NAG Library Function Document

nag_zwaxpby (f16ghc)

1 Purpose

nag_zwaxpby (f16ghc) computes the sum of two scaled vectors, preserving input, for complex scalars and vectors.

2 Specification

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

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

3 Description

nag_zwaxpby (f16ghc) performs the operation

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

where x and y are n-element complex vectors, and α and β are complex 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** – Complex *Input*

On entry: the scalar α.

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

On entry: the scalar β.

6: **y**[*dim*] – const Complex *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: **w**[*dim*] – Complex *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 ⟨*value*⟩ had an illegal value.

NE_INT

On entry, **incw** = ⟨*value*⟩.
Constraint: $\mathbf{incw} \neq 0$.

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

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

On entry, **n** = ⟨*value*⟩.
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

$$\alpha = 3 + 2i, \qquad x = (-4 + 2.1i, 3.7 + 4.5i, -6 + 1.2i)^{\mathrm{T}},$$
$$\beta = -i, \qquad y = (-3 - 2.4i, 6.4 - 5i, -5.1)^{\mathrm{T}}.$$

10.1 Program Text

```
/* nag_zwaxpby (f16ghc) 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;
  Complex  alpha, beta;
  /* Arrays */
  Complex  *w = 0, *x = 0, *y = 0;
  /* Nag Types */
  NagError fail;

  exit_status = 0;
  INIT_FAIL(fail);

  printf("nag_zwaxpby (f16ghc) 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 ) ", &alpha.re, &alpha.im);
  scanf(" ( %lf , %lf ) %*[^\n] ", &beta.re, &beta.im);

  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, Complex)) ||
          !(x = NAG_ALLOC(xlen, Complex)) ||
          !(y = NAG_ALLOC(ylen, Complex)))
        {
          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 , %lf ) ", &x[i].re, &x[i].im);
  scanf("%*[^\n] ");
  /* Input vector y */
  for (i = 0; i < ylen; i = i + incy)
    scanf(" ( %lf , %lf ) ", &y[i].re, &y[i].im);
  scanf("%*[^\n] ");

  /* nag_zwaxpby (f16ghc).
   * Performs w := alpha*x + beta*y  */
  nag_zwaxpby(n, alpha, x, incx, beta, y, incy, w, incw, &fail);

  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_zwaxpby (f16ghc).\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,%9.4f), ", w[i].re, w[i].im);
  printf("(%9.4f,%9.4f) )\n", w[wlen - 1].re, w[wlen - 1].im);

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

  return exit_status;
}
```

10.2 Program Data

```
nag_zwaxpby (f16ghc) Example Program Data
  3                                                     : n
  1   1   1                                             : incx, incy and incw
  ( 3., 2.)   ( 0.,-1.)                                 : alpha and beta
  (-4., 2.1)   ( 3.7, 4.5)   (-6., 1.2)                 : Array x
  (-3.,-2.4)   ( 6.4,-5.)   (-5.1,0.)                   : Array y
```

10.3 Program Results

```
nag_zwaxpby (f16ghc) Example Program Results

Result of the scaled vector addition is
w = ( ( -18.6000,   1.3000), (  -2.9000,  14.5000), ( -20.4000,  -3.3000) )
```