

## NAG Library Function Document

### nag\_sparse\_nsym\_precon\_ilu\_solve (f11dbc)

#### 1 Purpose

nag\_sparse\_nsym\_precon\_ilu\_solve (f11dbc) solves a system of linear equations involving the incomplete  $LU$  preconditioning matrix generated by nag\_sparse\_nsym\_fac (f11dac).

#### 2 Specification

```
#include <nag.h>
#include <nagf11.h>

void nag_sparse_nsym_precon_ilu_solve (Nag_TransType trans, Integer n,
    const double a[], Integer la, const Integer irow[],
    const Integer icol[], const Integer ipivp[], const Integer ipivq[],
    const Integer istr[], const Integer iddiag[],
    Nag_SparseNsym_CheckData check, const double y[], double x[],
    NagError *fail)
```

#### 3 Description

nag\_sparse\_nsym\_precon\_ilu\_solve (f11dbc) solves a system of linear equations

$$Mx = y, \quad \text{or} \quad M^T x = y,$$

according to the value of the argument **trans**, where the matrix  $M = PLDUQ$ , corresponds to an incomplete  $LU$  decomposition of a sparse matrix stored in coordinate storage (CS) format (see Section 2.1.1 in the f11 Chapter Introduction), as generated by nag\_sparse\_nsym\_fac (f11dac).

In the above decomposition  $L$  is a lower triangular sparse matrix with unit diagonal elements,  $D$  is a diagonal matrix,  $U$  is an upper triangular sparse matrix with unit diagonal elements and,  $P$  and  $Q$  are permutation matrices.  $L$ ,  $D$  and  $U$  are supplied to nag\_sparse\_nsym\_precon\_ilu\_solve (f11dbc) through the matrix

$$C = L + D^{-1} + U - 2I$$

which is an  $\mathbf{n}$  by  $\mathbf{n}$  sparse matrix, stored in CS format, as returned by nag\_sparse\_nsym\_fac (f11dac). The permutation matrices  $P$  and  $Q$  are returned from nag\_sparse\_nsym\_fac (f11dac) via the arrays **ipivp** and **ipivq**.

It is envisaged that a common use of nag\_sparse\_nsym\_precon\_ilu\_solve (f11dbc) will be to carry out the preconditioning step required in the application of nag\_sparse\_nsym\_basic\_solver (f11bec) to sparse linear systems. nag\_sparse\_nsym\_precon\_ilu\_solve (f11dbc) is used for this purpose by the Black Box function nag\_sparse\_nsym\_fac\_sol (f11dcc).

nag\_sparse\_nsym\_precon\_ilu\_solve (f11dbc) may also be used in combination with nag\_sparse\_nsym\_fac (f11dac) to solve a sparse system of linear equations directly (see Section 9.5 in nag\_sparse\_nsym\_fac (f11dac)). This use of nag\_sparse\_nsym\_precon\_ilu\_solve (f11dbc) is demonstrated in Section 10.

#### 4 References

None.

## 5 Arguments

- 1: **trans** – Nag\_TransType *Input*  
*On entry:* specifies whether or not the matrix  $M$  is transposed.  
**trans** = Nag\_NoTrans  
 $Mx = y$  is solved.  
**trans** = Nag\_Trans  
 $M^T x = y$  is solved.  
*Constraint:* **trans** = Nag\_NoTrans or Nag\_Trans.
- 2: **n** – Integer *Input*  
*On entry:*  $n$ , the order of the matrix  $M$ . This **must** be the same value as was supplied in the preceding call to nag\_sparse\_nsym\_fac (f11dac).  
*Constraint:* **n**  $\geq$  1.
- 3: **a[la]** – const double *Input*  
*On entry:* the values returned in the array **a** by a previous call to nag\_sparse\_nsym\_fac (f11dac).
- 4: **la** – Integer *Input*  
*On entry:* the dimension of the arrays **a**, **irow** and **icol**. This **must** be the same value returned by the preceding call to nag\_sparse\_nsym\_fac (f11dac).
- 5: **irow[la]** – const Integer *Input*  
6: **icol[la]** – const Integer *Input*  
7: **ipivp[n]** – const Integer *Input*  
8: **ipivq[n]** – const Integer *Input*  
9: **istr[n + 1]** – const Integer *Input*  
10: **idiag[n]** – const Integer *Input*  
*On entry:* the values returned in arrays **irow**, **icol**, **ipivp**, **ipivq**, **istr** and **idiag** by a previous call to nag\_sparse\_nsym\_fac (f11dac).
- 11: **check** – Nag\_SparseNsym\_CheckData *Input*  
*On entry:* specifies whether or not the CS representation of the matrix  $M$  should be checked.  
**check** = Nag\_SparseNsym\_Check  
Checks are carried on the values of **n**, **irow**, **icol**, **ipivp**, **ipivq**, **istr** and **idiag**.  
**check** = Nag\_SparseNsym\_NoCheck  
None of these checks are carried out.  
See also Section 9.2.  
*Constraint:* **check** = Nag\_SparseNsym\_Check or Nag\_SparseNsym\_NoCheck.
- 12: **y[n]** – const double *Input*  
*On entry:* the right-hand side vector  $y$ .
- 13: **x[n]** – double *Output*  
*On exit:* the solution vector  $x$ .
- 14: **fail** – NagError \* *Input/Output*  
The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

