1 Purpose
nag_dgbequ (f07bfc) computes diagonal scaling matrices $D_R$ and $D_C$ intended to equilibrate a real $m$ by $n$ band matrix $A$ of band width $(k_l + k_u + 1)$, and reduce its condition number.

2 Specification
#include <nag.h>
#include <nagf07.h>

void nag_dgbequ (Nag_OrderType order, Integer m, Integer n, Integer kl,
               Integer ku, const double ab[], Integer pdab, double r[], double c[],
               double *rowcnd, double *colcnd, double *amax, NagError *fail)

3 Description
nag_dgbequ (f07bfc) computes the diagonal scaling matrices. The diagonal scaling matrices are chosen to try to make the elements of largest absolute value in each row and column of the matrix $B$ given by

$$B = D_R A D_C$$

have absolute value 1. The diagonal elements of $D_R$ and $D_C$ are restricted to lie in the safe range $(\delta, 1/\delta)$, where $\delta$ is the value returned by function nag_real_safe_small_number (X02AMC). Use of these scaling factors is not guaranteed to reduce the condition number of $A$ but works well in practice.

4 References
None.

5 Arguments
1: order – Nag_OrderType
   
   On entry: the order argument specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by order = Nag_RowMajor. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.
   
   Constraint: order = Nag_RowMajor or Nag_ColMajor.

2: m – Integer
   
   On entry: $m$, the number of rows of the matrix $A$.
   
   Constraint: $m \geq 0$.

3: n – Integer
   
   On entry: $n$, the number of columns of the matrix $A$.
   
   Constraint: $n \geq 0$.

4: kl – Integer
   
   On entry: $k_l$, the number of subdiagonals of the matrix $A$.
   
   Constraint: $kl \geq 0$. 

Mark 25
5: ku – Integer  
    On entry: $k_u$, the number of superdiagonals of the matrix $A$.  
    Constraint: $k_u \geq 0$.

6: ab[dim] – const double  
    Note: the dimension, $dim$, of the array $ab$ must be at least  
        $\max(1, pdab \times n)$ when $\text{order} = \text{Nag\_ColMajor}$;  
        $\max(1, m \times pdab)$ when $\text{order} = \text{Nag\_RowMajor}$.  
    On entry: the $m$ by $n$ band matrix $A$ whose scaling factors are to be computed.  
    This is stored as a notional two-dimensional array with row elements or column elements stored  
    continguously. The storage of elements $A_{ij}$, for row $i = 1, \ldots, m$ and column  
    $j = \max(1, i - k_u), \ldots, \min(n, i + k_u)$, depends on the $\text{order}$ argument as follows:  
        if $\text{order} = \text{Nag\_ColMajor}$, $A_{ij}$ is stored as $ab[(j - 1) \times pdab + k_u + i - j]$;  
        if $\text{order} = \text{Nag\_RowMajor}$, $A_{ij}$ is stored as $ab[(i - 1) \times pdab + kl + j - i]$.  
    See Section 9 in nag_dgbsv (f07bac) for further details.

7: pdab – Integer  
    On entry: the stride separating row or column elements (depending on the value of $\text{order}$) of the matrix $A$ in the array $ab$.  
    Constraint: $pdab \geq kl + ku + 1$.

8: r[m] – double  
    On exit: if $\text{fail\_code} = \text{NE\_NOERROR}$ or $\text{fail\_code} = \text{NE\_MAT\_COL\_ZERO}$, $r$ contains the row  
    scale factors, the diagonal elements of $D_R$. The elements of $r$ will be positive.

9: c[n] – double  
    On exit: if $\text{fail\_code} = \text{NE\_NOERROR}$, $c$ contains the column scale factors, the diagonal elements  
    of $D_C$. The elements of $c$ will be positive.

10: rowcnd – double *  
    On exit: if $\text{fail\_code} = \text{NE\_NOERROR}$ or $\text{fail\_code} = \text{NE\_MAT\_COL\_ZERO}$, $\text{rowcnd}$ contains the ratio of the smallest value of $r[i - 1]$ to the largest value of $r[i - 1]$. If $\text{rowcnd} \geq 0.1$ and  
        $\text{amax}$ is neither too large nor too small, it is not worth scaling by $D_R$.  
    If $\text{colcnd} \geq 0.1$, it is not worth scaling by $D_C$.  

