NAG C Library Function Document

nag_superlu_matrix_norm (f11mlc)

1 Purpose

nag_superlu_matrix_norm (f11mlc) computes the 1-norm, the ∞-norm or the maximum absolute value of the elements of a real, square, sparse matrix which is held in compressed column (Harwell–Boeing) format.

2 Specification

```
#include <nag.h>
#include <nagf11.h>
void nag_superlu_matrix_norm (Nag_NormType norm, double *anorm, Integer n,
     const Integer icolzp[], const Integer irowix[], const double a[],
     NagError *fail)
```

3 Description

nag_superlu_matrix_norm (f11mlc) computes various quantities relating to norms of a real, sparse n by n matrix A presented in compressed column (Harwell–Boeing) format.

4 References

None.

5 Arguments

1: **norm** – Nag_NormType *Input*

On entry: specifies the value to be returned in **anorm**.

norm = **Nag_RealOneNorm**

The 1-norm $\|A\|_1$ of the matrix is computed, that is $\max\limits_{1\le j\le n}\sum\limits_{i=1}^{n}\left|A_{ij}\right|$.

norm = **Nag_RealInfNorm**

The ∞-norm $\|A\|_\infty$ of the matrix is computed, that is $\max\limits_{1\le i\le n}\sum\limits_{j=1}^{n}\left|A_{ij}\right|$.

norm = **Nag_RealMaxNorm**

The value $\max\limits_{1\le i,j\le n}\left|A_{ij}\right|$ (not a norm).

Constraint: **norm** = **Nag_RealOneNorm**, **Nag_RealInfNorm** or **Nag_RealMaxNorm**.

2: **anorm** – double * *Output*

On exit: the computed quantity relating the matrix.

3: **n** – Integer *Input*

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

Constraint: **n** ≥ 0.

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

Note: the dimension, *dim*, of the array **icolzp** must be at least **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.

5: **irowix**[dim] – const Integer *Input*

Note: the dimension, *dim*, of the array **irowix** must be at least **icolzp**[**n**] − 1, the number of non-zeros of the sparse matrix A.

On entry: the row index array of the sparse matrix A.

6: **a**[dim] – const double *Input*

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

On entry: the array of non-zero values in the sparse matrix A.

7: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.6 of the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.

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 consult NAG for assistance.

7 Accuracy

Not applicable.

8 Further Comments

None.

9 Example

To compute norms and maximum absolute value of the matrix A, where

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

9.1 Program Text

```
/* nag_superlu_matrix_norm (f11mlc) 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;
  Integer exit_status=0, i, n, nnz;
  double *a=0;
  Integer *icolzp=0, *irowix=0;
  /* Nag types */
  NagError fail;
  Nag_NormType norm;

  INIT_FAIL(fail);
  Vprintf( "nag_superlu_matrix_norm (f11mlc) Example Program Results\n\n");
  /* Skip heading in data file */
  Vscanf("%*[^\n] ");
  /* Read order of matrix and number of right hand sides */
  Vscanf("%ld%*[^\n] ",   &n);
  /* Read the matrix A */
  if ( !(icolzp = NAG_ALLOC(n+1, Integer)))
    {
      Vprintf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
  for (i = 1; i <= n + 1; ++i)
    Vscanf("%ld%*[^\n] ", &icolzp[i - 1]);
  nnz = icolzp[n] - 1;
  /* Allocate memory */
  if ( !(a = NAG_ALLOC(nnz, double)) ||
       !(irowix = NAG_ALLOC(nnz, Integer)) )
    {
      Vprintf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
  for (i = 1; i <= nnz; ++i)
    Vscanf("%lf%ld%*[^\n] ",   &a[i - 1],  &irowix[i - 1]);
  /* Calculate 1-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)
    {
      Vprintf("Error from nag_superlu_matrix_norm (f11mlc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  /* Output norm */
  Vprintf("%s\n%7.3f\n",   "One-norm",  anorm);

  /* Calculate M-norm */
  norm = Nag_RealMaxNorm;
  /* nag_superlu_matrix_norm (f11mlc), see above. */
```

```
  nag_superlu_matrix_norm(norm, &anorm, n, icolzp, irowix, a, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from nag_superlu_matrix_norm (f11mlc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  /* Output norm */
  Vprintf("\n");
  Vprintf("%s\n%7.3f\n",   "Max",  anorm);

  /* Calculate I-norm */
  norm = Nag_RealInfNorm;
  /* nag_superlu_matrix_norm (f11mlc), see above. */
  nag_superlu_matrix_norm(norm, &anorm, n, icolzp, irowix, a, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from nag_superlu_matrix_norm (f11mlc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  /* Output norm */
  Vprintf("\n");
  Vprintf("%s\n%7.3f\n",   "Infinity-norm",  anorm);

 END:
  if (a) NAG_FREE(a);
  if (icolzp) NAG_FREE(icolzp);
  if (irowix) NAG_FREE(irowix);

  return exit_status;
}
```

9.2 Program Data

```
nag_superlu_matrix_norm (f11mlc) 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
```

9.3 Program Results

```
nag_superlu_matrix_norm (f11mlc) Example Program Results

One-norm
  6.000

Max
  4.000

Infinity-norm
```

```
 6.000
```