

NAG C Library Function Document

nag_dger (f16pmc)

1 Purpose

nag_dger (f16pmc) performs a rank-1 update on a real general matrix.

2 Specification

```
#include <nag.h>
#include <nagf16.h>
```

```
void nag_dger (Nag_OrderType order, Nag_ConjType conj, Integer m, Integer n,
              double alpha, const double x[], Integer incx, const double y[], Integer incy,
              double beta, double a[], Integer pda, NagError *fail)
```

3 Description

nag_dger (f16pmc) performs the rank-1 update operation

$$A \leftarrow \alpha xy^T + \beta A,$$

where A is an m by n real matrix, x is an m element real vector, y is an n element real vector, and α and β are real scalars. If m or n is equal to zero or if β is equal to one and α is equal to zero, this function returns immediately.

4 References

The BLAS Technical Forum Standard (2001) www.netlib.org/blas/blast-forum

5 Arguments

- 1: **order** – Nag_OrderType *Input*
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 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this argument.
Constraint: **order** = **Nag_RowMajor** or **Nag_ColMajor**.
- 2: **conj** – Nag_ConjType *Input*
On entry: the argument **conj** is not referenced if x and y are real vectors. It is suggested that you set **conj** = **Nag_NoConj** where the elements y_i are not conjugated.
Constraint: **conj** = **Nag_NoConj**.
- 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: **alpha** – double *Input*
On entry: the scalar α .
- 6: **x**[*dim*] – const double *Input*
Note: the dimension, *dim*, of the array **x** must be at least $\max(1, 1 + (\mathbf{n} - 1)|\mathbf{incx}|)$.
On entry: the vector x .
- 7: **incx** – Integer *Input*
On entry: the increment in the subscripts of **x** between successive elements of x .
Constraint: **incx** $\neq 0$.
- 8: **y**[*dim*] – const double *Input*
Note: the dimension, *dim*, of the array **y** must be at least $\max(1, 1 + (\mathbf{n} - 1)|\mathbf{incy}|)$.
On entry: the vector y .
- 9: **incy** – Integer *Input*
On entry: the increment in the subscripts of **y** between successive elements of y .
Constraint: **incy** $\neq 0$.
- 10: **beta** – double *Input*
On entry: the scalar β .
- 11: **a**[*dim*] – double *Input/Output*
Note: the dimension, *dim*, of the array **a** must be at least
 $\max(1, \mathbf{pda} \times \mathbf{n})$ when **order** = **Nag_ColMajor**;
 $\max(1, \mathbf{pda} \times \mathbf{m})$ when **order** = **Nag_RowMajor**.
If **order** = **Nag_ColMajor**, the (*i*,*j*)th element of the matrix A is stored in **a**[(*j* - 1) \times **pda** + *i* - 1].
If **order** = **Nag_RowMajor**, the (*i*,*j*)th element of the matrix A is stored in **a**[(*i* - 1) \times **pda** + *j* - 1].
On entry: the m by n matrix A .
On exit: the updated matrix A .
- 12: **pda** – Integer *Input*
On entry: the stride separating matrix row or column elements (depending on the value of **order**) in the array **a**.
Constraints:
if **order** = **Nag_ColMajor**, **pda** $\geq \max(1, \mathbf{m})$;
if **order** = **Nag_RowMajor**, **pda** $\geq \max(1, \mathbf{n})$.
- 13: **fail** – NagError * *Input/Output*
The NAG error argument (see Section 2.6 of the Essential Introduction).

6 Error Indicators and Warnings

NE_BAD_PARAM

On entry, argument $\langle value \rangle$ had an illegal value.

NE_INT

On entry, **inex** = $\langle value \rangle$.
 Constraint: **inex** $\neq 0$.

On entry, **incy** = $\langle value \rangle$.
 Constraint: **incy** $\neq 0$.

On entry, **m** = $\langle value \rangle$.
 Constraint: **m** ≥ 0 .

On entry, **n** = $\langle value \rangle$.
 Constraint: **n** ≥ 0 .

NE_INT_2

On entry, **pda** = $\langle value \rangle$, **m** = $\langle value \rangle$.
 Constraint: **pda** $\geq \max(1, \mathbf{m})$.

On entry, **pda** = $\langle value \rangle$, **n** = $\langle value \rangle$.
 Constraint: **pda** $\geq \max(1, \mathbf{n})$.

