NAG Library Function Document

nag_sparse_herm_basic_solver (f11gsc)

1 Purpose

nag_sparse_herm_basic_solver (f11gsc) is an iterative solver for a complex Hermitian system of simultaneous linear equations; nag_sparse_herm_basic_solver (f11gsc) is the second in a suite of three functions, where the first function, nag_sparse_herm_basic_setup (f11grc), must be called prior to nag_sparse_herm_basic_solver (f11gsc) to set up the suite, and the third function in the suite, nag_sparse_herm_basic_diagnostic (f11gtc), can be used to return additional information about the computation.

These three functions are suitable for the solution of large sparse complex Hermitian systems of equations.

2 Specification

```
#include <nag.h>
#include <nagf11.h>
void nag_sparse_herm_basic_solver (Integer *irevcm, Complex u[],
     Complex v[], const double wgt[], Complex work[], Integer lwork,
     NagError *fail)
```

3 Description

nag_sparse_herm_basic_solver (f11gsc) solves the complex Hermitian system of linear simultaneous equations $Ax = b$ using either the preconditioned conjugate gradient method (see Hestenes and Stiefel (1952), Golub and Van Loan (1996), Barrett *et al.* (1994) and Dias da Cunha and Hopkins (1994)) or a preconditioned Lanczos method based upon the algorithm SYMMLQ (see Paige and Saunders (1975) and Barrett *et al.* (1994)).

For a general description of the methods employed you are referred to Section 3 in nag_sparse_herm_basic_setup (f11grc).

nag_sparse_herm_basic_solver (f11gsc) can solve the system after the first function in the suite, nag_sparse_herm_basic_setup (f11grc), has been called to initialize the computation and specify the method of solution. The third function in the suite, nag_sparse_herm_basic_diagnostic (f11gtc), can be used to return additional information generated by the computation during monitoring steps and after nag_sparse_herm_basic_solver (f11gsc) has completed its tasks.

nag_sparse_herm_basic_solver (f11gsc) uses **reverse communication**, i.e., nag_sparse_herm_basic_solver (f11gsc) returns repeatedly to the calling program with the argument **irevcm** (see Section 5) set to specified values which require the calling program to carry out a specific task: either to compute the matrix-vector product $v = Au$; to solve the preconditioning equation $Mv = u$; to notify the completion of the computation; or, to allow the calling program to monitor the solution. Through the argument **irevcm** the calling program can cause immediate or tidy termination of the execution. On final exit, the last iterates of the solution and of the residual vectors of the original system of equations are returned.

Reverse communication has the following advantages.

1. Maximum flexibility in the representation and storage of sparse matrices. All matrix operations are performed outside the solver function, thereby avoiding the need for a complicated interface with enough flexibility to cope with all types of storage schemes and sparsity patterns. This applies also to preconditioners.
2. Enhanced user interaction: you can closely monitor the progress of the solution and tidy or immediate termination can be requested. This is useful, for example, when alternative termination

criteria are to be employed or in case of failure of the external functions used to perform matrix operations.

4 References

Barrett R, Berry M, Chan T F, Demmel J, Donato J, Dongarra J, Eijkhout V, Pozo R, Romine C and Van der Vorst H (1994) *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods* SIAM, Philadelphia

Dias da Cunha R and Hopkins T (1994) PIM 1.1 — the parallel iterative method package for systems of linear equations user's guide — Fortran 77 version *Technical Report* Computing Laboratory, University of Kent at Canterbury, Kent, UK

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

Hestenes M and Stiefel E (1952) Methods of conjugate gradients for solving linear systems *J. Res. Nat. Bur. Stand.* **49** 409–436

Higham N J (1988) FORTRAN codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation *ACM Trans. Math. Software* **14** 381–396

Paige C C and Saunders M A (1975) Solution of sparse indefinite systems of linear equations *SIAM J. Numer. Anal.* **12** 617–629

5 Arguments

Note: this function uses **reverse communication.** Its use involves an initial entry, intermediate exits and re-entries, and a final exit, as indicated by the argument **irevcm**. Between intermediate exits and re-entries, **all arguments other than irevcm and v must remain unchanged**.

1: **irevcm** – Integer * *Input/Output*

On initial entry: **irevcm** $= 0$, otherwise an error condition will be raised.

On intermediate re-entry: **irevcm** must either be unchanged from its previous exit value, or can have one of the following values.

irevcm $= 5$
Tidy termination: the computation will terminate at the end of the current iteration. Further reverse communication exits may occur depending on when the termination request is issued. nag_sparse_herm_basic_solver (f11gsc) will then return with the termination code **irevcm** $= 4$. Note that before calling nag_sparse_herm_basic_solver (f11gsc) with **irevcm** $= 5$ the calling program must have performed the tasks required by the value of **irevcm** returned by the previous call to nag_sparse_herm_basic_solver (f11gsc), otherwise subsequently returned values may be invalid.

irevcm $= 6$
Immediate termination: nag_sparse_herm_basic_solver (f11gsc) will return immediately with termination code **irevcm** $= 4$ and with any useful information available. This includes the last iterate of the solution and, for conjugate gradient only, the last iterate of the residual vector. The residual vector is generally not available when the Lanczos method (SYMMLQ) is used. nag_sparse_herm_basic_solver (f11gsc) will then return with the termination code **irevcm** $= 4$.