11: colcnd – double *  
    On exit: if $\text{fail\_code} = \text{NE\_NOERROR}$, $\text{colcnd}$ contains the ratio of the smallest value of $c[i - 1]$ to the largest value of $c[i - 1]$.  

12: amax – double *  
    On exit: $\max|a_{ij}|$. If $\text{amax}$ is very close to overflow or underflow, the matrix $A$ should be scaled.

13: fail – NagError *  
    The NAG error argument (see Section 3.6 in the Essential Introduction).
6 Error Indicators and Warnings

**NE_ALLOC_FAIL**
Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

**NE_BAD_PARAM**
On entry, argument `<value>` had an illegal value.

**NE_INT**
On entry, `kl = <value>`.
Constraint: `kl ≥ 0`.
On entry, `ku = <value>`.
Constraint: `ku ≥ 0`.
On entry, `m = <value>`.
Constraint: `m ≥ 0`.
On entry, `n = <value>`.
Constraint: `n ≥ 0`.
On entry, `pdab = <value>`.
Constraint: `pdab > 0`.

**NE_INT_3**
On entry, `pdab = <value>`, `kl = <value>` and `ku = <value>`.
Constraint: `pdab ≥ kl + ku + 1`.

**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.

**NE_MAT_COL_ZERO**
Column `<value>` of `A` is exactly zero.

**NE_MAT_ROW_ZERO**
Row `<value>` of `A` is exactly zero.

**NE_NO_LICENCE**
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
The computed scale factors will be close to the exact scale factors.

8 Parallelism and Performance
Not applicable.
9 Further Comments

The complex analogue of this function is nag_zgbequ (f07btc).

10 Example

This example equilibrates the band matrix \( A \) given by

\[
A = \begin{pmatrix}
-0.23 & 2.54 & -3.66 \times 10^{-10} & 0 \\
-6.98 \times 10^{10} & 2.46 \times 10^{10} & -2.73 & -2.13 \times 10^{10} \\
0 & 2.56 & 2.46 \times 10^{-10} & 4.07 \\
0 & 0 & -4.78 \times 10^{-10} & -3.82
\end{pmatrix}.
\]

Details of the scaling factors, and the scaled matrix are output.

10.1 Program Text

/* nag_dgbequ (f07bfc) Example Program.  
* Copyright 2014 Numerical Algorithms Group.  
* Mark 23, 2011.  
*/

#include <stdio.h>
#include <nag.h>
#include <nagx04.h>
#include <nag_stdlib.h>
#include <nagf07.h>
#include <nagx02.h>

int main(void)
{
    /* Scalars */
    double amax, big, colcnd, rowcnd, small;
    Integer exit_status = 0, i, j, kl, ku, n, pdab;

    /* Arrays */
    double *ab = 0, *c = 0, *r = 0;

    /* Nag Types */
    NagError fail;
    Nag_OrderType order;
    Nag_Boolean scaled = Nag_FALSE;

    #ifdef NAG_COLUMN_MAJOR
    #define AB(I, J) ab[(J-1)*pdab + ku + I - J]
    order = Nag_ColMajor;
    #else
    #define AB(I, J) ab[(I-1)*pdab + kl + J - I]
    order = Nag_RowMajor;
    #endif

    INIT_FAIL(fail);

    printf("nag_dgbequ (f07bfc) Example Program Results\n\n");
    fflush(stdout);

    /* Skip heading in data file */
    #ifdef _WIN32
    scanf_s("%*\n");
    #else
    scanf("%*\n");
    #endif

    #ifdef _WIN32
    scanf_s("%NAG_IFMT%NAG_IFMT%NAG_IFMT%*\n", &n, &kl, &ku);
    #endif

