigenvectors; an orthonormal basis is always computed, but there is an additional storage cost if both are requested.

The banded matrices A and B must be stored using the LAPACK storage format for banded nonsymmetric matrices; please refer to Section 3.3.2 in the f07 Chapter Introduction for details on this storage format.

nag_real_symm_banded_sparse_eigensystem_sol (f12fgc) is based on the banded driver functions **dsbdr1** to **dsbdr6** from the ARPACK package, which uses the Implicitly Restarted Lanczos iteration method. The method is described in Lehoucq and Sorensen (1996) and Lehoucq (2001) while its use within the ARPACK software is described in great detail in Lehoucq *et al.* (1998). This suite of functions offers the same functionality as the ARPACK banded driver software for real symmetric problems, but the interface design is quite different in order to make the option setting clearer to you and to combine the different drivers into a general purpose function.

nag_real_symm_banded_sparse_eigensystem_sol (f12fgc), is a general purpose forward communication function that must be called following initialization by nag_real_symm_banded_sparse_eigensystem_init (f12ffc). nag_real_symm_banded_sparse_eigensystem_sol (f12fgc) uses options, set either by default or explicitly by calling nag_real_symm_sparse_eigensystem_option (f12fdc), to return the converged approximations to selected eigenvalues and (optionally):

– the corresponding approximate eigenvectors;

– an orthonormal basis for the associated approximate invariant subspace;

– both.

4 References

Lehoucq R B (2001) Implicitly Restarted Arnoldi Methods and Subspace Iteration *SIAM Journal on Matrix Analysis and Applications* **23** 551–562

Lehoucq R B and Scott J A (1996) An evaluation of software for computing eigenvalues of sparse nonsymmetric matrices *Preprint MCS-P547-1195* Argonne National Laboratory

Lehoucq R B and Sorensen D C (1996) Deflation Techniques for an Implicitly Restarted Arnoldi Iteration *SIAM Journal on Matrix Analysis and Applications* **17** 789–821

Lehoucq R B, Sorensen D C and Yang C (1998) *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods* SIAM, Philidelphia

5 Arguments

1: **kl** – Integer *Input*

On entry: the number of subdiagonals of the matrices A and B.

Constraint: $\mathbf{kl} \geq 0$.

2: **ku** – Integer *Input*

On entry: the number of superdiagonals of the matrices A and B. Since A and B are symmetric, the normal case is $\mathbf{ku} = \mathbf{kl}$.

Constraint: $\mathbf{ku} \geq 0$.

3: **ab**$[dim]$ – const double *Input*

Note: the dimension, *dim*, of the array **ab** must be at least $\max(1, \mathbf{n} \times (2\mathbf{kl} + \mathbf{ku} + 1))$ (see nag_real_symm_banded_sparse_eigensystem_init (f12ffc)).

On entry: must contain the matrix A in LAPACK column-ordered banded storage format for non-symmetric matrices (see Section 3.3.2 in the f07 Chapter Introduction).

4: **mb**$[dim]$ – const double *Input*

Note: the dimension, *dim*, of the array **mb** must be at least $\max(1, \mathbf{n} \times (2\mathbf{kl} + \mathbf{ku} + 1))$ (see nag_real_symm_banded_sparse_eigensystem_init (f12ffc)).

On entry: must contain the matrix B in LAPACK column-ordered banded storage format for non-symmetric matrices (see Section 3.3.2 in the f07 Chapter Introduction).

5: **sigma** – double *Input*

On entry: if one of the **Shifted** modes (see nag_real_symm_sparse_eigensystem_option (f12fdc)) has been selected then **sigma** contains the real shift used; otherwise **sigma** is not referenced.

6: **nconv** – Integer * *Output*

On exit: the number of converged eigenvalues.

7: **d**$[dim]$ – double *Output*

Note: the dimension, *dim*, of the array **d** must be at least **nev**.

On exit: the first **nconv** locations of the array **d** contain the converged approximate eigenvalues.

