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 $\left(\|A_j\|_\infty/\|U_j\|_\infty\right)$ 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
   On entry: $n$, the order of the matrix $A$.
   Constraint: $n \geq 0$.

2: icolzp[dim] – const Integer
   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.

3: a[dim] – const double
   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[7 × n] – const Integer
   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: \( \text{il}[\text{dim}] \) – const Integer  
\textit{Input}

\textbf{Note:} the dimension, \( \text{dim} \), of the array \text{il} must be at least as large as the dimension of the array of the same name in \text{nag_superlu_lu_factorize} (f11mec).

\textit{On entry:} records the sparsity pattern of matrix \( L \) as computed by \text{nag_superlu_lu_factorize} (f11mec).

6: \( \text{lval}[\text{dim}] \) – const double  
\textit{Input}

\textbf{Note:} the dimension, \( \text{dim} \), of the array \text{lval} must be at least as large as the dimension of the array of the same name in \text{nag_superlu_lu_factorize} (f11mec).

\textit{On entry:} records the nonzero values of matrix \( L \) and some nonzero values of matrix \( U \) as computed by \text{nag_superlu_lu_factorize} (f11mec).

7: \( \text{iu}[\text{dim}] \) – const Integer  
\textit{Input}

\textbf{Note:} the dimension, \( \text{dim} \), of the array \text{iu} must be at least as large as the dimension of the array of the same name in \text{nag_superlu_lu_factorize} (f11mec).

\textit{On entry:} records the sparsity pattern of matrix \( U \) as computed by \text{nag_superlu_lu_factorize} (f11mec).

8: \( \text{uval}[\text{dim}] \) – const double  
\textit{Input}

\textbf{Note:} the dimension, \( \text{dim} \), of the array \text{uval} must be at least as large as the dimension of the array of the same name in \text{nag_superlu_lu_factorize} (f11mec).

\textit{On entry:} records some nonzero values of matrix \( U \) as computed by \text{nag_superlu_lu_factorize} (f11mec).

9: \( \text{rpg} \) – double *  
\textit{Output}

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

10: \( \text{fail} \) – \text{NagError} *  
\textit{Input/Output}

The NAG error argument (see Section 3.6 in the Essential Introduction).

6  \textbf{Error Indicators and Warnings}

\textbf{NE_ALLOC_FAIL}

Dynamic memory allocation failed.

See Section 3.2.1.2 in the Essential Introduction for further information.

\textbf{NE_BAD_PARAM}

On entry, argument \( \langle \text{value} \rangle \) had an illegal value.

\textbf{NE_INT}

On entry, \( n = \langle \text{value} \rangle \).

Constraint: \( n \geq 0 \).

\textbf{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 the Essential Introduction for further information.
Incorrect column permutations in array iprm.

Your licence key may have expired or may not have been installed correctly. See Section 3.6.5 in the Essential Introduction for further information.

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 2014 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, nnz1, nnz2, nzlmx, nzlmx;
    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 */ 
#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, 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;
}

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卢 (f1lmmc) Example Program Results

Reciprocal pivot growth
1.000