Immediate termination may be useful, for example, when errors are detected during matrix-vector multiplication or during the solution of the preconditioning equation.

Changing **irevcm** to any other value between calls will result in an error.

On intermediate exit: has the following meanings.

irevcm $= 1$
The calling program must compute the matrix-vector product $v = Au$, where u and v are stored in **u** and **v**, respectively.

irevcm $= 2$
The calling program must solve the preconditioning equation $Mv = u$, where u and v are stored in **u** and **v**, respectively.

irevcm $= 3$
Monitoring step: the solution and residual at the current iteration are returned in the arrays **u** and **v**, respectively. No action by the calling program is required. To return additional information nag_sparse_herm_basic_diagnostic (f11gtc) can be called at this step.

On final exit: if **irevcm** $= 4$, nag_sparse_herm_basic_solver (f11gsc) has completed its tasks. The value of **fail.code** determines whether the iteration has been successfully completed, errors have been detected or the calling program has requested termination.

Constraint: on initial entry, **irevcm** $= 0$; on re-entry, either **irevcm** must remain unchanged or be reset to **irevcm** $= 5$ or 6.

2: **u**$[dim]$ – Complex *Input/Output*

Note: the dimension, *dim*, of the array **u** must be at least n.

On initial entry: an initial estimate, x_0, of the solution of the system of equations $Ax = b$.

On intermediate re-entry: must remain unchanged.

On intermediate exit: the returned value of **irevcm** determines the contents of **u** in the following way.

If **irevcm** $= 1$ or 2, **u** holds the vector u on which the operation specified by **irevcm** is to be carried out.

If **irevcm** $= 3$, **u** holds the current iterate of the solution vector.

On final exit: if after the first call **fail.code** = NE_INT or NE_OUT_OF_SEQUENCE, the array **u** is unchanged from the initial entry to nag_sparse_herm_basic_solver (f11gsc). If after an intermediate call **fail.code** = NE_INT or NE_OUT_OF_SEQUENCE, the array **u** is unchanged from the last entry to nag_sparse_herm_basic_solver (f11gsc). Otherwise, **u** holds the last iterate of the solution of the system of equations, for all returned values of **fail.code**.

3: **v**$[dim]$ – Complex *Input/Output*

Note: the dimension, *dim*, of the array **v** must be at least n.

On initial entry: the right-hand side b of the system of equations $Ax = b$.

On intermediate re-entry: the returned value of **irevcm** determines the contents of **v** in the following way.

If **irevcm** $= 1$ or 2, **v** must store the vector v, the result of the operation specified by the value of **irevcm** returned by the previous call to nag_sparse_herm_basic_solver (f11gsc)

If **irevcm** $= 3$, **v** must remain unchanged.

On intermediate exit: if **irevcm** $= 3$, **v** holds the current iterate of the residual vector. Note that this is an approximation to the true residual vector. Otherwise, it does not contain any useful information.

On final exit: if after the first call **fail.code** = NE_INT or NE_OUT_OF_SEQUENCE, the array **v** is unchanged from the initial entry to nag_sparse_herm_basic_solver (f11gsc). If after an intermediate call **fail.code** = NE_INT or NE_OUT_OF_SEQUENCE, the array **v** is unchanged from the last entry to nag_sparse_herm_basic_solver (f11gsc). Otherwise, **v** stores the last iterate of the residual vector unless the Lanczos method (SYMMLQ) was used and **fail.code** = NE_COEFF_NOT_POS_DEF, NE_CONVERGENCE, NE_PRECOND_NOT_POS_DEF, NE_SINGULAR, NE_USER_STOP or NE_WEIGHT_ZERO, in which case **v** is set to 0.0.

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

Note: the dimension, *dim*, of the array **wgt** must be at least $\max(1, n)$.

On entry: the user-supplied weights, if these are to be used in the computation of the vector norms in the termination criterion (see Sections 3 and 5 in nag_sparse_herm_basic_setup (f11grc)).

Constraint: if weights are to be used, at least one element of **wgt** must be nonzero.

5: **work**[**lwork**] – Complex *Communication Array*

On initial entry: the array **work** as returned by nag_sparse_herm_basic_setup (f11grc) (see also Section 5 in nag_sparse_herm_basic_setup (f11grc)).

On intermediate re-entry: must remain unchanged.