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

The BLAS standard requires accurate implementations which avoid unnecessary over/underflow (see Section 2.7 of The BLAS Technical Forum Standard (2001)).

8 Further Comments

The argument **conj** is not referenced in this case where x and y are real vectors.

9 Example

Perform rank-1 update of real matrix A using vectors x and y :

$$A \leftarrow A - xy^T,$$

where A is the 3 by 2 matrix given by

$$A = \begin{pmatrix} 3.0 & 2.0 \\ 3.0 & 4.0 \\ 5.0 & 9.0 \end{pmatrix},$$

$$x = (2.0, 3.0, 5.0)^T \quad \text{and} \quad y = (0.0, 1.0, 0.0)^T.$$

9.1 Program Text

```

/* nag_dger (f16pmc) Example Program.
 *
 * Copyright 2005 Numerical Algorithms Group.
 *
 * Mark 8, 2005.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf16.h>
#include <nagx04.h>

int main(void)

```

```

{
/* Scalars */
double alpha, beta;
Integer exit_status, i, incx, incy, j, m, n, pda, xlen, ylen;

/* Arrays */
double *a=0, *x=0, *y=0;

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

#ifdef NAG_COLUMN_MAJOR
#define A(I,J) a[(J-1)*pda + I - 1]
order = Nag_ColMajor;
#else
#define A(I,J) a[(I-1)*pda + J - 1]
order = Nag_RowMajor;
#endif

exit_status = 0;
conj = Nag_NoConj;
INIT_FAIL(fail);

Vprintf( "nag_dger (f16pmc) Example Program Results\n\n");

/* Skip heading in data file */
Vscanf("%*[\n] ");

/* Read the problem dimensions */
Vscanf("%ld%ld%*[\n] ", &m, &n);

/* Read scalar parameters */
Vscanf("%lf%lf%*[\n] ", &alpha, &beta);
/* Read increment parameters */
Vscanf("%ld%ld%*[\n] ", &incx, &incy);

#ifdef NAG_COLUMN_MAJOR
pda = m;
#else
pda = n;
#endif

xlen = MAX(1, 1 + (m - 1)*ABS(incx));
ylen = MAX(1, 1 + (n - 1)*ABS(incy));

if (m > 0 && n > 0)
{
/* Allocate memory */
if ( !(a = NAG_ALLOC(m*n, double)) ||
!(x = NAG_ALLOC(xlen, double)) ||
!(y = NAG_ALLOC(ylen, double)) )
{
Vprintf("Allocation failure\n");
exit_status = -1;
goto END;
}
}
else
{
Vprintf("Invalid m or n\n");
exit_status = 1;
return exit_status;
}

/* Input matrix A and vectors x and y */
for (i = 1; i <= m; ++i)

```

```

    {
        for (j = 1; j <= n; ++j)
            Vscanf("%lf", &A(i,j));
        Vscanf("%*[\n] ");
    }
for (i = 0; i < xlen; ++i)
    Vscanf("%lf%*[\n] ", &x[i]);
for (i = 0; i < ylen; ++i)
    Vscanf("%lf%*[\n] ", &y[i]);

/* nag_dger(f16pmc).
 * Rank one update of real matrix.
 *
 */
nag_dger(order, conj, m, n, alpha, x, incx, y, incy, beta,
         a, pda, &fail);
if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from nag_dger.\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

/* Print updated matrix A */
/* nag_gen_real_mat_print (x04cac).
 * Print real general matrix (easy-to-use)
 */
nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, m,
                      n, a, pda, "Updated Matrix A", 0, &fail);
if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from nag_gen_real_mat_print (x04cac).\n%s\n",
                fail.message);
        exit_status = 1;
        goto END;
    }

END:
if (a) NAG_FREE(a);
if (x) NAG_FREE(x);
if (y) NAG_FREE(y);

return exit_status;
}

```

9.2 Program Data

```

nag_dger (f16pmc) Example Program Data
 3 2           : m, n the dimensions of matrix A
-1.0 1.0      : alpha, beta
 1 2           : incx, incy
 3.0 2.0
 3.0 4.0
 5.0 9.0      : the end of matrix A
 2.0
 3.0
 5.0          : the end of vector x
 1.0
 0.0
 1.0
 0.0          : the end of vector y

```

9.3 Program Results

nag_dger (f16pmc) Example Program Results

Updated Matrix A

```

      1      2
1    1.0000  0.0000

```

2	0.0000	1.0000
3	0.0000	4.0000
