NAG Library Function Document
nag_sparse_nherm_fac_sol (f11dqc)

1 Purpose

nag_sparse_nherm_fac_sol (f11dqc) solves a complex sparse non-Hermitian system of linear equations, represented in coordinate storage format, using a restarted generalized minimal residual (RGMRES), conjugate gradient squared (CGS), stabilized bi-conjugate gradient (Bi-CGSTAB), or transpose-free quasi-minimal residual (TFQMR) method, with incomplete LU preconditioning.

2 Specification

```c
#include <nag.h>
#include <nagf11.h>
void nag_sparse_nherm_fac_sol (Nag_SparseNsym_Method method, Integer n,
   Integer nnz, const Complex a[], Integer la, const Integer irow[],
   const Integer icol[], const Integer ipivp[], const Integer ipivq[],
   const Integer istr[], const Integer idiag[], const Complex b[],
   Integer m, double tol, Integer maxitn, Complex x[], double *rnorm,
   Integer *itn, NagError *fail)
```

3 Description

nag_sparse_nherm_fac_sol (f11dqc) solves a complex sparse non-Hermitian linear system of equations

\[ Ax = b, \]

using a preconditioned RGMRES (see Saad and Schultz (1986)), CGS (see Sonneveld (1989)), Bi-CGSTAB(\(\ell\)) (see Van der Vorst (1989) and Sleijpen and Fokkema (1993)), or TFQMR (see Freund and Nachtigal (1991) and Freund (1993)) method.

nag_sparse_nherm_fac_sol (f11dqc) uses the incomplete LU factorization determined by nag_sparse_nherm_fac (f11dnc) as the preconditioning matrix. A call to nag_sparse_nherm_fac_sol (f11dqc) must always be preceded by a call to nag_sparse_nherm_fac (f11dnc). Alternative preconditioners for the same storage scheme are available by calling nag_sparse_nherm_sol (f11dsc).

The matrix \(A\), and the preconditioning matrix \(M\), are represented in coordinate storage (CS) format (see Section 2.1.1 in the f11 Chapter Introduction) in the arrays \(a\), \(irow\) and \(icol\), as returned from nag_sparse_nherm_fac (f11dnc). The array \(a\) holds the nonzero entries in these matrices, while \(irow\) and \(icol\) hold the corresponding row and column indices.

4 References


Numer. Math. 60 315–339

Saad Y and Schultz M (1986) GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems


Sleijpen G L G and Fokkema D R (1993) BiCGSTAB(\(\ell\)) for linear equations involving matrices with complex spectrum

ETNA 1 11–32

5 Arguments

1: method – Nag_SparseNsym_Method

On entry: specifies the iterative method to be used.

method = Nag_SparseNsym_RGMRES
Restarted generalized minimum residual method.

method = Nag_SparseNsym_CGS
Conjugate gradient squared method.

method = Nag_SparseNsym_BiCGSTAB
Bi-conjugate gradient stabilized (Bi) method.

method = Nag_SparseNsym_TFQMR
Transpose-free quasi-minimal residual method.

Constraint: method = Nag_SparseNsym_RGMRES, Nag_SparseNsym_CGS,
Nag_SparseNsym_BiCGSTAB or Nag_SparseNsym_TFQMR.

2: n – Integer

On entry: n, the order of the matrix A. This must be the same value as was supplied in the precedent call to nag_sparse_nherm_fac (f11dnc).

Constraint: n ≥ 1.

3: nnz – Integer

On entry: the number of nonzero elements in the matrix A. This must be the same value as was supplied in the precedent call to nag_sparse_nherm_fac (f11dnc).

Constraint: 1 ≤ nnz ≤ n².

4: a|la| – const Complex

On entry: the values returned in the array a by a previous call to nag_sparse_nherm_fac (f11dnc).

5: la – Integer

On entry: the dimension of the arrays a, irow and icol. This must be the same value as was supplied in the precedent call to nag_sparse_nherm_fac (f11dnc).

Constraint: la ≥ 2 × nnz.

6: irow|la| – const Integer

7: icol|la| – const Integer

8: ipivp|n| – const Integer

9: ipivq|n| – const Integer

10: istr|n + 1| – const Integer

11: idiag|n| – const Integer

On entry: the values returned in arrays irow, icol, ipivp, ipivq, istr andidiag by a previous call to nag_sparse_nherm_fac (f11dnc).

ipivp and ipivq are restored on exit.
12: \( b[n] \) – const Complex

\[ \text{On entry: the right-hand side vector } b. \]

