NAG Library Function Document

nag_daxpby (f16ecc)

1 Purpose

nag_daxpby (f16ecc) computes the sum of two scaled vectors, for real vectors and scalars.

nag_daxpby (f16ecc) performs the operation

$$y \leftarrow \alpha x + \beta y.$$

2 Specification

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

```
void nag_daxpby (Integer n, double alpha, const double x[], Integer incx,
     double beta, const double y[], Integer incy, NagError *fail)
```

3 Description

nag_daxpby (f16ecc) performs the operation

$$y \leftarrow \alpha x + \beta y$$

where x and y are n-element real vectors, and α and β real scalars. If n is equal to zero, or if α is equal to zero and β is equal to 1, this function returns immediately.

4 References

Basic Linear Algebra Subprograms Technical (BLAST) Forum (2001) *Basic Linear Algebra Subprograms Technical (BLAST) Forum Standard* University of Tennessee, Knoxville, Tennessee http://www.netlib.org/blas/blast-forum/blas-report.pdf

5 Arguments

1: **n** – Integer *Input*

On entry: n, the number of elements in x and y.

Constraint: $\mathbf{n} \geq 0$.

2: **alpha** – double *Input*

On entry: the scalar α.

3: **x**[*dim*] – const double *Input*

Note: the dimension, *dim*, of the array **x** must be at least $\max(1, 1 + (\mathbf{n} - 1) \times |\mathbf{incx}|)$.

On entry: the n-element vector x.

If $\mathbf{incx} > 0$, x_i must be stored in $\mathbf{x}[(i - 1) \times |\mathbf{incx}|]$, for $i = 1, 2, \ldots, \mathbf{n}$.

If $\mathbf{incx} < 0$, x_i must be stored in $\mathbf{x}[(\mathbf{n} - i) \times |\mathbf{incx}| - 2]$, for $i = 1, 2, \ldots, \mathbf{n}$.

Intermediate elements of **x** are not referenced.

4: **incx** – Integer *Input*

On entry: the increment in the subscripts of **x** between successive elements of x.

Constraint: **incx** $\neq 0$.

5: **beta** – double *Input*

On entry: the scalar β.

6: **y**[*dim*] – const double *Input*

Note: the dimension, *dim*, of the array **y** must be at least $\max(1, 1 + (\mathbf{n} - 1) \times |\mathbf{incy}|)$.

On entry: the n-element vector y.

If **incy** > 0, y_i must be stored in $\mathbf{y}[1 + (i - 1) \times \mathbf{incy}]$, for $i = 1, 2, \dots, \mathbf{n}$.

If **incy** < 0, y_i must be stored in $\mathbf{y}[1 - (\mathbf{n} - i) \times \mathbf{incy}]$, for $i = 1, 2, \dots, \mathbf{n}$.

Intermediate elements of **y** are not referenced.

On exit: the updated vector y stored in the array elements used to supply the original vector y.

Intermediate elements of **y** are unchanged.

7: **incy** – Integer *Input*

On entry: the increment in the subscripts of **y** between successive elements of y.

Constraint: **incy** $\neq 0$.

8: **fail** – NagError * *Input/Output*

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

6 Error Indicators and Warnings

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INT

On entry, **incx** = ⟨*value*⟩.
Constraint: **incx** $\neq 0$.

On entry, **incy** = ⟨*value*⟩.
Constraint: **incy** $\neq 0$.

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 0.

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

nag_daxpby (f16ecc) is not threaded by NAG in any implementation.

nag_daxpby (f16ecc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

None.

10 Example

This example computes the result of a scaled vector accumulation for

$$\alpha = 3, \qquad x = (-4, 2.1, 3.7, 4.5, -6)^{\mathrm{T}},$$
$$\beta = -1, \qquad y = (-3, -2.4, 6.4, -5, -5.1)^{\mathrm{T}}.$$

See Section 10 in nag_real_banded_sparse_eigensystem_sol (f12agc) and nag_real_symm_banded_sparse_eigensystem_sol (f12fgc).

10.1 Program Text

```
/* nag_daxpby (f16ecc) Example Program.
 *
 * Copyright 2013 Numerical Algorithms Group.
 *
 * Mark 24, 2013.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf16.h>
int main(void)
{
  /* Scalars */
  Integer  exit_status, i, incx, incy, n, xlen, ylen;
  double   alpha, beta;
  /* Arrays */
  double   *x = 0, *y = 0;
  /* Nag Types */
  NagError fail;

  exit_status = 0;
  INIT_FAIL(fail);

  printf("nag_daxpby (f16ecc) Example Program Results\n\n");

  /* Skip heading in data file */
  scanf("%*[^\n] ");
  /* Read number of elements */
  scanf("%ld%*[^\n] ", &n);
  /* Read increments */
  scanf("%ld%ld%*[^\n] ", &incx, &incy);
  /* Read factors alpha and beta */
  scanf("%lf%lf%*[^\n] ", &alpha, &beta);

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

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

  /* Input vector x */
      for (i = 0; i < xlen ; i = i + abs(incx))
        scanf("%lf", &x[i]);
      scanf("%*[^\n] ");

  /* Input vector y */
  for (i = 0; i < ylen; i = i + abs(incy))
    scanf("%lf", &y[i]);
  scanf("%*[^\n] ");

  /* nag_daxpby (f16ecc).
   * Performs y := alpha*x + beta*y  */
  nag_daxpby(n, alpha, x, incx, beta, y, incy, &fail);

  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_daxpby (f16ecc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  /* Print the result */
  printf("Result of the scaled vector accumulation is\n");
  printf("y = (");

  for (i = 0; i < ylen - 1; i = i + abs(incy))
    printf("%9.4f, ", y[i]);
  printf("%9.4f)\n", y[ylen - 1]);

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

  return exit_status;
}
```

10.2 Program Data

```
nag_daxpby (f16ecc) Example Program Data
   5                                   : n
  -1    -1                             : incx and incy
   3.0  -1.0                           : alpha and beta
  -4.0   2.1   3.7   4.5   -6.0        : x
  -3.   -2.4   6.4  -5.0   -5.1        : y
```

10.3 Program Results

```
nag_daxpby (f16ecc) Example Program Results

Result of the scaled vector accumulation is
y = (  -9.0000,    8.7000,    4.7000,   18.5000,  -12.9000)
```