    /* etc */
# else
    scanf("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%*[\n] ", &n, &kl, &ku);
#endif
if (n < 0 || kl < 0 || ku < 0)
{
    printf("Invalid n or kl or ku\n");
    exit_status = 1;
    goto END;
}
/* Allocate arrays */
pdab = kl+ku+1;
if (!(ab = NAG_ALLOC(pdab * n, double)) ||
    !(c = NAG_ALLOC(n, double)) ||
    !(r = NAG_ALLOC(n, double)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}
/* Read the band matrix A from data file */
for (i = 1; i <= n; ++i)
    for (j = MAX(i - kl, 1); j <= MIN(i + ku, n); ++j)
#ifdef _WIN32
    scanf_s("%lf", &AB(i, j));
#else
    scanf("%lf", &AB(i, j));
#endif
#ifdef _WIN32
    scanf_s("%*[\n]");
#else
    scanf("%*[\n] ");
#endif
/* Print the matrix A using nag_band_real_mat_print (x04cec). */
fflush(stdout);
nag_band_real_mat_print(order, n, n, kl, ku, ab, pdab, "Matrix A", 0, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_band_real_mat_print (x04cec). %s\n", fail.message);
    exit_status = 1;
    goto END;
}
printf("\n");
/* Compute row and column scaling factors using nag_dgbequ (f07bfc). */
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dgbequ (f07bfc). %s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Print rowcnd, colcnd, amax and the scale factors */
printf("Rowcnd = %10.1e, colcnd = %10.1e, amax = %10.1e\n", rowcnd, colcnd, amax);
printf("Row scale factors\n");
for (i = 1; i <= n; ++i) printf("%11.2e%s", r[i-1], i%7 == 0?"\n":" ");
printf("\n\nColumn scale factors\n");
for (i = 1; i <= n; ++i) printf("%11.2e%s", c[i-1], i%7 == 0?"\n":" ");
/* Compute values close to underflow and overflow using
   * nag_real_safe_small_number (x02amc), nag_machine_precision (x02ajc) and
   * nag_real_base (x02bhc)
   */
small = nag_real_safe_small_number / nag_machine_precision * nag_real_base;
big = 1.0 / small;

Mark 25

f07bfc.5
if (colcnd < 0.1)
{
    scaled = Nag_TRUE;
    /* column scale A */
    for (j = 1; j <= n; ++j)
        for (i = MAX(1, j - ku); i <= MIN(n, j + kl); ++i)
            AB(i, j) *= c[j - 1];
}
if (rowcnd < 0.1 || amax < small || amax > big)
{
    /* row scale A */
    scaled = Nag_TRUE;
    for (j = 1; j <= n; ++j)
        for (i = MAX(1, j - ku); i <= MIN(n, j + kl); ++i)
            AB(i, j) *= r[i-1];
}
if (scaled)
{
    /* Print the row and column scaled matrix using
       nag_band_real_mat_print (x04cec). */
    printf("\n\n");
    fflush(stdout);
    nag_band_real_mat_print(order, n, n, kl, ku, ab, pdab, "Scaled matrix",
                            0, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_band_real_mat_print (x04cec).\n%s\n",
                fail.message);
        exit_status = 1;
        goto END;
    }
}
END:
NAG_FREE(ab);
NAG_FREE(c);
NAG_FREE(r);
return exit_status;

10.2 Program Data

nag_dgbequ (f07bfc) Example Program Data

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>-2.30e-01</td>
<td>2.54e+00</td>
<td>-3.66e-10</td>
<td></td>
</tr>
<tr>
<td>-6.98e+10</td>
<td>2.46e+10</td>
<td>-2.73e+00</td>
<td>-2.13e+10</td>
</tr>
<tr>
<td>2.56e+00</td>
<td>2.46e-10</td>
<td>4.07e+00</td>
<td></td>
</tr>
<tr>
<td>-4.78e-10</td>
<td>-3.82e+00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Matrix A

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-2.30e-01</td>
<td>2.54e+00</td>
<td>-3.66e-10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-6.98e+10</td>
<td>2.46e+10</td>
<td>-2.73e+00</td>
<td>-2.13e+10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2.56e+00</td>
<td>2.46e-10</td>
<td>4.07e+00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-4.78e-10</td>
<td>-3.82e+00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

rowcnd = 3.6e-11, colcnd = 1.4e-10, amax = 7.0e+10

10.3 Program Results

nag_dgbequ (f07bfc) Example Program Results

Row scale factors

3.94e-01  1.43e-11  2.46e-01  2.62e-01

Column scale factors

1.00e+00  1.00e+00  6.94e+09  1.00e+00
Scaled matrix

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.0906</td>
<td>1.0000</td>
<td>-1.0000</td>
</tr>
<tr>
<td>2</td>
<td>-1.0000</td>
<td>0.3524</td>
<td>-0.2714</td>
</tr>
<tr>
<td>3</td>
<td>0.6290</td>
<td>0.4195</td>
<td>1.0000</td>
</tr>
<tr>
<td>4</td>
<td>-0.8684</td>
<td>-1.0000</td>
<td></td>
</tr>
</tbody>
</table>