13: \( m \) – Integer

\[ \text{On entry: if } \text{method} = \text{Nag_SparseNsym_RGMRES}, m \text{ is the dimension of the restart subspace.} \]

\[ \text{If } \text{method} = \text{Nag_SparseNsym_BiCGSTAB}, m \text{ is the order } \ell \text{ of the polynomial Bi-CGSTAB method.} \]

\[ \text{Otherwise, } m \text{ is not referenced.} \]

\[ \text{Constraints:} \]

\[ \text{if } \text{method} = \text{Nag_SparseNsym_RGMRES}, 0 < m \leq \min(n, 50); \]

\[ \text{if } \text{method} = \text{Nag_SparseNsym_BiCGSTAB}, 0 < m \leq \min(n, 10). \]

14: \( \text{tol} \) – double

\[ \text{On entry: the required tolerance. Let } x_k \text{ denote the approximate solution at iteration } k, \text{ and } r_k \text{ the} \]

\[ \text{corresponding residual. The algorithm is considered to have converged at iteration } k \text{ if} \]

\[ \|r_k\|_\infty \leq \tau \times (\|b\|_\infty + \|A\|_\infty \|x_k\|_\infty). \]

\[ \text{If } \text{tol} \leq 0.0, \tau = \max(\sqrt{\varepsilon}, 10\varepsilon, \sqrt{n\varepsilon}) \text{ is used, where } \varepsilon \text{ is the } \text{machine precision.} \]

\[ \text{Otherwise } \tau = \max(\text{tol}, 10\varepsilon, \sqrt{n\varepsilon}) \text{ is used.} \]

\[ \text{Constraint: } \text{tol} < 1.0. \]

15: \( \text{maxitn} \) – Integer

\[ \text{On entry: the maximum number of iterations allowed.} \]

\[ \text{Constraint: } \text{maxitn} \geq 1. \]

16: \( x[n] \) – Complex

\[ \text{On entry: an initial approximation to the solution vector } x. \]

\[ \text{On exit: an improved approximation to the solution vector } x. \]

17: \( \text{rnorm} \) – double *

\[ \text{On exit: the final value of the residual norm } \|r_k\|_\infty, \text{where } k \text{ is the output value of itn.} \]

18: \( \text{itn} \) – Integer *

\[ \text{On exit: the number of iterations carried out.} \]

19: \( \text{fail} \) – NagError *

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

6 Error Indicators and Warnings

**NE_ACCURACY**

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

**NE_ALG_FAIL**

Algorithmic breakdown. A solution is returned, although it is possible that it is completely inaccurate.
Dynamic memory allocation failed. See Section 3.2.1.2 in the Essential Introduction for further information.

On entry, argument <value> had an illegal value.

The solution has not converged after <value> iterations.

On entry, maxitn = <value>. Constraint: maxitn ≥ 1.
On entry, n = <value>. Constraint: n ≥ 1.

On entry, la = <value> and nnz = <value>. Constraint: la ≥ 2 × nnz.
On entry, m = <value> and n = <value>. Constraint: m ≥ 1 and m ≤ min(n, <value>).
On entry, nnz = <value> and n = <value>. Constraint: nnz ≤ n².

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.

On entry, i = <value>, icol[i-1] = <value>, and n = <value>. Constraint: icol[i-1] ≥ 1 and icol[i-1] ≤ n.
Check that a, irow, icol, ipivp, ipivq, istr and idiq have not been corrupted between calls to nag_sparse_nherm_fac_sol (f11dqc) and nag_sparse_nherm_fac (f11dnc).
On entry, i = <value>, irow[i-1] = <value>, and n = <value>. Constraint: irow[i-1] ≥ 1 and irow[i-1] ≤ n.
Check that a, irow, icol, ipivp, ipivq, istr and idiq have not been corrupted between calls to nag_sparse_nherm_fac_sol (f11dqc) and nag_sparse_nherm_fac (f11dnc).

The CS representation of the preconditioner is invalid.
Check that a, irow, icol, ipivp, ipivq, istr and idiq have not been corrupted between calls to nag_sparse_nherm_fac (f11dnc) and nag_sparse_nherm_fac_sol (f11dqc).

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_NOT STRICTLY_INCREASING

On entry, \(a[i-1]\) is out of order: \(i = \langle value\rangle\).
Check that \(a, irow, icol, ipivp, ipivq, istr\) and \(idiag\) have not been corrupted between calls to \(nag_sparse_nherm_fac_sol\) (f11dqc) and \(nag_sparse_nherm_fac\) (f11dnc).

On entry, the location \((irow[i-1], icol[i-1])\) is a duplicate: \(i = \langle value\rangle\).
Check that \(a, irow, icol, ipivp, ipivq, istr\) and \(idiag\) have not been corrupted between calls to \(nag_sparse_nherm_fac_sol\) (f11dqc) and \(nag_sparse_nherm_fac\) (f11dnc).

NE_REAL

On entry, \(tol = \langle value\rangle\).
Constraint: \(tol < 1.0\).