8: **z**[*dim*] – double *Output*

Note: the dimension, *dim*, of the array **z** must be at least **nev** $\times$ **n** (see nag_real_symm_banded_sparse_eigensystem_init (f12ffc)).

On exit: if the default option **Vectors** = Ritz (see nag_real_symm_sparse_eigensystem_option (f12fdc)) has been selected then **z** contains the final set of eigenvectors corresponding to the eigenvalues held in **d**. The real eigenvector associated with eigenvalue i, for $i = 0, 1, \ldots, \mathbf{nconv} - 1$, is stored at locations $\mathbf{z}[i \times n + j]$, for $j = 0, 1, \ldots, n - 1$.

9: **resid**[*dim*] – double *Input/Output*

Note: the dimension, *dim*, of the array **resid** must be at least **n** (see nag_real_symm_banded_sparse_eigensystem_init (f12ffc)).

On entry: need not be set unless the option **Initial Residual** has been set in a prior call to nag_real_symm_sparse_eigensystem_option (f12fdc) in which case **resid** should contain an initial residual vector.

On exit: contains the final residual vector.

10: **v**[*dim*] – double *Output*

Note: the dimension, *dim*, of the array **v** must be at least **n** $\times$ **ncv** (see nag_real_symm_banded_sparse_eigensystem_init (f12ffc)).

On exit: if the option **Vectors** has been set to Schur or Ritz and **z** does not equal **v** then the first **nconv** sections of **v**, of length n, will contain approximate Schur vectors that span the desired invariant subspace. The ith Schur vector is stored in locations $\mathbf{v}[i \times n + j]$, for $i = 0, 1, \ldots, \mathbf{nconv} - 1$ and $j = 0, 1, \ldots, n - 1$.

11: **comm**[*dim*] – double *Communication Array*

Note: the dimension, *dim*, of the array **comm** must be at least $\max(1, \mathbf{lcomm})$ (see nag_real_symm_banded_sparse_eigensystem_init (f12ffc)).

On initial entry: must remain unchanged from the prior call to nag_real_symm_banded_sparse_eigensystem_init (f12ffc) and nag_real_symm_sparse_eigensystem_option (f12fdc).

On exit: contains no useful information.

12: **icomm**[*dim*] – Integer *Communication Array*

Note: the dimension, *dim*, of the array **icomm** must be at least $\max(1, \mathbf{licomm})$ (see nag_real_symm_banded_sparse_eigensystem_init (f12ffc)).

On initial entry: must remain unchanged from the prior call to nag_real_symm_sparse_eigensystem_iter (f12fbc) and nag_real_symm_sparse_eigensystem_option (f12fdc).

On exit: contains no useful information.

13: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.6 of the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Error: unable to allocate requested internal workspace.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_BOTH_ENDS_1

Eigenvalues from both ends of the spectrum were requested, but the number of eigenvalues (**nev**) requested is one.

NE_INT

The maximum number of iterations ≤ 0, the option **Iteration Limit** has been set to $\langle value \rangle$.

On entry, **kl** $= \langle value \rangle$.
Constraint: **kl** ≥ 0.

On entry, **ku** $= \langle value \rangle$.
Constraint: **ku** ≥ 0.

NE_INT_2

The maximum number of iterations has been reached. The maximum number of iterations $= \langle value \rangle$. The number of converged eigenvalues $= \langle value \rangle$.

NE_INTERNAL_EIGVAL_FAIL

Error in internal call to compute eigenvalues and corresponding error bounds of the current upper Hessenberg matrix. Please contact NAG.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please consult NAG for assistance.

NE_INVALID_OPTION

On entry, **Vectors** = Select, but this is not yet implemented.

NE_MAX_ITER

During calculation of a tridiagonal form, there was a failure to compute $\langle value \rangle$ eigenvalues in a total of $\langle value \rangle$ iterations.

NE_NO_LANCZOS_FAC

Could not build an Lanczos factorization. The size of the current Lanczos factorization $= \langle value \rangle$.

