NAG Library Function Document

nag_zaxpby (f16gcc)

1 Purpose

nag_zaxpby (f16gcc) computes the sum of two scaled vectors, for complex scalars and vectors.

2 Specification

```c
#include <nag.h>
#include <nagf16.h>
void nag_zaxpby (Integer n, Complex alpha, const Complex x[], Integer incx,
                Complex beta, Complex y[], Integer incy, NagError *fail)
```

3 Description

nag_zaxpby (f16gcc) performs the operation

\[ y \leftarrow \alpha x + \beta y, \]

where \( x \) and \( y \) are \( n \)-element complex vectors, and \( \alpha \) and \( \beta \) are complex scalars. If \( n \) is equal to zero, or if \( \alpha \) is equal to zero and \( \beta \) is equal to 1, this function returns immediately.

4 References


5 Arguments

1: \( n \) – Integer

\( n \), the number of elements in \( x \) and \( y \).

\textit{Input} \hspace{1cm} \textit{Constraint:} \( n \geq 0 \).

2: \( \alpha \) – Complex

\( \alpha \), the scalar \( \alpha \).

\textit{Input}

3: \( x[dim] \) – const Complex

The dimension, \( dim \), of the array \( x \) must be at least \( \max(1,1+(n-1) \times |\text{incx}|) \).

\( x \), the \( n \)-element vector.

\textit{Input}

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

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

Intermediate elements of \( x \) are not referenced.

4: \( \text{incx} \) – Integer

\( \text{incx} \), the increment in the subscripts of \( x \) between successive elements of \( x \).

\textit{Input} \hspace{1cm} \textit{Constraint:} \( \text{incx} \neq 0 \).
5: \textbf{beta} – Complex \hspace{1cm} \textit{Input}

\textit{On entry:} the scalar $\beta$.

6: $y[\text{dim}]$ – Complex \hspace{1cm} \textit{Input/Output}

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

\textit{On entry:} the $n$-element vector $y$.

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

Intermediate elements of $y$ are not referenced.

\textit{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: $\text{incy}$ – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} the increment in the subscripts of $y$ between successive elements of $y$.

\textit{Constraint:} $\text{incy} \neq 0$.

8: $\text{fail}$ – NagError * \hspace{1cm} \textit{Input/Output}

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

6 \hspace{0.5cm} \textbf{Error Indicators and Warnings}

\textbf{NE_ALLOC_FAIL}

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

\textbf{NE_BAD_PARAM}

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

\textbf{NE_INT}

On entry, $\text{inex} = \langle\text{value}\rangle$.
 CONSTRAINT: $\text{inex} \neq 0$.

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

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

\textbf{NE_INTERNAL_ERROR}

An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in the Essential Introduction for further information.

\textbf{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 \hspace{0.5cm} \textbf{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

\texttt{nag\_zaxpby} (f16gcc) is not threaded by NAG in any implementation.

\texttt{nag\_zaxpby} (f16gcc) 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 X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also 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

\[ \alpha = 3 + 2i, \quad x = (-4 + 2.1i, 3.7 + 4.5i, -6 + 1.2i)^T, \]
\[ \beta = -i, \quad y = (-3 - 2.4i, 6.4 - 5i, -5.1)^T. \]

10.1 Program Text

\begin{verbatim}
/* nag_zaxpby (f16gcc) Example Program. */
* Copyright 2014 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;
  Complex alpha, beta;
  /* Arrays */
  Complex *x = 0, *y = 0;
  /* Nag Types */
  NagError fail;

  exit_status = 0;
  INIT_FAIL(fail);

  printf("nag_zaxpby (f16gcc) Example Program Results\n\n");

  /* Skip heading in data file */
  #ifdef _WIN32
    scanf_s("%*[\n ] ");
  #else
    scanf("%*[\n ] ");
  #endif
  /* Read number of elements */
  #ifdef _WIN32
    scanf_s("%"NAG_IFMT"%*[\n ] ", &n);
  #else
    scanf("%"NAG_IFMT"%*[\n ] ", &n);
  #endif
  /* Read increments */
  #ifdef _WIN32
    scanf_s("%"NAG_IFMT"%"NAG_IFMT"%*[\n ] ", &incx, &incy);
  #else

\end{verbatim}
In the given code snippet, the following operations are performed:

1. **Read factors alpha and beta**
   - The code includes conditional compilation based on `_WIN32` to use `scanf_s` for Windows platforms.
   - It reads the real and imaginary parts of `alpha` and `beta` using `scanf`.

2. **Allocate memory**
   - Depending on the value of `n`, memory is allocated for the vectors `x` and `y`.
   - If allocation fails, an error message is printed, and execution stops.

3. **Input vectors x and y**
   - The code reads the elements of vectors `x` and `y` from the input.
   - It uses `scanf` to read each element in the vectors.

4. **Perform scaled vector accumulation**
   - The function `nag_zaxpby` is called with the appropriate parameters.
   - The result is printed if there are no errors.

The code snippet is part of the NAG Library Manual for FL6GCC Mark 25, which is a collection of numerical algorithms for use in scientific computing.

The main function is called `nag_zaxpby` which performs the operation `y := alpha*x + beta*y`.

The code also handles errors gracefully by printing error messages and exiting with appropriate status codes.
printf("y = (\n");

for (i = 0; i < ylen; i = i + abs(incy))
{
    printf(" ( %9.4f, %9.4f )", y[i].re, y[i].im);
    (i != ylen - 1) ? printf(",") : printf(" )");
    printf("\n");
}

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

return exit_status;
}

10.2 Program Data

nag_zaxpby (f16gcc) Example Program Data

\begin{verbatim}
3 1
( 3.0, 2.0) ( 0.0,-1.0)
(-4.0, 2.1) ( 3.7, 4.5) (-6.0, 1.2)
(-3.0,-2.4) ( 6.4,-5.0) (-5.1, 0.0)
\end{verbatim}

: n
: incx and incy
: alpha and beta
: x
: y

10.3 Program Results

nag_zaxpby (f16gcc) Example Program Results

Result of the scaled vector accumulation is
y = ( 
( -18.6000, 1.3000 ),
( -2.9000, 14.5000 ),
( -20.4000, -3.3000 ) )