NAG Library Function Document

nag_zgb_norm (f16ubc)

1 Purpose

nag_zgb_norm (f16ubc) calculates the value of the 1-norm, the $\infty$-norm, the Frobenius norm or the maximum absolute value of the elements of a complex $m$ by $n$ band matrix stored in banded packed form.

2 Specification

```c
#include <nag.h>
#include <nagf16.h>
void nag_zgb_norm (Nag_OrderType order, Nag_NormType norm, Integer m,
                   Integer n, Integer kl, Integer ku, const Complex ab[], Integer pdab,
                   double *r, NagError *fail)
```

3 Description

Given a complex $m$ by $n$ band matrix, $A$, nag_zgb_norm (f16ubc) calculates one of the values given by

\[
\|A\|_1 = \max_j \sum_{i=1}^{m} |a_{ij}| \quad \text{(the 1-norm of $A$)},
\]

\[
\|A\|_{\infty} = \max_i \sum_{j=1}^{n} |a_{ij}| \quad \text{(the $\infty$-norm of $A$)},
\]

\[
\|A\|_F = \left( \sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2} \quad \text{(the Frobenius norm of $A$)}, \quad \text{or}
\]

\[
\max_{i,j} |a_{ij}| \quad \text{(the maximum absolute element value of $A$)}.
\]

4 References


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: norm – Nag_NormType

On entry: specifies the value to be returned.

norm = Nag_OneNorm

The 1-norm.
**norm** = Nag_FrobeniusNorm
The Frobenius (or Euclidean) norm.

**norm** = Nag_InfNorm
The \( \infty \)-norm.

**norm** = Nag_MaxNorm
The value \( \max_{i,j} |a_{ij}| \) (not a norm).

*Constraint: \( \text{norm} = \text{Nag\_OneNorm, Nag\_FrobeniusNorm, Nag\_InfNorm or Nag\_MaxNorm} \).*

3: **m** – Integer
*Input*

*On entry:* \( m \), the number of rows of the matrix \( A \).

*Constraint: \( m \geq 0 \).*

4: **n** – Integer
*Input*

*On entry:* \( n \), the number of columns of the matrix \( A \).

*Constraint: \( n \geq 0 \).*

5: **kl** – Integer
*Input*

*On entry:* \( k_l \), the number of subdiagonals within the band of \( A \).

*Constraint: \( k_l \geq 0 \).*

6: **ku** – Integer
*Input*

*On entry:* \( k_u \), the number of superdiagonals within the band of \( A \).

*Constraint: \( k_u \geq 0 \).*

7: **ab[dim]** – const Complex
*Input*

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

\[
\max(1, pdab \times n) \text{ when order = Nag\_ColMajor;}
\]

\[
\max(1, m \times pdab) \text{ when order = Nag\_RowMajor.}
\]

*On entry:* the \( m \) by \( n \) band matrix \( A \).

This is stored as a notional two-dimensional array with row elements or column elements stored contiguously. The storage of elements \( A_{ij} \), for row \( i = 1, \ldots, m \) and column \( j = \max(1, i - k_l), \ldots, \min(n, i + k_u) \), depends on the order argument as follows:

- if order = Nag\_ColMajor, \( A_{ij} \) is stored as \( ab[j - 1] \times pdab + ku + i - j \);
- if order = Nag\_RowMajor, \( A_{ij} \) is stored as \( ab[i - 1] \times pdab + kl + j - i \).

8: **pdab** – Integer
*Input*

*On entry:* the stride separating row or column elements (depending on the value of order) of the matrix \( A \) in the array \( ab \).

*Constraint: \( pdab \geq kl + ku + 1 \).*

9: **r** – double *
*Output*

*On exit:* the value of the norm specified by norm.

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.
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.

NE_INT_3
On entry, pdab = ⟨value⟩, kl = ⟨value⟩, ku = ⟨value⟩.
Constraint: pdab ≥ kl + ku + 1.

NE_INTERNAL_ERROR
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_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 BLAS standard requires accurate implementations which avoid unnecessary over/underflow (see Section 2.7 of Basic Linear Algebra Subprograms Technical (BLAST) Forum (2001)).

8 Parallelism and Performance

Not applicable.

9 Further Comments

None.

10 Example

Reads in a 6 by 4 banded complex matrix $A$ with two subdiagonals and one superdiagonal, and prints the four norms of $A$. 