## 6 Error Indicators and Warnings

### NE\_ALLOC\_FAIL

Dynamic memory allocation failed.

See Section 3.2.1.2 in How to Use the NAG Library and its Documentation for further information.

### NE\_BAD\_PARAM

On entry, argument  $\langle value \rangle$  had an illegal value.

### NE\_INT

On entry,  $\mathbf{n} = \langle value \rangle$ .

Constraint:  $\mathbf{n} \geq 1$ .

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

### NE\_INVALID\_CS

On entry,  $i = \langle value \rangle$ ,  $\mathbf{icol}[i - 1] = \langle value \rangle$ , and  $\mathbf{n} = \langle value \rangle$ .

Constraint:  $\mathbf{icol}[i - 1] \geq 1$  and  $\mathbf{icol}[i - 1] \leq \mathbf{n}$ .

Check that  $\mathbf{a}$ ,  $\mathbf{irow}$ ,  $\mathbf{icol}$ ,  $\mathbf{ipivp}$ ,  $\mathbf{ipivq}$ ,  $\mathbf{istr}$  and  $\mathbf{idiag}$  have not been corrupted between calls to `nag_sparse_nsym_precon_ilu_solve` (f11dbc) and `nag_sparse_nsym_fac` (f11dac).

On entry,  $i = \langle value \rangle$ ,  $\mathbf{irow}[i - 1] = \langle value \rangle$ ,  $\mathbf{n} = \langle value \rangle$ .

Constraint:  $\mathbf{irow}[i - 1] \geq 1$  and  $\mathbf{irow}[i - 1] \leq \mathbf{n}$ .

Check that  $\mathbf{a}$ ,  $\mathbf{irow}$ ,  $\mathbf{icol}$ ,  $\mathbf{ipivp}$ ,  $\mathbf{ipivq}$ ,  $\mathbf{istr}$  and  $\mathbf{idiag}$  have not been corrupted between calls to `nag_sparse_nsym_precon_ilu_solve` (f11dbc) and `nag_sparse_nsym_fac` (f11dac).

### NE\_INVALID\_CS\_PRECOND

On entry,  $\mathbf{idiag}[i - 1]$  appears to be incorrect:  $i = \langle value \rangle$ .

Check that  $\mathbf{a}$ ,  $\mathbf{irow}$ ,  $\mathbf{icol}$ ,  $\mathbf{ipivp}$ ,  $\mathbf{ipivq}$ ,  $\mathbf{istr}$  and  $\mathbf{idiag}$  have not been corrupted between calls to `nag_sparse_nsym_precon_ilu_solve` (f11dbc) and `nag_sparse_nsym_fac` (f11dac).

On entry,  $\mathbf{istr}$  appears to be invalid.

Check that  $\mathbf{a}$ ,  $\mathbf{irow}$ ,  $\mathbf{icol}$ ,  $\mathbf{ipivp}$ ,  $\mathbf{ipivq}$ ,  $\mathbf{istr}$  and  $\mathbf{idiag}$  have not been corrupted between calls to `nag_sparse_nsym_precon_ilu_solve` (f11dbc) and `nag_sparse_nsym_fac` (f11dac).

On entry,  $\mathbf{istr}[i - 1]$  is inconsistent with  $\mathbf{irow}$ :  $i = \langle value \rangle$ .

Check that  $\mathbf{a}$ ,  $\mathbf{irow}$ ,  $\mathbf{icol}$ ,  $\mathbf{ipivp}$ ,  $\mathbf{ipivq}$ ,  $\mathbf{istr}$  and  $\mathbf{idiag}$  have not been corrupted between calls to `nag_sparse_nsym_precon_ilu_solve` (f11dbc) and `nag_sparse_nsym_fac` (f11dac).

