

## NAG Library Function Document

### nag\_superlu\_condition\_number\_lu (f11mgc)

#### 1 Purpose

nag\_superlu\_condition\_number\_lu (f11mgc) computes an estimate of the reciprocal of the condition number of a sparse matrix given an  $LU$  factorization of the matrix computed by nag\_superlu\_lu\_factorize (f11mec).

#### 2 Specification

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

void nag_superlu_condition_number_lu (Nag_NormType norm, Integer n,
    const Integer il[], const double lval[], const Integer iu[],
    const double uval[], double anorm, double *rcond, NagError *fail)
```

#### 3 Description

nag\_superlu\_condition\_number\_lu (f11mgc) estimates the condition number of a real sparse matrix  $A$ , in either the 1-norm or the  $\infty$ -norm:

$$\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1 \quad \text{or} \quad \kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty.$$

Note that  $\kappa_\infty(A) = \kappa_1(A^T)$ .

Because the condition number is infinite if  $A$  is singular, the function actually returns an estimate of the **reciprocal** of the condition number.

The function should be preceded by a call to nag\_superlu\_matrix\_norm (f11mlc) to compute  $\|A\|_1$  or  $\|A\|_\infty$ , and a call to nag\_superlu\_lu\_factorize (f11mec) to compute the  $LU$  factorization of  $A$ . The function then estimates  $\|A^{-1}\|_1$  or  $\|A^{-1}\|_\infty$  and computes the reciprocal of the condition number.

#### 4 References

None.

#### 5 Arguments

- 1: **norm** – Nag\_NormType *Input*  
*On entry:* indicates whether  $\kappa_1(A)$  or  $\kappa_\infty(A)$  is to be estimated.  
**norm** = Nag\_RealOneNorm  
 $\kappa_1(A)$  is estimated.  
**norm** = Nag\_RealInfNorm  
 $\kappa_\infty(A)$  is estimated.  
*Constraint:* **norm** = Nag\_RealOneNorm or Nag\_RealInfNorm.
- 2: **n** – Integer *Input*  
*On entry:*  $n$ , the order of the matrix  $A$ .  
*Constraint:*  $n \geq 0$ .

- 3: **il**[*dim*] – const Integer *Input*  
**Note:** the dimension, *dim*, of the array **il** must be at least as large as the dimension of the array of the same name in nag\_superlu\_lu\_factorize (f11mec).  
*On entry:* records the sparsity pattern of matrix *L* as computed by nag\_superlu\_lu\_factorize (f11mec).
- 4: **lval**[*dim*] – const double *Input*  
**Note:** the dimension, *dim*, of the array **lval** must be at least as large as the dimension of the array of the same name in nag\_superlu\_lu\_factorize (f11mec).  
*On entry:* records the nonzero values of matrix *L* and some nonzero values of matrix *U* as computed by nag\_superlu\_lu\_factorize (f11mec).
- 5: **iu**[*dim*] – const Integer *Input*  
**Note:** the dimension, *dim*, of the array **iu** must be at least as large as the dimension of the array of the same name in nag\_superlu\_lu\_factorize (f11mec).  
*On entry:* records the sparsity pattern of matrix *U* as computed by nag\_superlu\_lu\_factorize (f11mec).
- 6: **uval**[*dim*] – const double *Input*  
**Note:** the dimension, *dim*, of the array **uval** must be at least as large as the dimension of the array of the same name in nag\_superlu\_lu\_factorize (f11mec).  
*On entry:* records some nonzero values of matrix *U* as computed by nag\_superlu\_lu\_factorize (f11mec).
- 7: **anorm** – double *Input*  
*On entry:* if **norm** = Nag\_RealOneNorm, the 1-norm of the matrix *A*.  
If **norm** = Nag\_RealInfNorm, the  $\infty$ -norm of the matrix *A*.  
**anorm** may be computed by calling nag\_superlu\_matrix\_norm (f11mlc) with the same value for the argument **norm**.  
*Constraint:* **anorm**  $\geq$  0.0.
- 8: **rcond** – double \* *Output*  
*On exit:* an estimate of the reciprocal of the condition number of *A*. **rcond** is set to zero if exact singularity is detected or the estimate underflows. If **rcond** is less than *machine precision*, *A* is singular to working precision.
- 9: **fail** – NagError \* *Input/Output*  
The NAG error argument (see Section 3.6 in the Essential Introduction).

## 6 Error Indicators and Warnings

### NE\_ALLOC\_FAIL

Dynamic memory allocation failed.

### NE\_BAD\_PARAM

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

**NE\_INT**

On entry, **n** =  $\langle value \rangle$ .  
 Constraint: **n**  $\geq 0$ .

**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.

**NE\_REAL**

On entry, **anorm** =  $\langle value \rangle$ .  
 Constraint: **anorm**  $\geq 0.0$ .

**7 Accuracy**

The computed estimate **rcond** is never less than the true value  $\rho$ , and in practice is nearly always less than  $10\rho$ , although examples can be constructed where **rcond** is much larger.

**8 Parallelism and Performance**

nag\_superlu\_condition\_number\_lu (f11mgc) is not threaded by NAG in any implementation.

nag\_superlu\_condition\_number\_lu (f11mgc) 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 Users' Note for your implementation for any additional implementation-specific information.

**9 Further Comments**

A call to nag\_superlu\_condition\_number\_lu (f11mgc) involves solving a number of systems of linear equations of the form  $Ax = b$  or  $A^T x = b$ .

**10 Example**

This example estimates the condition number in the 1-norm of the matrix  $A$ , where

$$A = \begin{pmatrix} 2.00 & 1.00 & 0 & 0 & 0 \\ 0 & 0 & 1.00 & -1.00 & 0 \\ 4.00 & 0 & 1.00 & 0 & 1.00 \\ 0 & 0 & 0 & 1.00 & 2.00 \\ 0 & -2.00 & 0 & 0 & 3.00 \end{pmatrix}.$$

Here  $A$  is nonsymmetric and must first be factorized by nag\_superlu\_lu\_factorize (f11mec). The true condition number in the 1-norm is 20.25.

**10.1 Program Text**

