NAG Library Function Document

nag_dger (f16pmc)

1 Purpose

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

2 Specification

```c
#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 \( \alpha \) and \( \beta \) are real scalars. If \( m \) or \( n \) is equal to zero or if \( \beta \) is equal to one and \( \alpha \) is equal to zero, this function returns immediately.

4 References


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 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: `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:  \textbf{n} – Integer  \hspace{1cm} \textit{Input}
\textit{On entry:} \textit{n}, the number of columns of the matrix \textit{A}.
\textit{Constraint:} \textit{n} \geq 0.

5:  \textbf{alpha} – double  \hspace{1cm} \textit{Input}
\textit{On entry:} the scalar \textit{\alpha}.

6:  \textbf{x}[\textit{dim}] – const double  \hspace{1cm} \textit{Input}
\textit{Note:} the dimension, \textit{dim}, of the array \textit{x} must be at least \textit{max(1, 1 + (n - 1) |\textit{incx}|)}.
\textit{On entry:} the vector \textit{x}.

7:  \textbf{incx} – Integer  \hspace{1cm} \textit{Input}
\textit{On entry:} the increment in the subscripts of \textit{x} between successive elements of \textit{x}.
\textit{Constraint:} \textit{incx} \neq 0.

8:  \textbf{y}[\textit{dim}] – const double  \hspace{1cm} \textit{Input}
\textit{Note:} the dimension, \textit{dim}, of the array \textit{y} must be at least \textit{max(1, 1 + (n - 1) |\textit{incy}|)}.
\textit{On entry:} the vector \textit{y}.

9:  \textbf{incy} – Integer  \hspace{1cm} \textit{Input}
\textit{On entry:} the increment in the subscripts of \textit{y} between successive elements of \textit{y}.
\textit{Constraint:} \textit{incy} \neq 0.

10:  \textbf{beta} – double  \hspace{1cm} \textit{Input}
\textit{On entry:} the scalar \textit{\beta}.

11:  \textbf{a}[\textit{dim}] – double  \hspace{1cm} \textit{Input/Output}
\textit{Note:} the dimension, \textit{dim}, of the array \textit{a} must be at least
\textit{\textit{max(1, pda} \times \textit{n}) when } \textit{order} = \textit{Nag\_ColMajor};
\textit{\textit{max(1, m} \times \textit{pda}) when } \textit{order} = \textit{Nag\_RowMajor}.
\textit{If } \textit{order} = \textit{Nag\_ColMajor}, \textit{A}_{ij} is stored in \textit{a}_{(j - 1) \times \textit{pda} + i - 1]}.
\textit{If } \textit{order} = \textit{Nag\_RowMajor}, \textit{A}_{ij} is stored in \textit{a}_{(i - 1) \times \textit{pda} + j - 1]}.
\textit{On entry:} the \textit{m} by \textit{n} matrix \textit{A}.
\textit{On exit:} the updated matrix \textit{A}.

12:  \textbf{pda} – Integer  \hspace{1cm} \textit{Input}
\textit{On entry:} the stride separating row or column elements (depending on the value of \textit{order}) in the array \textit{a}.
\textit{Constraints:}
\textit{if } \textit{order} = \textit{Nag\_ColMajor}, \textit{pda} \geq \textit{max(1, m)};
\textit{if } \textit{order} = \textit{Nag\_RowMajor}, \textit{pda} \geq \textit{n}.

13:  fail – NagError *  \hspace{1cm} \textit{Input/Output}
\textit{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 \langle value \rangle had an illegal value.

NE_INT
On entry, \texttt{incx} = \langle value \rangle.
Constraint: \texttt{incx} \neq 0.
On entry, \texttt{incy} = \langle value \rangle.
Constraint: \texttt{incy} \neq 0.
On entry, \texttt{m} = \langle value \rangle.
Constraint: \texttt{m} \geq 0.
On entry, \texttt{n} = \langle value \rangle.
Constraint: \texttt{n} \geq 0.

NE_INT_2
On entry, \texttt{pda} = \langle value \rangle, \texttt{m} = \langle value \rangle.
Constraint: \texttt{pda} \geq \text{max}(1, \texttt{m}).
On entry, \texttt{pda} = \langle value \rangle and \texttt{n} = \langle value \rangle.
Constraint: \texttt{pda} \geq \texttt{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 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_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
The argument \texttt{conj} is not referenced in this case where \texttt{x} and \texttt{y} are real vectors.
10 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

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

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

10.1 Program Text

/
/* nag_dger (f16pmc) Example Program.
 * Copyright 2014 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);

    printf("nag_dger (f16pmc) 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("%*[\n] ");
    #else
    scanf("%*[\n] ");
    #endif

}
```c
scanf_s("%"NAG_IFMT"%"NAG_IFMT"%[^\n] ", &m, &n);
#else
    scanf("%"NAG_IFMT"%"NAG_IFMT"%[^\n] ", &m, &n);
#endif

/* Read scalar parameters */
#ifdef _WIN32
    scanf_s("%lf%lf%[^\n] ", &alpha, &beta);
#else
    scanf("%lf%lf%[^\n] ", &alpha, &beta);
#endif

/* Read increment parameters */
#ifdef _WIN32
    scanf_s("%"NAG_IFMT"%"NAG_IFMT"%[^\n] ", &incx, &incy);
#else
    scanf("%"NAG_IFMT"%"NAG_IFMT"%[^\n] ", &incx, &incy);
#endif

#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)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
}
else
{
    printf("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)
    {
        #ifdef _WIN32
            scanf_s("%lf", &A(i, j));
        #else
            scanf("%lf", &A(i, j));
        #endif
        #ifdef _WIN32
            scanf_s("%[^\n] ");
        #else
            scanf("%[^\n] ");
        #endif
    }
    for (i = 0; i < xlen; ++i)
    {
        #ifdef _WIN32
            scanf_s("%lf%[^\n] ", &x[i]);
        #else
            scanf("%lf%[^\n] ", &x[i]);
        #endif
    }
    for (i = 0; i < ylen; ++i)
    {
        #ifdef _WIN32
            scanf_s("%lf%[^\n] ", &y[i]);
        #else
            scanf("%lf%[^\n] ", &y[i]);
        #endif
    }
```

The code snippet is from the NAG Interface to BLAS, specifically the function f16pmc. The code reads scalar parameters, increment parameters, and allocates memory. It then reads the matrix A and vectors x and y.
```c
#include <stdio.h>

# ifndef _WIN32
    #else
        scanf("%lf%*[^
", &y[i]);
    #endif

/* 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)
{
    printf("Error from nag_dger.
%s
", fail.message);
    exit_status = 1;
    goto END;
}

/* Print updated matrix A */
/* nag_gen_real_mat_print (x04cac).  
 * Print real general matrix (easy-to-use)  
 */
fflush(stdout);
nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, m,  
                       n, a, pda, "Updated Matrix A", 0, &fail);

if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_real_mat_print (x04cac).
%s
", fail.message);
    exit_status = 1;
    goto END;
}

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

return exit_status;
}

10.2 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

10.3 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
```