NAG Library Function Document

nag_imin_val (f16dpc)

1 Purpose

nag_imin_val (f16dpc) computes the smallest component of an integer vector, along with the index of that component.

2 Specification

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

3 Description

nag_imin_val (f16dpc) computes the smallest component, $i$, of an $n$-element integer vector $x$, and determines the smallest index, $k$, such that

$$i = x_k = \min_j x_j.$$ 

4 References


5 Arguments

1:  
   $n$ – Integer  
   \textit{Input}  
   \textit{On entry}: $n$, the number of elements in $x$.  
   \textit{Constraint}: $n \geq 0$.

2:  
   $x[\text{dim}]$ – const Integer  
   \textit{Input}  
   \textit{Note}: the dimension, $\text{dim}$, of the array $x$ must be at least $\max(1, 1 + (n - 1) \times |\text{incx}|)$.  
   \textit{On entry}: the $n$-element vector $x$.  
   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. If $n = 0$, $x$ is not referenced and may be NULL.

3:  
   $\text{incx}$ – Integer  
   \textit{Input}  
   \textit{On entry}: the increment in the subscripts of $x$ between successive elements of $x$.  
   \textit{Constraint}: $\text{incx} \neq 0$.

4:  
   $k$ – Integer  
   \textit{Output}  
   \textit{On exit}: $k$, the index, from the set $\{0, |\text{incx}|, \ldots, (n - 1) \times |\text{incx}|\}$, of the smallest component of $x$. If $n = 0$ on input then $k$ is returned as $-1$. 

Mark 25
5: i – Integer *  
   *Output
   On exit: i, the smallest component of x. If n = 0 on input then i is returned as 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 (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 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_imin_val (f16dpc) 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, j, k, n, xlen;
    /* Arrays */
    Integer *x = 0;
    /* Nag Types */
    NagError fail;

    exit_status = 0;
    INIT_FAIL(fail);

    printf("nag_imin_val (f16dpc) 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, Integer)))
        {
            printf("Allocation failure\n");
            exit_status = -1;
            goto END;
        }
    }
    else
    {
        printf("Invalid n\n");
        exit_status = 1;
        goto END;
    }

    /* Input vector x */
    for (j = 0; j < xlen; j = j + incx)
    #ifdef _WIN32
    scanf_s("%"NAG_IFMT"", &x[j]);
    #else
    scanf("%"NAG_IFMT"", &x[j]);
    #endif
    #ifdef _WIN32
    scanf_s("%*[\n ] ");
    #else
    scanf("%*[\n ] ");
    #endif

    ...

END:
exit_status = 0;
return
}
#endif

/* nag_imin_val (f16dpc).
 * Get minimum value (i) and location of that value (k)
 * of Integer vector */
	nag_imin_val(n, x, incx, &k, &i, &fail);

if (fail.code != NE_NOERROR)
{
    printf("Error from nag_imin_val (f16dpc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print the minimum value */
printf("Minimum element of x is %12"NAG_IFMT"\n", i);
/* 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_imin_val (f16dpc) Example Program Data

5 1
10 11 -2 9

: n and incx
: Array x

10.3 Program Results

nag_imin_val (f16dpc) Example Program Results

Minimum element of x is     -2
Index of minimum element of x is     3