7 Accuracy

On successful termination, the final residual \(r_k = b - Ax_k\), where \(k = itn\), satisfies the termination criterion
\[
\|r_k\|_\infty \leq \tau \times (\|b\|_\infty + \|A\|_\infty \|x_k\|_\infty).
\]
The value of the final residual norm is returned in \(rnorm\).

8 Parallelism and Performance

\(nag_sparse_nherm_fac_sol\) (f11dqc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

\(nag_sparse_nherm_fac\) (f11dqc) 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 time taken by \(nag_sparse_nherm_fac_sol\) (f11dqc) for each iteration is roughly proportional to the value of \(nnzc\) returned from the preceding call to \(nag_sparse_nherm_fac\) (f11dnc).

The number of iterations required to achieve a prescribed accuracy cannot be easily determined \(a priori\), as it can depend dramatically on the conditioning and spectrum of the preconditioned coefficient matrix \(A = M^{-1}A\).

10 Example

This example solves a complex sparse non-Hermitian linear system of equations using the CGS method, with incomplete \(LU\) preconditioning.

10.1 Program Text

/* nag_sparse_nherm_fac_sol (f11dqc) Example Program. */
/* Copyright 2014 Numerical Algorithms Group. */
/* Mark 23, 2011. */
#include <nag.h>
#include <nag_stdlib.h>
#include <naga02.h>
#include <nagf11.h>
int main(void)
{
    /* Scalars */
    Integer exit_status = 0;
    double dtol, rnorm, tol;
    Integer i, itn, la, lfill, m, maxitn, n, nnz, nnzc, npivm;
    /* Arrays */
    Complex *a = 0, *b = 0, *x = 0;
    Integer *icol = 0, *idiag = 0, *ipivp = 0, *ipivq = 0,
             *irow = 0, *istr = 0;
    char
    /* NAG types */
    Nag_SparseNsym_Method method;
    Nag_SparseNsym_Piv pstrat;
    Nag_SparseNsym_Fact milu;
    NagError fail;

    INIT_FAIL(fail);

    printf("nag_sparse_nherm_fac_sol (f11dqc) Example Program Results\n\n");

    /* Skip heading in data file*/
    #ifdef _WIN32
        scanf_s("%*[\n\n]");
    #else
        scanf("%*[\n\n]");
    #endif
    /* Read algorithmic parameters*/
    #ifdef _WIN32
        scanf("%"NAG_IFMT"%"NAG_IFMT"%*[\n\n]", &n, &m);
    #else
        scanf("%"NAG_IFMT"%"NAG_IFMT"%*[\n\n]", &n, &m);
    #endif
    #ifdef _WIN32
        scanf("%"NAG_IFMT"%*[\n\n]", &nnz);
    #else
        scanf("%"NAG_IFMT"%*[\n\n]", &nnz);
    #endif
    la = 2 * nnz;
    if (  
        !(a = NAG_ALLOC((la), Complex)) ||  
        !(b = NAG_ALLOC((n), Complex)) ||  
        !(x = NAG_ALLOC((n), Complex)) ||  
        !(icol = NAG_ALLOC((la), Integer)) ||  
        !(idiag = NAG_ALLOC((n), Integer)) ||  
        !(ipivp = NAG_ALLOC((n), Integer)) ||  
        !(ipivq = NAG_ALLOC((n), Integer)) ||  
        !(irow = NAG_ALLOC((la), Integer)) ||  
        !(istr = NAG_ALLOC((n + 1), Integer))
    ) {  
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    #ifdef _WIN32
        scanf_s("%39s%*[\n\n]", nag_enum_arg, _countof(nag_enum_arg));
    #else
        scanf("%39s%*[\n\n]", nag_enum_arg);
    #endif
    /* nag_enum_name_to_value (x04nac).
     * Converts NAG enum member name to value */
    method = (Nag_SparseNsym_Method) nag_enum_name_to_value(nag_enum_arg);
    #ifdef _WIN32
        scanf_s("%"NAG_IFMT"%lf%*[\n\n]", &lfill, &dtol);
    #else
        scanf("%"NAG_IFMT"%lf%*[\n\n]", &lfill, &dtol);
    #endif
    #ifdef _WIN32
        scanf_s("%39s%*[\n\n]", nag_enum_arg, _countof(nag_enum_arg));
    #else
        scanf("%39s%*[\n\n]", nag_enum_arg, _countof(nag_enum_arg));
    #endif
}