Mark 25
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf16.h>

int main(void)
{
    double r_one, r_inf, r_f, r_max;
    Integer ab_size, exit_status, i, j, kl, ku;
    Integer m, n, pdab;

    /* Arrays */
    Complex *ab = 0;

    /* Nag Types */
    NagError fail;
    Nag_OrderType order;

    exit_status = 0;
    INIT_FAIL(fail);

    printf("nag_zgb_norm (f16ubc) Example Program Results\n\n");

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

    /* Read the problem dimensions */
    #ifdef _WIN32
        scanf_s("%NAG_IFMT"%NAG_IFMT"%NAG_IFMT"%NAG_IFMT"%[\n ] ",
                 &m, &n, &kl, &ku);
    #else
        scanf("%NAG_IFMT"%NAG_IFMT"%NAG_IFMT"%NAG_IFMT"%[\n ] ",
                 &m, &n, &kl, &ku);
    #endif

    pdab = kl + ku + 1;
    #ifdef NAG_COLUMN_MAJOR
        ab_size = pdab*n;
    #else
        ab_size = pdab*m;
    #endif

    if (m > 0 & n > 0)
    {
        /* Allocate memory */
        if (!(ab = NAG_ALLOC(ab_size, Complex)))
        {
            printf("Allocation failure\n");
        }
    
    */}  // End of program text.
exit_status = -1;
goto END;
}

else
{
    printf("Invalid m or n\n");
    exit_status = 1;
    return exit_status;
}

/* Input matrix A. */
for (i = 1; i <= m; ++i)
{
    for (j = MAX(1, i-kl); j <= MIN(n, i+ku); ++j)
    {
        #ifdef _WIN32
            scanf_s(" ( %lf , %lf )", &AB(i, j).re, &AB(i, j).im);
        #else
            scanf(" ( %lf , %lf )", &AB(i, j).re, &AB(i, j).im);
        #endif
        #ifdef _WIN32
            scanf_s("%*[\n ]");
        #else
            scanf("%*[\n ]");
        #endif
    }
}

/* nag_zgb_norm (f16ubc). *
* calculates norm of Complex valued general band matrix. *
*
*/

nag_zgb_norm(order, Nag_OneNorm, m, n, kl, ku, ab, pdab, &r_one, &fail);
if (fail.code != NE_NOERROR) goto GB_FAIL;

nag_zgb_norm(order, Nag_InfNorm, m, n, kl, ku, ab, pdab, &r_inf, &fail);
if (fail.code != NE_NOERROR) goto GB_FAIL;

nag_zgb_norm(order, Nag_FrobeniusNorm, m, n, kl, ku, ab, pdab, &r_f, &fail);
if (fail.code != NE_NOERROR) goto GB_FAIL;

nag_zgb_norm(order, Nag_MaxNorm, m, n, kl, ku, ab, pdab, &r_max, &fail);
if (fail.code != NE_NOERROR) goto GB_FAIL;

/* Print norms of A. */
printf(" Norms of banded matrix A:

");
printf(" One norm = %7.4f\n", r_one);
printf(" Infinity norm = %7.4f\n", r_inf);
printf(" Frobenius norm = %7.4f\n", r_f);
printf(" Maximum norm = %7.4f\n", r_max);
goto END;

GB_FAIL:
printf("Error from nag_zgb_norm.\n%\s\n", fail.message);
exit_status = 1;

END:
NAG_FREE(ab);
return exit_status;
10.2 Program Data

nag_zgb_norm (f16ubc) Example Program Data

6 4 2 1 :Values of m, n, kl, ku

( 1.0, 1.0) ( 1.0, 2.0)
( 2.0, 1.0) ( 2.0, 2.0) ( 2.0, 3.0)
( 3.0, 1.0) ( 3.0, 2.0) ( 3.0, 3.0) ( 3.0, 4.0)
( 4.0, 2.0) ( 4.0, 3.0) ( 4.0, 4.0)
( 5.0, 3.0) ( 5.0, 4.0)
( 6.0, 4.0) : the end of matrix A

10.3 Program Results

nag_zgb_norm (f16ubc) Example Program Results

Norms of banded matrix A:

One norm = 24.2711
Infinity norm = 16.0105
Frobenius norm = 17.4069
Maximum norm = 7.2111