### NE\_INVALID\_ROWCOL\_PIVOT

On entry,  $i = \langle value \rangle$ ,  $\mathbf{ipivp}[i - 1] = \langle value \rangle$ ,  $\mathbf{n} = \langle value \rangle$ .

Constraint:  $\mathbf{ipivp}[i - 1] \geq 1$  and  $\mathbf{ipivp}[i - 1] \leq \mathbf{n}$ .

Check that  $\mathbf{a}$ ,  $\mathbf{irow}$ ,  $\mathbf{icol}$ ,  $\mathbf{ipivp}$ ,  $\mathbf{ipivq}$ ,  $\mathbf{istr}$  and  $\mathbf{idiag}$  have not been corrupted between calls to `nag_sparse_nsym_precon_ilu_solve` (f11dbc) and `nag_sparse_nsym_fac` (f11dac).

On entry,  $i = \langle value \rangle$ ,  $\mathbf{ipivq}[i - 1] = \langle value \rangle$ ,  $\mathbf{n} = \langle value \rangle$ .

Constraint:  $\mathbf{ipivq}[i - 1] \geq 1$  and  $\mathbf{ipivq}[i - 1] \leq \mathbf{n}$ .

Check that  $\mathbf{a}$ ,  $\mathbf{irow}$ ,  $\mathbf{icol}$ ,  $\mathbf{ipivp}$ ,  $\mathbf{ipivq}$ ,  $\mathbf{istr}$  and  $\mathbf{idiag}$  have not been corrupted between calls to `nag_sparse_nsym_precon_ilu_solve` (f11dbc) and `nag_sparse_nsym_fac` (f11dac).

On entry, **ipivp**[ $i - 1$ ] is a repeated value:  $i = \langle value \rangle$ .

Check that **a**, **irow**, **icol**, **ipivp**, **ipivq**, **istr** and **idiag** have not been corrupted between calls to `nag_sparse_nsym_precon_ilu_solve` (f11dbc) and `nag_sparse_nsym_fac` (f11dac).

On entry, **ipivq**[ $i - 1$ ] is a repeated value:  $i = \langle value \rangle$ .

Check that **a**, **irow**, **icol**, **ipivp**, **ipivq**, **istr** and **idiag** have not been corrupted between calls to `nag_sparse_nsym_precon_ilu_solve` (f11dbc) and `nag_sparse_nsym_fac` (f11dac).

## NE\_NO\_LICENCE

Your licence key may have expired or may not have been installed correctly.

See Section 3.6.5 in How to Use the NAG Library and its Documentation 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_nsym_precon_ilu_solve` (f11dbc) and `nag_sparse_nsym_fac` (f11dac).

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_nsym_precon_ilu_solve` (f11dbc) and `nag_sparse_nsym_fac` (f11dac).

## 7 Accuracy

If **trans** = Nag\_NoTrans the computed solution  $x$  is the exact solution of a perturbed system of equations  $(M + \delta M)x = y$ , where

$$|\delta M| \leq c(n)\epsilon P|L||D||U|Q,$$

$c(n)$  is a modest linear function of  $n$ , and  $\epsilon$  is the *machine precision*. An equivalent result holds when **trans** = Nag\_Trans.

## 8 Parallelism and Performance

`nag_sparse_nsym_precon_ilu_solve` (f11dbc) is not threaded in any implementation.

## 9 Further Comments

### 9.1 Timing

The time taken for a call to `nag_sparse_nsym_precon_ilu_solve` (f11dbc) is proportional to the value of **nnzc** returned from `nag_sparse_nsym_fac` (f11dac).

### 9.2 Use of check

It is expected that a common use of `nag_sparse_nsym_precon_ilu_solve` (f11dbc) will be to carry out the preconditioning step required in the application of `nag_sparse_nsym_basic_solver` (f11bec) to sparse linear systems. In this situation `nag_sparse_nsym_precon_ilu_solve` (f11dbc) is likely to be called many times with the same matrix  $M$ . In the interests of both reliability and efficiency, you are recommended to set **check** = Nag\_SparseNsym\_Check for the first of such calls, and for all subsequent calls set **check** = Nag\_SparseNsym\_NoCheck.

