

NAG Library Function Document

nag_dmin_val (f16jpc)

1 Purpose

nag_dmin_val (f16jpc) computes the smallest component of a real vector, along with the index of that component.

2 Specification

```
#include <nag.h>
#include <nagf16.h>
void nag_dmin_val (Integer n, const double x[], Integer incx, Integer *k,
                  double *r, NagError *fail)
```

3 Description

nag_dmin_val (f16jpc) computes the smallest component, r , of an n -element real vector x , and determines the smallest index, k , such that

$$r = x_k = \min_j x_j.$$

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 | <i>Input</i> |
| | <i>On entry:</i> n , the number of elements in x . | |
| | <i>Constraint:</i> $n \geq 0$. | |
| 2: | x [<i>dim</i>] – const double | <i>Input</i> |
| | Note: the dimension, <i>dim</i> , of the array x must be at least $\max(1, 1 + (n - 1) \times \mathbf{incx})$. | |
| | <i>On entry:</i> the vector x . Element x_i is stored in x [($i - 1$) \times incx], for $i = 1, 2, \dots, n$. | |
| 3: | incx – Integer | <i>Input</i> |
| | <i>On entry:</i> the increment in the subscripts of x between successive elements of x . | |
| | <i>Constraint:</i> incx \neq 0. | |
| 4: | k – Integer * | <i>Output</i> |
| | <i>On exit:</i> k , the index, from the set $\{0, \mathbf{incx} , \dots, (n - 1) \times \mathbf{incx} \}$, of the smallest component of x . If $n = 0$ on input then k is returned as -1 . | |
| 5: | r – double * | <i>Output</i> |
| | <i>On exit:</i> r , the smallest component of x . If $n = 0$ on input then r is returned as 0.0. | |

6: **fail** – NagError *

Input/Output

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

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.

See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM

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

NE_INT

On entry, **inex** = $\langle value \rangle$.

Constraint: **inex** $\neq 0$.

On entry, **n** = $\langle value \rangle$.

Constraint: **n** ≥ 0 .

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.

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 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 smallest component and index of that component for the vector

$$x = (1, 10, 11, -2, 9)^T.$$

10.1 Program Text

```
/* nag_dmin_val (f16jpc) Example Program.
 *
 * Copyright 2014 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, incx, k, n, xlen;
    double   r;
    /* Arrays */
    double   *x = 0;
    /* Nag Types */
    NagError fail;

    exit_status = 0;
    INIT_FAIL(fail);

    printf("nag_dmin_val (f16jpc) 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

    xlen = MAX(1, 1 + (n - 1)*ABS(incx));

    if (n > 0)
    {
        /* Allocate memory */
        if (!(x = NAG_ALLOC(xlen, 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)
#ifdef _WIN32
        scanf_s("%lf", &x[i]);
#else
        scanf("%lf", &x[i]);
#endif
    /* Input scalar r */
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif

    /* nag_dmin_val (f16jpc).
     * Get minimum value (i) and location of that value (k)
     * of double vector */
    nag_dmin_val(n, x, incx, &k, &r, &fail);

    if (fail.code != NE_NOERROR)
    {

```

```
    printf("Error from nag_dmin_val (f16jpc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

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

END:
NAG_FREE(x);

return exit_status;
}
```

10.2 Program Data

```
nag_dmin_val (f16jpc) Example Program Data
  5  1
  1.0  10.0  11.0  -2.0  9.0
: n and incx
: Array x
```

10.3 Program Results

```
nag_dmin_val (f16jpc) Example Program Results

Minimum element of x is      -2.00000
Index of minimum element of x is  3
```
