NAG Library Function Document

nag_sparse_nsym_basic_solver (f11bec)

1 Purpose

nag_sparse_nsym_basic_solver (f11bec) is an iterative solver for a real general (nonsymmetric) system of simultaneous linear equations; nag_sparse_nsym_basic_solver (f11bec) is the second in a suite of three functions, where the first function, nag_sparse_nsym_basic_setup (f11bdc), must be called prior to nag_sparse_nsym_basic_solver (f11bec) to set up the suite, and the third function in the suite, nag_sparse_nsym_basic_diagnostic (f11bfc), can be used to return additional information about the computation.

These three functions are suitable for the solution of large sparse general (nonsymmetric) systems of equations.

2 Specification

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

3 Description

nag_sparse_nsym_basic_solver (f11bec) solves the general (nonsymmetric) system of linear simultaneous equations $Ax = b$ of order n, where n is large and the coefficient matrix A is sparse, using one of four available methods: RGMRES, the preconditioned restarted generalized minimum residual method (see Saad and Schultz (1986)); CGS, the preconditioned conjugate gradient squared method (see Sonneveld (1989)); Bi-CGSTAB(ℓ), the bi-conjugate gradient stabilized method of order ℓ (see Van der Vorst (1989) and Sleijpen and Fokkema (1993)); or TFQMR, the transpose-free quasi-minimal residual method (see Freund and Nachtigal (1991) and Freund (1993)).

For a general description of the methods employed you are referred to Section 3 in nag_sparse_nsym_basic_setup (f11bdc).

nag_sparse_nsym_basic_solver (f11bec) can solve the system after the first function in the suite, nag_sparse_nsym_basic_setup (f11bdc), has been called to initialize the computation and specify the method of solution. The third function in the suite, nag_sparse_nsym_basic_diagnostic (f11bfc), can be used to return additional information generated by the computation, during monitoring steps and after nag_sparse_nsym_basic_solver (f11bec) has completed its tasks.

nag_sparse_nsym_basic_solver (f11bec) uses **reverse communication**, i.e., it 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 one of the following tasks:

– compute the matrix-vector product $v = Au$ or $v = A^{\mathrm{T}}u$ (the four methods require the matrix transpose-vector product only if $\|A\|_1$ or $\|A\|_\infty$ is estimated internally by Higham's method (see Higham (1988)));

– solve the preconditioning equation $Mv = u$;

– notify the completion of the computation;

– 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

Freund R W (1993) A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems *SIAM J. Sci. Comput.* **14** 470–482

Freund R W and Nachtigal N (1991) QMR: a Quasi-Minimal Residual Method for Non-Hermitian Linear Systems *Numer. Math.* **60** 315–339

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

Saad Y and Schultz M (1986) GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems *SIAM J. Sci. Statist. Comput.* **7** 856–869

Sleijpen G L G and Fokkema D R (1993) BiCGSTAB(ℓ) for linear equations involving matrices with complex spectrum *ETNA* **1** 11–32

Sonneveld P (1989) CGS, a fast Lanczos-type solver for nonsymmetric linear systems *SIAM J. Sci. Statist. Comput.* **10** 36–52

Van der Vorst H (1989) Bi-CGSTAB, a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems *SIAM J. Sci. Statist. Comput.* **13** 631–644

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: 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_nsym_basic_solver (f11bec) will then return with the termination code **irevcm** = 4. Note that before calling nag_sparse_nsym_basic_solver (f11bec) 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_nsym_basic_solver (f11bec), otherwise subsequently returned values may be invalid.

irevcm = 6
: Immediate termination: nag_sparse_nsym_basic_solver (f11bec) will return immediately with termination code **irevcm** = 4 and with any useful information available. This includes the last iterate of the solution. The residual vector is generally not available.
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 $v = A^{\mathrm{T}}u$, where u and v are stored in **u** and **v**, respectively; RGMRES, CGS and Bi-CGSTAB(ℓ) methods return **irevcm** $= -1$ only if the matrix norm $\|A\|_1$ or $\|A\|_\infty$ is estimated internally using Higham's method. This can only happen if **iterm** $= 1$ in nag_sparse_nsym_basic_setup (f11bdc).

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. nag_sparse_nsym_basic_diagnostic (f11bfc) can be called at this step to return additional information.

On final exit: **irevcm** $= 4$: nag_sparse_nsym_basic_solver (f11bec) has completed its tasks. The value of **fail** 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 5 or 6.

2: **u**$[dim]$ – double *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** $= -1$, 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 **fail**.**code** $=$ NE_OUT_OF_SEQUENCE or **fail**.**code** $=$ NE_INT, the array **u** is unchanged from the last entry to nag_sparse_nsym_basic_solver (f11bec).

Otherwise, **u** holds the last available iterate of the solution of the system of equations, for all returned values of **fail**.

3: **v**$[dim]$ – double *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** $= -1$, 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_nsym_basic_solver (f11bec);

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 **fail.code** = NE_OUT_OF_SEQUENCE or **fail.code** = NE_INT, the array **v** is unchanged from the last entry to nag_sparse_nsym_basic_solver (f11bec).