## 10 Example

This example reads in a sparse nonsymmetric matrix  $A$  and a vector  $y$ . It then calls `nag_sparse_nsym_fac` (f11dac), with **lfill** = -1 and **dtol** = 0.0, to compute the **complete**  $LU$  decomposition

$$A = PLDUQ.$$

Finally it calls `nag_sparse_nsym_precon_ilu_solve` (f11dbc) to solve the system

$$PLDUQx = y.$$

## 10.1 Program Text

```

/* nag_sparse_nsym_precon_ilu_solve (f11dbc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf11.h>
int main(void)
{
    /* Scalars */
    Integer exit_status = 0;
    double dtol;
    Integer i, la, lfill, n, nnz, nnzc, npivm;
    /* Arrays */
    double *a = 0, *x = 0, *y = 0;
    Integer *icol = 0, *idiag = 0, *ipivp = 0, *ipivq = 0, *irow = 0, *istr = 0;
    /* NAG types */
    Nag_SparseNsym_Piv pstrat;
    Nag_SparseNsym_Fact milu;
    Nag_SparseNsym_CheckData check;
    Nag_TransType trans;
    NagError fail;

    INIT_FAIL(fail);

    printf("nag_sparse_nsym_precon_ilu_solve (f11dbc) Example Program Results");
    printf("\n\n");
    /* Skip heading in data file */
#ifdef _WIN32
    scanf_s("%*[\n]");
#else
    scanf("%*[\n]");
#endif
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "%*[\n]", &n);
#else
    scanf("%" NAG_IFMT "%*[\n]", &n);
#endif
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "%*[\n]", &nnz);
#else
    scanf("%" NAG_IFMT "%*[\n]", &nnz);
#endif
    la = 2 * nnz;
    if (!(a = NAG_ALLOC((la), double)) ||
        !(x = NAG_ALLOC((n), double)) ||
        !(y = NAG_ALLOC((n), double)) ||
        !(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;
    }
    /* Read the nonzero elements of the matrix a */

```

```

    for (i = 0; i < nnz; i++)
#ifdef _WIN32
        scanf_s("%lf%" NAG_IFMT "%" NAG_IFMT "%*[\n] ", &a[i], &irow[i],
                &icol[i]);
#else
        scanf("%lf%" NAG_IFMT "%" NAG_IFMT "%*[\n] ", &a[i], &irow[i], &icol[i]);
#endif
    /* Read the vector y */
#ifdef _WIN32
        for (i = 0; i < n; i++)
            scanf_s("%lf", &y[i]);
#else
        for (i = 0; i < n; i++)
            scanf("%lf", &y[i]);
#endif

    /* Calculate LU factorization */
    lfill = -1;
    dtol = 0.0;
    pstrat = Nag_SparseNsym_CompletePiv;
    milu = Nag_SparseNsym_UnModFact;
    /* nag_sparse_nsym_fac (f11dac).
     * Incomplete LU factorization (nonsymmetric)
     */
    nag_sparse_nsym_fac(n, nnz, &a, &la, &irow, &icol, lfill, dtol, pstrat,
                       milu, ipivp, ipivq, istr, idiag, &nnzc, &npivm, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_sparse_nsym_fac (f11dac)\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }
    /* Check value of npivm */
    if (npivm > 0) {
        printf("Factorization is not complete \n");
        goto END;
    }
    /* Solve P L D U x = y */
    check = Nag_SparseNsym_Check;
    trans = Nag_NoTrans;
    /* nag_sparse_nsym_precon_ilu_solve (f11dbc)
     * Solution of linear system involving incomplete LU preconditioning matrix
     */
    nag_sparse_nsym_precon_ilu_solve(trans, n, a, la, irow, icol, ipivp, ipivq,
                                     istr, idiag, check, y, x, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_sparse_nsym_precon_ilu_solve (f11dbc)\n%s\n",
              fail.message);
        exit_status = 2;
        goto END;
    }
    /* Output results */
    printf(" Solution of linear system \n");
    for (i = 0; i < n; i++)
        printf("%16.4e\n", x[i]);
END:
    NAG_FREE(a);
    NAG_FREE(x);
    NAG_FREE(y);
    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_nsym_precon_ilu_solve (f11dbc) Example Program Data
 4                                     : n
11                                    : nnz
 1.   1   2
 1.   1   3
-1.   2   1
 2.   2   3
 2.   2   4
 3.   3   1
-2.   3   4
 1.   4   1
-2.   4   2
 1.   4   3
 1.   4   4      : a[i], irow[i], icol[i], i=0,...,nnz-1
5.0 13.0 -5.0 4.0 : y[i], i=0,...,n-1
```

## 10.3 Program Results

```
nag_sparse_nsym_precon_ilu_solve (f11dbc) Example Program Results
```

```
Solution of linear system
 1.0000e+00
 2.0000e+00
 3.0000e+00
 4.0000e+00
```

---