NAG Library Function Document

nag_superlu_diagnostic_lu (f11mmc)

1 Purpose

nag_superlu_diagnostic_lu (f11mmc) computes the reciprocal pivot growth factor of an LU factorization of a real sparse matrix in compressed column (Harwell–Boeing) format.

2 Specification

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

```
void nag_superlu_diagnostic_lu (Integer n, const Integer icolzp[],
     const double a[], const Integer iprm[], const Integer il[],
     const double lval[], const Integer iu[], const double uval[],
     double *rpg, NagError *fail)
```

3 Description

nag_superlu_diagnostic_lu (f11mmc) computes the reciprocal pivot growth factor $\max_j\left(\left\|A_j\right\|_\infty/\left\|U_j\right\|_\infty\right)$ from the columns A_j and U_j of an LU factorization of the matrix A, $P_rAP_c = LU$ where P_r is a row permutation matrix, P_c is a column permutation matrix, L is unit lower triangular and U is upper triangular as computed by nag_superlu_lu_factorize (f11mec).

4 References

None.

5 Arguments

1: **n** – Integer *Input*

On entry: n, the order of the matrix A.

Constraint: $\mathbf{n} \geq 0$.

2: **icolzp**[*dim*] – const Integer *Input*

Note: the dimension, *dim*, of the array **icolzp** must be at least $\mathbf{n} + 1$.

On entry: **icolzp**$[i-1]$ contains the index in A of the start of a new column. See Section 2.1.3 in the f11 Chapter Introduction.

3: **a**[*dim*] – const double *Input*

Note: the dimension, *dim*, of the array **a** must be at least **icolzp**[**n**] $- 1$, the number of nonzeros of the sparse matrix A.

On entry: the array of nonzero values in the sparse matrix A.

4: **iprm**[$\mathbf{7} \times \mathbf{n}$] – const Integer *Input*

On entry: the column permutation which defines P_c, the row permutation which defines P_r, plus associated data structures as computed by nag_superlu_lu_factorize (f11mec).

5: **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).

6: **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).

7: **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).

8: **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).

9: **rpg** – double * *Output*

On exit: the reciprocal pivot growth factor $\max_j\left(\left\|A_j\right\|_\infty/\left\|U_j\right\|_\infty\right)$. If the reciprocal pivot growth factor is much less than 1, the stability of the LU factorization may be poor.

10: **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, $\mathbf{n} = \langle value\rangle$.
Constraint: $\mathbf{n} \geq 0$.

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_INVALID_PERM_COL

Incorrect column permutations in array **iprm**.

7 Accuracy

Not applicable.

8 Parallelism and Performance

Not applicable.

9 Further Comments

If the reciprocal pivot growth factor, **rpg**, is much less than 1, then the factorization of the matrix A could be poor. This means that using the factorization to obtain solutions to a linear system, forward error bounds and estimates of the condition number could be unreliable. Consider increasing the **thresh** argument in the call to nag_superlu_lu_factorize (f11mec).

10 Example

To compute the reciprocal pivot growth for the factorization of the matrix A, where

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

In this case, it should be equal to 1.0.

10.1 Program Text

```
/* nag_superlu_diagnostic_lu (f11mmc) 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                     flop, rpg, 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 */
  Nag_ColumnPermutationType ispec;
  NagError                   fail;

  INIT_FAIL(fail);

  printf(
          "nag_superlu_diagnostic_lu (f11mmc) Example Program Results\n\n");
  /* Skip heading in data file */
  scanf("%*[^\n] ");
  /* Read order of matrix */
  scanf("%ld%*[^\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%*[^\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%*[^\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, &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 reciprocal pivot growth */
  /* nag_superlu_diagnostic_lu (f11mmc).
   * Real sparse nonsymmetric linear systems, diagnostic for
   * nag_superlu_lu_factorize (f11mec)
   */
  nag_superlu_diagnostic_lu(n, icolzp, a, iprm, il, lval, iu, uval, &rpg,
                            &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_superlu_diagnostic_lu (f11mmc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  /* Output result */
  printf("\n");
  printf("%s\n%7.3f\n", "Reciprocal pivot growth", rpg);

 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_diagnostic_lu (f11mmc) 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_diagnostic_lu (f11mmc) Example Program Results


Reciprocal pivot growth
  1.000
```