

NAG Library Function Document

nag_zamin_val (f16jtc)

1 Purpose

nag_zamin_val (f16jtc) computes, with respect to absolute value, the smallest component of a complex vector, along with the index of that component.

2 Specification

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

void nag_zamin_val (Integer n, const Complex x[], Integer incx, Integer *k,
                   double *r, NagError *fail)
```

3 Description

nag_zamin_val (f16jtc) computes, with respect to absolute value, the smallest component, r , of an n -element complex vector x , and determines the smallest index, k , such that

$$r = |\operatorname{Re} x_k| + |\operatorname{Im} x_k| = \min_j |\operatorname{Re} x_j| + |\operatorname{Im} x_j|.$$

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: $n \geq 0$.
- 2: **x[dim]** – const Complex *Input*
Note: the dimension, dim , of the array **x** must be at least $\max(1, 1 + (n - 1) \times |\mathbf{incx}|)$.
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 $i = 1, 2, \dots, n$.
 If $\mathbf{incx} < 0$, x_i must be stored in $\mathbf{x}[(n - i) \times |\mathbf{incx}|]$, for $i = 1, 2, \dots, n$.
 Intermediate elements of **x** are not referenced. If $n = 0$, **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: **k** – Integer * *Output*
On exit: k , the index, from the set $\{0, 1, \dots, n - 1\}$, of the smallest component of x with respect to absolute value. If $n = 0$ on input then **k** is returned as -1 .

- 5: **r** – double * *Output*
On exit: *r*, the smallest component of *x* with respect to absolute value. If **n** = 0 on input then **r** is returned as 0.0.
- 6: **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 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

On entry, argument *<value>* had an illegal value.

NE_INT

On entry, **incx** = *<value>*.
 Constraint: **incx** ≠ 0.

On entry, **n** = *<value>*.
 Constraint: **n** ≥ 0.

NE_INTERNAL_ERROR

An unexpected error has been triggered by this function. Please contact NAG.
 See Section 2.7.6 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 2.7.5 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_zamin_val (f16jtc) is not threaded in any implementation.

9 Further Comments

None.

10 Example

This example computes the smallest component with respect to absolute value and index of that component for the vector

$$x = (-4 + 2.1i, 3.7 + 4.5i, -6 + 1.2i)^T.$$

10.1 Program Text

```

/* nag_zamin_val (f16jtc) 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, k, n;
    double r;
    /* Arrays */
    Complex *x = 0;
    /* Nag Types */
    NagError fail;

    exit_status = 0;
    INIT_FAIL(fail);

    printf("nag_zamin_val (f16jtc) 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_zamin_val (f16jtc).
 * Get absolutely minimum value (r) and location of that value (k)
 * of Complex array */
nag_zamin_val(n, x, incx, &k, &r, &fail);

if (fail.code != NE_NOERROR) {
    printf("Error from nag_zamin_val (f16jtc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print the absolutely minimum value */
printf("Absolutely minimum element of x is %12.5f\n", r);
/* Print its location */
printf("Index of absolutely minimum element of x is %3" NAG_IFMT "\n", k);

END:
    NAG_FREE(x);

    return exit_status;
}

```

10.2 Program Data

```

nag_zamin_val (f16jtc) Example Program Data
   3      1                                : n and incx
  (-4., 2.1)   ( 3.7, 4.5)   (-6., 1.2)   : Vector x

```

10.3 Program Results

```

nag_zamin_val (f16jtc) Example Program Results

Absolutely minimum element of x is      6.10000
Index of absolutely minimum element of x is    0

```