NE_NO_SHIFTS_APPLIED

No shifts could be applied during a cycle of the implicitly restarted Lanczos iteration.

NE_OPT_INCOMPAT

The options **Generalized** and **Regular** are incompatible.

NE_REAL_BAND_FAC

Failure during internal factorization of banded matrix. Please contact NAG.

NE_REAL_BAND_SOL

Failure during internal solution of banded system. Please contact NAG.

NE_ZERO_EIGS_FOUND

The number of eigenvalues found to sufficient accuracy is zero.

NE_ZERO_INIT_RESID

The option **Initial Residual** was selected but the starting vector held in **resid** is zero.

7 Accuracy

The relative accuracy of a Ritz value, λ, is considered acceptable if its Ritz estimate $\leq$ **Tolerance** $\times |\lambda|$. The default **Tolerance** used is the ***machine precision*** given by nag_machine_precision (X02AJC).

8 Further Comments

None.

9 Example

The example solves $Ax = \lambda x$ in regular mode, where A is obtained from the standard central difference discretization of the two-dimensional convection-diffusion operator $\frac{d^2u}{dx^2} + \frac{d^2u}{dy^2} = \rho\frac{du}{dx}$ on the unit square with zero Dirichlet boundary conditions. A is stored in LAPACK banded storage format.

