

NAG Library Function Document

nag_zsum (f16glc)

1 Purpose

nag_zsum (f16glc) sums the elements of a complex vector.

2 Specification

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

```
Complex nag_zsum (Integer n, const Complex x[], Integer incx, NagError *fail)
```

3 Description

nag_zsum (f16glc) returns the sum

$$x_1 + x_2 + \cdots + x_n$$

of the elements of an n -element complex vector x .

If $\mathbf{n} = 0$ on entry, nag_zsum (f16glc) returns the value $0 + 0i$.

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 .
Constraint: $\mathbf{n} \geq 0$.
- 2: **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 $i = 1, 2, \dots, \mathbf{n}$.
 If $\mathbf{incx} < 0$, x_i must be stored in $\mathbf{x}[(\mathbf{n} - i) \times |\mathbf{incx}|]$, for $i = 1, 2, \dots, \mathbf{n}$.
 Intermediate elements of **x** are not referenced. If $\mathbf{n} = 0$, **x** is not referenced and may be **NULL**.
- 3: **incx** – Integer *Input*
On entry: the increment in the subscripts of **x** between successive elements of x .
Constraint: $\mathbf{incx} \neq 0$.
- 4: **fail** – NagError * *Input/Output*
 The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.

See Section 3.2.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

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

NE_INT

On entry, $\mathbf{incx} = \langle value \rangle$.

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

On entry, $\mathbf{n} = \langle value \rangle$.

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

NE_INTERNAL_ERROR

An unexpected error has been triggered by this function. Please contact NAG.

See Section 3.6.6 in How to Use the NAG Library and its Documentation 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 How to Use the NAG Library and its Documentation 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

nag_zsum (f16glc) is not threaded in any implementation.

9 Further Comments

None.

10 Example

This example computes the sum of the elements of

$$x = (1.1 + 10.2i, 11.5 - 2.7i, 9.2)^T.$$

10.1 Program Text

```

/* nag_zsum (f16glc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

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

```

```

#include <nagf16.h>

int main(void)
{
    /* Scalars */
    Integer exit_status, i, incx, ix, n;
    Complex sumval;
    /* Arrays */
    Complex *x = 0;
    /* Nag Types */
    NagError fail;

    exit_status = 0;
    INIT_FAIL(fail);

    printf("nag_zsum (f16glc) Example Program Results\n\n");

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

    if (n > 0) {
        /* Allocate memory */
        if (!(x = NAG_ALLOC(MAX(1, 1 + (n - 1) * ABS(incx)), Complex)))
        {
            printf("Allocation failure\n");
            exit_status = -1;
            goto END;
        }
    }
    else {
        printf("Invalid n\n");
        exit_status = 1;
        goto END;
    }

    /* Read the vector x and store forwards or backwards
     * as determined by incx. */
    for (i = 0, ix = (incx > 0 ? 0 : (1-n)*incx); i < n; i++, ix += incx)
#ifdef _WIN32
        scanf_s(" ( %lf , %lf ) ", &x[ix].re, &x[ix].im);
#else
        scanf(" ( %lf , %lf ) ", &x[ix].re, &x[ix].im);
#endif
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif

    /* nag_zsum (f16glc).
     * Sum elements of a vector of Complexes */
    sumval = nag_zsum(n, x, incx, &fail);

    if (fail.code != NE_NOERROR) {
        printf("Error from nag_zsum (f16glc).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

    /* Print the result. */
    printf("Sum of elements of x is (%9.5f,%9.5f)\n", sumval.re, sumval.im);
}

```

```
END:
  NAG_FREE(x);

  return exit_status;
}
```

10.2 Program Data

```
nag_zsum (f16glc) Example Program Data
  3  1                                     : n and incx
  ( 1.1, 10.2)  ( 11.5,-2.7)  ( 9.2, 0.)   : Vector x
```

10.3 Program Results

```
nag_zsum (f16glc) Example Program Results

Sum of elements of x is ( 21.80000,  7.50000)
```
