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 ∞-norm:

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

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

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 $\left\|A^{-1}\right\|_1$ or $\left\|A^{-1}\right\|_\infty$ 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: $\mathbf{n} \geq 0$.

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 ∞-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** ≥ 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 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 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_INT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 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.

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_REAL

On entry, **anorm** = ⟨*value*⟩.
Constraint: **anorm** ≥ 0.0.

7 Accuracy

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

8 Parallelism and Performance

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 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

A call to nag_superlu_condition_number_lu (f11mgc) involves solving a number of systems of linear equations of the form $Ax = b$ or $A^{\mathrm{T}}x = b$.

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.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#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 */
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
  /* Read order of matrix */
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%*[^\n] ", &n);
#else
  scanf("%" NAG_IFMT "%*[^\n] ", &n);
#endif
  /* 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)
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "%*[^\n] ", &icolzp[i - 1]);
#else
    scanf("%" NAG_IFMT "%*[^\n] ", &icolzp[i - 1]);
#endif
  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)
#ifdef _WIN32
    scanf_s("%lf%" NAG_IFMT "%*[^\n] ", &a[i - 1], &irowix[i - 1]);
#else
    scanf("%lf%" NAG_IFMT "%*[^\n] ", &a[i - 1], &irowix[i - 1]);
#endif

  /* 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, &nzlumx, 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
```