9.1 Program Text

```
/* nag_real_symm_banded_sparse_eigensystem_sol (f12fgc) Example Program.
 *
 * Copyright 2005 Numerical Algorithms Group.
 *
 * Mark 8, 2005.
 */

#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <stdio.h>
#include <nagf12.h>
#include <nagf16.h>

int main(void)
{
  /* Constants */
  Integer licomm=140;

  /* Scalars */
  double alpha, h2, resr, sigma;
  Integer exit_status, i, j, k, kl, ku, ksub, ksup, lcomm;
  Integer ldab, n, nconv, ncv, nev, nx;
  /* Nag types */
  NagError fail;

  /* Arrays */
  double *ab=0, *ax=0, *comm=0, *eigv=0, *eigest=0, *mb=0;
  double *resid=0, *v=0;
  Integer *icomm=0;

  exit_status = 0;
  INIT_FAIL(fail);

  Vprintf("nag_real_symm_banded_sparse_eigensystem_sol (f12fgc) Example "
          "Program Results\n\n");

  /* Skip heading in data file */
  Vscanf("%*[^\n] ");
  Vscanf("%ld%ld%ld%*[^\n] ", &nx, &nev, &ncv);
  n = nx * nx;
  kl = nx;
  ku = nx;
  ldab = 2*kl + ku + 1;
  lcomm = 3*n + ncv*ncv + 8*ncv + 60;
  /* Allocate memory */
  if ( !(ab = NAG_ALLOC(ldab * n, double)) ||
      !(ax = NAG_ALLOC(n, double)) ||
      !(comm = NAG_ALLOC(lcomm, double)) ||
      !(eigv = NAG_ALLOC(ncv, double)) ||
      !(eigest = NAG_ALLOC(ncv, double)) ||
      !(mb = NAG_ALLOC(1, double)) ||
      !(resid = NAG_ALLOC(n, double)) ||
      !(v = NAG_ALLOC(n * ncv, double)) ||
      !(icomm = NAG_ALLOC(licomm, Integer)) )
    {
      Vprintf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
  /* Initialise communication arrays for problem using
     nag_real_symm_banded_sparse_eigensystem_init (f12ffc). */
  nag_real_symm_banded_sparse_eigensystem_init(n, nev, ncv, icomm,
                                              licomm, comm, lcomm,
                                              &fail);
  /* Construct the matrix A in banded form and store in AB */
  /* ku, kl are number of superdiagonals and subdiagonals within the
     band of matrices A and M. */
  /* Main diagonal of A. */
  h2 = 1.0 / ((nx + 1) * (nx + 1));
  k = kl + ku;
  for (j = 0; j <= n-1; ++j)
    {
      ab[k] = 4.0 / h2;
      k = k + ldab;
    }
  /* First subdiagonal and superdiagonal of A. */
  ksup = kl + ku - 1;
  ksub = kl + ku + 1;
  for (i = 1; i <= nx; ++i)
    {
      ksup =  ksup + ldab;
      for (j = 1; j <=  nx - 1; ++j)
        {
          ab[ksub] = - 1.0 / h2;
          ab[ksup] = - 1.0 / h2;
          ksup = ksup + ldab;
          ksub = ksub + ldab;
        }
      ksub = ksub + ldab;
    }
  /* kl-th subdiagonal and ku-th super-diagonal. */
  ksup = kl + nx*ldab;
  ksub = 2*kl + ku;
  for (i = 1; i <= nx - 1; ++i)
    {
      for (j = 1; j <= nx; ++j)
        {
          ab[ksup] = - 1.0 / h2;
          ab[ksub] = - 1.0 / h2;
          ksup = ksup + ldab;
          ksub = ksub + ldab;
        }
    }

  /* Find eigenvalues of largest magnitude and the corresponding
     eigenvectors using nag_real_symm_banded_sparse_eigensystem_sol
     (f12fgc). */

  nag_real_symm_banded_sparse_eigensystem_sol(kl, ku, ab, mb, sigma,
                                             &nconv, eigv, v, resid, v,
                                             comm, icomm, &fail);
  if (fail.code == NE_NOERROR)
    {
      /* Compute the residual norm  ||A*x - lambda*x||. */
      k = 0;
      for (j = 0; j <= nconv-1; ++j)
        {
          /* ax <- AV_k, where V_k is kth Ritz vector. */
          /* Compute matrix-vector multiply using nag_dgbmv (f16pbc). */
          nag_dgbmv(Nag_ColMajor, Nag_NoTrans, n, n, kl, ku, 1.0,
                    &ab[kl], ldab, &v[k], 1, 0.0, ax, 1, &fail);
          /* ax <- ax - (lambda_j) * V_k. */
          alpha = -eigv[j];
          /* Compute vector update using nag_daxpby (f16ecc). */
          nag_daxpby(n, alpha, &v[k], 1, 1.0, ax, 1, &fail);
          /* Compute 2-norm of Ritz estimates using nag_dge_norm
             (f16rac).*/
          nag_dge_norm(Nag_ColMajor, Nag_FrobeniusNorm, n, 1, ax, n,
                       &resr, &fail);
          /* Scale. */
          eigest[j] = resr / fabs(eigv[j]);
          k = k + n;
        }

      /* Print computed eigenvalues. */
      Vprintf("\n The %4ld Ritz values are:\n\n", nconv);
      for (j = 0; j <= nconv-1; ++j)
        {
          if (eigest[j] <= 1.0e-10)
            {
              Vprintf("%8ld%5s%7.4f\n", j+1, "", eigv[j]);
            }
          else
            {
              Vprintf("%8ld%5s%7.4f%5s%18.16f\n", j+1, "", eigv[j],
                      " *** ",eigest[j]);
            }
        }
    }
  else
    {
      Vprintf(" Error from nag_real_symm_banded_sparse_eigensystem_sol "
              "(f12fgc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
 END:
  if (ab) NAG_FREE(ab);
  if (ax) NAG_FREE(ax);
  if (comm) NAG_FREE(comm);
  if (eigv) NAG_FREE(eigv);
  if (eigest) NAG_FREE(eigest);
  if (mb) NAG_FREE(mb);
  if (resid) NAG_FREE(resid);
  if (v) NAG_FREE(v);
  if (icomm) NAG_FREE(icomm);

  return exit_status;
}
```

9.2 Program Data

```
nag_real_symm_banded_sparse_eigensystem_sol (f12fgc) Example Program Data
 10  4  10 : Values for nx, nev and ncv
```

9.3 Program Results

```
nag_real_symm_banded_sparse_eigensystem_sol (f12fgc) Example Program Results


 The    4 Ritz values are:

       1     891.1667
       2     919.7807
       3     919.7807
       4     948.3946
```