f1ldqc

NAG Library Manual

f1ldqc.6

Mark 25
```c
scanf("%39s\n", nag_enum_arg);
#endif
pstrat = (Nag_SparseNsym_Piv) nag_enum_name_to_value(nag_enum_arg);
#endif
milu = (Nag_SparseNsym_Fact) nag_enum_name_to_value(nag_enum_arg);
#endif
for (i = 0; i < nnz; i++) scanf(" ( %lf , %lf ) %"NAG_IFMT"%"NAG_IFMT "%[\n]",
    &a[i].re, &a[i].im, &irow[i], &icol[i]);
#ifdef _WIN32
    scanf_s("%lf%*f%*[\n]", &tol, &maxitn);
#else
    scanf("%lf%*[\n]", &tol, &maxitn);
#endif
for (i = 0; i < nnz; i++) scanf(" ( %lf , %lf ) %"NAG_IFMT"%NAG_IFMT%*[\n]",
    &a[i].re, &a[i].im, &irow[i], &icol[i]);
#ifdef _WIN32
    scanf_s("%*[\n]");
#else
    scanf("%*[\n]");
#endif
for (i = 0; i < nnz; i++) scanf(" ( %lf , %lf )", &b[i].re, &b[i].im);
#endif
for (i = 0; i < nnz; i++) scanf(" ( %lf , %lf )", &x[i].re, &x[i].im);
#endif
for (i = 0; i < n; i++) scanf(" ( %lf , %lf )", &x[i].re, &x[i].im);
#endif
for (i = 0; i < n; i++) scanf(" ( %lf , %lf )", &x[i].re, &x[i].im);
#endif
for (i = 0; i < n; i++) scanf(" ( %lf , %lf )", &x[i].re, &x[i].im);
endif
#endif
for (i = 0; i < n; i++) scanf(" ( %lf , %lf )", &x[i].re, &x[i].im);
#endif
for (i = 0; i < n; i++) scanf(" ( %lf , %lf )", &x[i].re, &x[i].im);
#endif
for (i = 0; i < n; i++) scanf(" ( %lf , %lf )", &x[i].re, &x[i].im);
endif
#endif
for (i = 0; i < n; i++) scanf(" ( %lf , %lf )", &x[i].re, &x[i].im);
endif
endif
```

Mark 25

```bash
f11dqc.7
```

**f11** – Large Scale Linear Systems

`f11dqc`
printf(" (%13.4e, %13.4e) \n", x[i].re, x[i].im); 

END:
NAG_FREE(a);
NAG_FREE(b);
NAG_FREE(x);
NAG_FREE(icol);
NAG_FREE(idiag);
NAG_FREE(ipivp);
NAG_FREE(ipivq);
NAG_FREE(irow);
NAG_FREE(istr);
return exit_status;
}

10.2 Program Data

nag_sparse_nherm_fac_sol (f11dqc) Example Program Data

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>nnz</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>4</td>
<td>24</td>
</tr>
</tbody>
</table>

Nag_SparseNsym_CGS : method
0 0.0 : lfill, dtol
Nag_SparseNsym_CompletePiv : pstrat
Nag_SparseNsym_UnModFact : milu
1.0E-10 100 : tol, maxitn

<table>
<thead>
<tr>
<th>a[i]</th>
<th>irow[i]</th>
<th>icol[i]</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0, 1.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-1.0, 1.</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1.0,-3.</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>4.0, 7.</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>-3.0, 0.</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2.0, 4.</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>-7.0,-5.</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2.0, 1.</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3.0, 2.</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>-4.0, 2.</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>0.0, 1.</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5.0,-3.</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>-1.0, 2.</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>8.0, 6.</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>-3.0,-4.</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>-6.0,-2.</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>5.0,-2.</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>2.0, 0.</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>0.0,-5.</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>-1.0, 5.</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>6.0, 2.</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>-1.0, 4.</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>2.0, 0.</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>3.0, 3.</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

: a[i], irow[i], icol[i], i=0,...,nnz-1

<table>
<thead>
<tr>
<th>b[i]</th>
<th>i=0,...,n-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0, 0.</td>
<td></td>
</tr>
<tr>
<td>0.0, 0.</td>
<td></td>
</tr>
<tr>
<td>0.0, 0.</td>
<td></td>
</tr>
<tr>
<td>0.0, 0.</td>
<td></td>
</tr>
<tr>
<td>0.0, 0.</td>
<td></td>
</tr>
<tr>
<td>0.0, 0.</td>
<td></td>
</tr>
</tbody>
</table>

: x[i], i=0,...,n-1
10.3 Program Results

nag_sparse_nherm_fac_sol (f11dqc) Example Program Results

Converged in 4 iterations
Final residual norm = 1.348e-11

Solution
( 1.0000e+00,  1.0000e+00)
( 2.0000e+00, -1.0000e+00)
( 3.0000e+00,  1.0000e+00)
( 4.0000e+00, -1.0000e+00)
( 3.0000e+00, -1.0000e+00)
( 2.0000e+00,  1.0000e+00)
( 1.0000e+00, -1.0000e+00)
(-1.7424e-12,  3.0000e+00)