

NAG Library Function Document

nag_search_double (m01nac)

1 Purpose

nag_search_double (m01nac) searches an ordered vector of double numbers and returns the index of the first value equal to the sought-after item.

2 Specification

```
#include <nag.h>
#include <nagm01.h>
```

```
Integer nag_search_double (Nag_Boolean validate, const double rv[],
                          Integer m1, Integer m2, double item, NagError *fail)
```

3 Description

nag_search_double (m01nac) is based on Professor Niklaus Wirth's implementation of the Binary Search algorithm (see Wirth (2004)), but with two modifications. First, if the sought-after item is less than the value of the first element of the array to be searched, -1 is returned. Second, if a value equal to the sought-after item is not found, the index of the immediate lower value is returned.

4 References

Wirth N (2004) *Algorithms and Data Structures* 35–36 Prentice Hall

5 Arguments

- 1: **validate** – Nag_Boolean *Input*
On entry: if **validate** is set to Nag_TRUE argument checking will be performed. If **validate** is set to Nag_FALSE nag_search_double (m01nac) will be called without argument checking (which includes checking that array **rv** is sorted in ascending order) and the function will return with **fail.code** = NE_NOERROR. See Section 9 for further details.
- 2: **rv[m2 + 1]** – const double *Input*
On entry: elements **m1** to **m2** contain double values to be searched.
Constraint: elements **m1** to **m2** of **rv** must be sorted in ascending order.
- 3: **m1** – Integer *Input*
On entry: the index of the first element of **rv** to be searched.
Constraint: **m1** ≥ 0 .
- 4: **m2** – Integer *Input*
On entry: the index of the last element of **rv** to be searched.
Constraint: **m2** \geq **m1**.
- 5: **item** – double *Input*
On entry: the sought-after item.

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 $\langle value \rangle$ had an illegal value.

NE_INT

On entry, $m1 = \langle value \rangle$.

Constraint: $m1 \geq 0$.

NE_INT_2

On entry, $m1 = \langle value \rangle$, $m2 = \langle value \rangle$.

Constraint: $m1 \leq m2$.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

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.

NE_NOT_INCREASING

On entry, **rv** must be sorted in ascending order: **rv** element $\langle value \rangle >$ element $\langle value \rangle$.

7 Accuracy

Not applicable.

8 Parallelism and Performance

nag_search_double (m01nac) is not threaded in any implementation.

9 Further Comments

The argument **validate** should be used with caution. Set it to Nag_FALSE only if you are confident that the other arguments are correct, in particular that array **rv** is in fact arranged in ascending order. If you wish to search the same array **rv** many times, you are recommended to set **validate** to Nag_TRUE on first call of nag_search_double (m01nac) and to Nag_FALSE on subsequent calls, in order to minimize the amount of time spent checking **rv**, which may be significant if **rv** is large.

The time taken by nag_search_double (m01nac) is $O(\log(n))$, where $n = m2 - m1 + 1$, when **validate** = Nag_FALSE.

10 Example

This example reads a list of double precision numbers and sought-after items and performs the search for these items.

10.1 Program Text

```
/* nag_search_double (m01nac) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
/* Pre-processor includes */
#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagm01.h>

int main(void)
{
    /*Logical scalar and array declarations */
    Nag_Boolean validate;
    /*Integer scalar and array declarations */
    Integer exit_status = 0;
    Integer i, index, lenrv, m1, m2;
    /*Double scalar and array declarations */
    double item;
    double *rv = 0;
    NagError fail;

    INIT_FAIL(fail);

    printf("%s\n", "nag_search_double (m01nac) Example Program Results");
    printf("\n");
#ifdef _WIN32
    scanf_s("%*[^\\n] ");
#else
    scanf("%*[^\\n] ");
#endif
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "%*[^\\n] ", &lenrv);
#else
    scanf("%" NAG_IFMT "%*[^\\n] ", &lenrv);
#endif
    if (!(rv = NAG_ALLOC(lenrv, double)))
    {
        printf("Allocation failure\\n");
        exit_status = -1;
        goto END;
    }
    /* Read in Reference Vector rv */
    for (i = 0; i < lenrv; i++)
#ifdef _WIN32
        scanf_s("%lf ", &rv[i]);
#else
        scanf("%lf ", &rv[i]);
#endif
#ifdef _WIN32
    scanf_s("%*[^\\n] ");
#else
    scanf("%*[^\\n] ");
#endif
    /* Read items sought in the reference vector */
    validate = Nag_TRUE;
#ifdef _WIN32
```

```

    while (scanf_s("%lf%*[\n] ", &item) != EOF)
#else
    while (scanf("%lf%*[\n] ", &item) != EOF)
#endif
    {
        m1 = 0;
        m2 = lenrv - 1;
        /*
         * nag_search_double (m01nac)
         * Binary search in set of double precision numbers
         */
        index = nag_search_double(validate, rv, m1, m2, item, &fail);
        if (fail.code != NE_NOERROR) {
            printf("Error from nag_search_double (m01nac).\n%s\n", fail.message);
            exit_status = 1;
            goto END;
        }
        if (validate) {
            /* Print the reference vector */
            printf("%s\n", "Reference Vector is:");
            for (i = 0; i < lenrv; i++)
                printf("%7.1f%s", rv[i], (i + 1) % 8 ? " " : "\n");
            printf("\n");
            validate = Nag_FALSE;
        }
        printf("\n");
        printf("  Search for item   %7.1f returned index: %4" NAG_IFMT "\n", item,
              index);
    }
END:
    NAG_FREE(rv);

    return exit_status;
}

```

10.2 Program Data

nag_search_double (m01nac) Example Program Data								
16								: lenrv
0.5	0.6	1.1	1.2	1.3	1.3	2.1	2.3	
2.3	4.1	5.8	5.9	6.5	6.5	8.6	9.9	: rv
2.1								: item 1
0.4								: item 2
7.1								: item 3
10.0								: item 4

10.3 Program Results

nag_search_double (m01nac) Example Program Results

Reference Vector is:							
0.5	0.6	1.1	1.2	1.3	1.3	2.1	2.3
2.3	4.1	5.8	5.9	6.5	6.5	8.6	9.9
Search for item	2.1 returned index:					6	
Search for item	0.4 returned index:					-1	
Search for item	7.1 returned index:					13	
Search for item	10.0 returned index:					15	