6: **lwork** – Integer *Input*

On initial entry: the dimension of the array **work** (see also Section 3 in nag_sparse_herm_basic_setup (f11grc)). The required amount of workspace is as follows:

Method	**Requirements**
CG	**lwork** $= 120 + 5n + p$.
SYMMLQ	**lwork** $= 120 + 6n + p$.

where

$p = 2 \times (\textbf{maxits} + 1)$, when an estimate of $\sigma_1(A)$ (**sigmax**) is computed;

$p = 0$, otherwise.

Constraint: **lwork** $\geq$ **lwreq**, where **lwreq** is returned by nag_sparse_herm_basic_setup (f11grc).

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

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_ACCURACY

fail.errnum $= 1$

User-requested termination: the required accuracy could not be obtained. However, a reasonable accuracy may have been achieved.

fail.errnum $= 2$

The required accuracy could not be obtained. However, a reasonable accuracy may have been achieved.

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM

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

NE_COEFF_NOT_POS_DEF

The matrix of the coefficients A appears not to be positive definite. The computation cannot continue.

NE_CONVERGENCE

The solution has not converged after ⟨*value*⟩ iterations.

User-requested tidy termination. The solution has not converged after ⟨*value*⟩ iterations.

NE_INT

On entry, **lwork** = ⟨*value*⟩.
Constraint: **lwork** ≥ **lwreq**, where **lwreq** is returned by nag_sparse_herm_basic_setup (f11grc).

On initial entry, **irevcm** = ⟨*value*⟩.
Constraint: **irevcm** = 0.

On intermediate re-entry, **irevcm** = ⟨*value*⟩.
Constraint: either **irevcm** must be unchanged from its previous exit value or **irevcm** = 5 or 6.

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 contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in the Essential Introduction for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

NE_OUT_OF_SEQUENCE

Either nag_sparse_herm_basic_setup (f11grc) was not called before calling nag_sparse_herm_basic_solver (f11gsc) or it has returned an error.

nag_sparse_herm_basic_solver (f11gsc) has already completed its tasks. You need to set a new problem.

NE_PRECOND_NOT_POS_DEF

The preconditioner appears not to be positive definite. The computation cannot continue.

NE_SINGULAR

The matrix of the coefficients A appears to be singular. The computation cannot continue.

NE_USER_STOP

User-requested immediate termination.

NE_WEIGHT_ZERO

The weights in array **wgt** are all zero.

7 Accuracy

On completion, i.e., **irevcm** = 4 on exit, the arrays **u** and **v** will return the solution and residual vectors, x_k and $r_k = b - Ax_k$, respectively, at the kth iteration, the last iteration performed, unless an immediate termination was requested and the Lanczos method (SYMMLQ) was used.

On successful completion, the termination criterion is satisfied to within the user-specified tolerance, as described in Section 3 in nag_sparse_herm_basic_setup (f11grc). The computed values of the left- and right-hand sides of the termination criterion selected can be obtained by a call to nag_sparse_herm_basic_diagnostic (f11gtc).

8 Parallelism and Performance

nag_sparse_herm_basic_solver (f11gsc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_sparse_herm_basic_solver (f11gsc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

The number of operations carried out by nag_sparse_herm_basic_solver (f11gsc) for each iteration is likely to be principally determined by the computation of the matrix-vector products $v = Au$ and by the solution of the preconditioning equation $Mv = u$ in the calling program. Each of these operations is carried out once every iteration.

The number of the remaining operations in nag_sparse_herm_basic_solver (f11gsc) for each iteration is approximately proportional to n. Note that the Lanczos method (SYMMLQ) requires a slightly larger number of operations than the conjugate gradient method.

The number of iterations required to achieve a prescribed accuracy cannot be easily determined at the onset, as it can depend dramatically on the conditioning and spectrum of the preconditioned matrix of the coefficients $\bar{A} = E^{-1}AE^{-\mathrm{H}}$.

Additional matrix-vector products are required for the computation of $\|A\|_1 = \|A\|_\infty$, when this has not been supplied to nag_sparse_herm_basic_setup (f11grc) and is required by the termination criterion employed.

The number of operations required to compute $\sigma_1\left(\bar{A}\right)$ is negligible for reasonable values of **sigtol** and **maxits** (see Sections 5 and 9 in nag_sparse_herm_basic_setup (f11grc)).

If the termination criterion $\|r_k\|_p \leq \tau\left(\|b\|_p + \|A\|_p \times \|x_k\|_p\right)$ is used (see Section 3 in nag_sparse_herm_basic_setup (f11grc)) and $\|x_0\| \gg \|x_k\|$, so that because of loss of significant digits the required accuracy could not be obtained, the iteration is restarted automatically at some suitable point: nag_sparse_herm_basic_solver (f11gsc) sets $x_0 = x_k$ and the computation begins again. For particularly badly scaled problems, more than one restart may be necessary. Naturally, restarting adds to computational costs: it is recommended that the iteration should start from a value x_0 which is as close to the true solution $\tilde{x}$ as can be estimated. Otherwise, the iteration should start from $x_0 = 0$.

10 Example

See Section 10 in nag_sparse_herm_basic_setup (f11grc).