If **fail.code** = NE_NOERROR or NE_ACCURACY, the array **v** contains the true residual vector of the system of equations (see also Section 6).

Otherwise, **v** stores the last available iterate of the residual vector unless **fail.code** = NE_USER_STOP is returned on last entry, 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_nsym_basic_setup (f11bdc)).

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

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

On initial entry: the array **work** as returned by nag_sparse_nsym_basic_setup (f11bdc) (see also Section 5 in nag_sparse_nsym_basic_setup (f11bdc)).

On intermediate re-entry: must remain unchanged.

6: **lwork** – Integer *Input*

On initial entry: the dimension of the array **work** (see also Sections 3 and 5 in nag_sparse_nsym_basic_setup (f11bdc)). The required amount of workspace is as follows:

Method	**Requirements**
RGMRES	**lwork** $= 100 + n(m+3) + m(m+5) + 1$, where m is the dimension of the basis.
CGS	**lwork** $= 100 + 7n$.
Bi-CGSTAB(ℓ)	**lwork** $= 100 + (2n + \ell)(\ell + 2) + p$, where ℓ is the order of the method.
TFQMR	**lwork** $= 100 + 10n$,

where

$p = 2n$ if $\ell > 1$ and **iterm** $= 2$ was supplied.

$p = n$ if $\ell > 1$ and a preconditioner is used or **iterm** $= 2$ was supplied.

$p = 0$ otherwise.

Constraint: **lwork** $\geq$ **lwreq**, where **lwreq** is returned by nag_sparse_nsym_basic_setup (f11bdc).

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

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

Error details reported in **fail** are only valid on final exit. On intermediate exit, returned values of **fail** should be ignored.

6 Error Indicators and Warnings

NE_ACCURACY

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

User-requested tidy termination. The required accuracy has not been achieved. However, a reasonable accuracy may have been achieved.

nag_sparse_nsym_basic_solver (f11bec) has terminated with reasonable accuracy: the last iterate of the residual satisfied the termination criterion but the exact residual $r = b - Ax$, did not. After the first occurrence of this situation, the iteration was restarted once, but nag_sparse_nsym_basic_solver (f11bec) could not improve on the accuracy. This error code usually implies that your problem has been fully and satisfactorily solved to within or close to the accuracy available on your system. Further iterations are unlikely to improve on this situation. You should call nag_sparse_nsym_basic_diagnostic (f11bfc) to check the values of the left- and right-hand sides of the termination condition.

NE_ALG_FAIL

Algorithm breakdown at iteration no. ⟨*value*⟩.

The last available iterates of the solution and residuals are returned, although it is possible that they are completely inaccurate.

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

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

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** $\geq$ **lwreq**, where **lwreq** is returned by nag_sparse_nsym_basic_setup (f11bdc).

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 2.7.6 in How to Use the NAG Library and its Documentation for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

NE_OUT_OF_SEQUENCE

Either nag_sparse_nsym_basic_setup (f11bdc) was not called before calling nag_sparse_nsym_basic_solver (f11bec) or it has returned an error.

nag_sparse_nsym_basic_solver (f11bec) has already completed its tasks. You need to set a new problem.

nag_sparse_nsym_basic_solver (f11bec) has been called again after returning the termination code **irevcm** $= 4$*. No further computation has been carried out and all input data and data stored for access by nag_sparse_nsym_basic_diagnostic (f11bfc) have remained unchanged.*

NE_USER_STOP

User-requested immediate termination.

The array ***u*** *returns the last iterate of the solution, the array* ***v*** *returns the last iterate of the residual vector, for the CGS and TFQMR methods only.*

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.

On successful completion, the termination criterion is satisfied to within the user-specified tolerance, as described in Section 3 in nag_sparse_nsym_basic_setup (f11bdc). The computed values of the left- and right-hand sides of the termination criterion selected can be obtained by a call to nag_sparse_nsym_basic_diagnostic (f11bfc).

8 Parallelism and Performance

nag_sparse_nsym_basic_solver (f11bec) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_sparse_nsym_basic_solver (f11bec) 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_nsym_basic_solver (f11bec) 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_nsym_basic_solver (f11bec) for each iteration is approximately proportional to n.

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 $\bar{A} = M^{-1}A$ (RGMRES, CGS and TFQMR methods) or $\bar{A} = AM^{-1}$ (Bi-CGSTAB(ℓ) method).

Additional matrix-vector products are required for the computation of $\|A\|_1$ or $\|A\|_\infty$, when this has not been supplied to nag_sparse_nsym_basic_setup (f11bdc) and is required by the termination criterion employed.

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_nsym_basic_setup (f11bdc)) and $\|x_0\| \gg \|x_k\|$, then the required accuracy cannot be obtained due to loss of significant digits. The iteration is restarted automatically at some suitable point: nag_sparse_nsym_basic_solver (f11bec) sets $x_0 = x_k$ and the computation begins again. For particularly badly scaled problems, more than one restart may be necessary. This does not apply to the RGMRES method which, by its own nature, self-restarts every super-iteration. 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_nsym_basic_setup (f11bdc).