# NAG Library Function Document

# nag_dwaxpby (f16ehc)

## 1 Purpose

nag_dwaxpby (f16ehc) computes the sum of two scaled vectors, preserving input, for real scalars and vectors.

## 2 Specification

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

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

## 3 Description

nag_dwaxpby (f16ehc) performs the operation

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

where $x$ and $y$ are $n$-element real vectors, and $\alpha$ and $\beta$ are real scalars.

## 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 $x$, $y$ and $w$.

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

2: **alpha** – double *Input*

*On entry*: the scalar $\alpha$.

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*: $\mathbf{incx} \neq 0$.

5: **beta** – double *Input*

*On entry*: the scalar $\beta$.

6: $\mathbf{y}[dim]$ – 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 $\mathbf{incy} > 0$, $y_i$ must be stored in $\mathbf{y}[1 + (i - 1) \times \mathbf{incy} - 1]$, for $i = 1, 2, \ldots, \mathbf{n}$.

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

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

7: **incy** – Integer *Input*

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

*Constraint*: $\mathbf{incy} \neq 0$.

8: $\mathbf{w}[dim]$ – double *Output*

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

*On exit*: the $n$-element vector $w$.

If $\mathbf{incw} > 0$, $w_i$ is in $\mathbf{w}[1 + (i - 1) \times \mathbf{incw} - 1]$, for $i = 1, 2, \ldots, \mathbf{n}$.

If $\mathbf{incw} < 0$, $w_i$ is in $\mathbf{w}[1 + (\mathbf{n} - i) \times \mathbf{incw} - 1]$, for $i = 1, 2, \ldots, \mathbf{n}$.

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

9: **incw** – Integer *Input*

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

*Constraint*: $\mathbf{incw} \neq 0$.

10: **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 $\langle value \rangle$ had an illegal value.

### NE_INT

On entry, $\mathbf{incw} = \langle value \rangle$.
Constraint: $\mathbf{incw} \neq 0$.

On entry, $\mathbf{incx} = \langle value \rangle$.
Constraint: $\mathbf{incx} \neq 0$.

On entry, $\mathbf{incy} = \langle value \rangle$.
Constraint: $\mathbf{incy} \neq 0$.

On entry, $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{n} \geq 0$.

## 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

Not applicable.

## 9 Further Comments

None.

## 10 Example

This example computes the result of a scaled vector accumulation for

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

### 10.1 Program Text

```
/* nag_dwaxpby (f16ehc) Example Program.
 *
 * Copyright 2005 Numerical Algorithms Group.
 *
 * Mark 9, 2009.
 */

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

int main(void)
{
  /* Scalars */
  Integer  exit_status, i, incw, incx, incy, n, wlen, xlen, ylen;
  double   alpha, beta;
  /* Arrays */
  double   *w = 0, *x = 0, *y = 0;
  /* Nag Types */
  NagError fail;

  exit_status = 0;
  INIT_FAIL(fail);

  printf("nag_dwaxpby (f16ehc) 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%ld%*[^\n] ", &incx, &incy, &incw);
  /* Read factors alpha and beta */
  scanf("%lf%lf%*[^\n] ", &alpha, &beta);

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

  if (n > 0)
    {
      /* Allocate memory */
      if (!(w = NAG_ALLOC(wlen, double)) ||
          !(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 + incx)
    scanf("%lf", &x[i]);
  scanf("%*[^\n] ");

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

  /* nag_dwaxpby (f16ehc).
   * Performs w := alpha*x + beta*y  */
  nag_dwaxpby(n, alpha, x, incx, beta, y, incy, w, incw, &fail);

  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_dwaxpby (f16ehc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

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

  for (i = 0; i < wlen - 1; i = i + incw)
    printf("%9.4f, ", w[i]);
  printf("%9.4f)\n", w[wlen - 1]);

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

  return exit_status;
}
```

## 10.2 Program Data

```
nag_dwaxpby (f16ehc) Example Program Data
  5                                                        : n
  1   1   1                                                : incx, incy and incw
  3.0   -1.0                                               : alpha and beta
  -4.0   2.1   3.7   4.5   -6.0                            : Array x
  -3.0   -2.4   6.4   -5.0   -5.1                          : Array y
```

## 10.3 Program Results

```
nag_dwaxpby (f16ehc) Example Program Results

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