```

/* nag_superlu_condition_number_lu (f11mgc) Example Program.
 *
 * Copyright 2005 Numerical Algorithms Group.
 *
 * Mark 8, 2005.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf11.h>

```

```

int main(void)
{
    double          anorm, d, flop, rcond, thresh;
    Integer         exit_status = 0, i, n, nnz, nnzl, nnzu, nzlmx,
                   nzlumx, nzumx;
    double          *a = 0, *lval = 0, *uval = 0;
    Integer         *icolzp = 0, *il = 0, *iprm = 0, *irowix = 0;
    Integer         *iu = 0;
    /* Nag types */
    NagError        fail;
    Nag_ColumnPermutationType ispec;
    Nag_NormType    norm;

    INIT_FAIL(fail);

    printf("nag_superlu_condition_number_lu (f11mgc) Example Program "
           "Results\n\n");
    /* Skip heading in data file */
    scanf("%s[^\n] ");
    /* Read order of matrix */
    scanf("%ld%s[^\n] ", &n);
    /* Read the matrix A */
    if (!(icolzp = NAG_ALLOC(n+1, Integer)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    for (i = 1; i <= n + 1; ++i)
        scanf("%ld%s[^\n] ", &icolzp[i - 1]);
    nnz = icolzp[n] - 1;
    /* Allocate memory */
    if (!(irowix = NAG_ALLOC(nnz, Integer)) ||
        !(a = NAG_ALLOC(nnz, double)) ||
        !(il = NAG_ALLOC(7*n+8*nnz+4, Integer)) ||
        !(iu = NAG_ALLOC(2*n+8*nnz+1, Integer)) ||
        !(uval = NAG_ALLOC(8*nnz, double)) ||
        !(lval = NAG_ALLOC(8*nnz, double)) ||
        !(iprm = NAG_ALLOC(7*n, Integer)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    for (i = 1; i <= nnz; ++i)
        scanf("%lf%ld%s[^\n] ", &a[i - 1], &irowix[i - 1]);

    /* Calculate COLAMD permutation */
    ispec = Nag_Sparse_Colamd;
    /* nag_superlu_column_permutation (f11mdc).
     * Real sparse nonsymmetric linear systems, setup for
     * nag_superlu_lu_factorize (f11mec)
     */
    nag_superlu_column_permutation(ispec, n, icolzp, irowix, iprm, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf(
            "Error from nag_superlu_column_permutation (f11mdc).\n%s\n",
            fail.message);
        exit_status = 1;
        goto END;
    }

    /* Factorise */
    thresh = 1.;
    nzlmx = 8*nnz;
    nzlumx = 8*nnz;
    nzumx = 8*nnz;
    /* nag_superlu_lu_factorize (f11mec).
     * LU factorization of real sparse matrix
     */
}

```

```

nag_superlu_lu_factorize(n, irowix, a, iprm, thresh, nzlmx, &nzlxm, nzumx,
                        il, lval, iu, uval, &nnzl, &nnzu, &flop, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_superlu_lu_factorize (f11mec).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
}

/* Calculate norm */
norm = Nag_RealOneNorm;
/* nag_superlu_matrix_norm (f11mlc).
 * 1-norm, infinity-norm, largest absolute element, real
 * general matrix
 */
nag_superlu_matrix_norm(norm, &anorm, n, icolzp, irowix, a, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_superlu_matrix_norm (f11mlc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
}

/* Calculate condition number */
/* nag_superlu_condition_number_lu (f11mgc).
 * Estimate condition number of real matrix, matrix already
 * factorized by nag_superlu_lu_factorize (f11mec)
 */
nag_superlu_condition_number_lu(norm, n, il, lval, iu, uval, anorm, &rcond,
                                &fail);
if (fail.code != NE_NOERROR)
{
    printf(
        "Error from nag_superlu_condition_number_lu (f11mgc).\n%s\n",
        fail.message);
    exit_status = 1;
    goto END;
}

/* Output result */
d = 1 / rcond;
printf("\n  Norm,  Condition number\n %7.3f,%7.3f\n\n", anorm, d);

END:
NAG_FREE(a);
NAG_FREE(lval);
NAG_FREE(uval);
NAG_FREE(icolzp);
NAG_FREE(il);
NAG_FREE(iprm);
NAG_FREE(irowix);
NAG_FREE(iu);

return exit_status;
}

```

## 10.2 Program Data

```

nag_superlu_condition_number_lu (f11mgc) Example Program Data
5  n
1
3
5
7
9
12  icolzp(i) i=0..n
2.  1
4.  3

```

```
1. 1
-2. 5
1. 2
1. 3
-1. 2
1. 4
1. 3
2. 4
3. 5    a(i), irowix(i) i=0..nnz-1
```

### **10.3 Program Results**

nag\_superlu\_condition\_number\_lu (f11mgc) Example Program Results

```
Norm, Condition number
6.000, 20.250
```

---