NAG Library Function Document

nag_example_file_io (x04aec)

1 Purpose

nag_example_file_io (x04aec) reads command-line arguments and returns either a file pointer or a file name depending on the argument **flag** specified in **argv**.

2 Specification

```
#include <nag.h>
#include <nagx04.h>
FILE * nag_example_file_io (int argc, const char *argv[], const char *flag,
       char **fname)
```

3 Description

nag_example_file_io (x04aec) returns either a FILE pointer or the name of a file depending on the argument flag specified in **argv**. If the argument **flag** is either "-data" or "-results", a FILE pointer is returned for data input or output respectively. If the argument **flag** is "-options", "-nag_write" or "-nag_read", a char pointer is returned in **fname** to hold the name of an options file, a data output or a data input file.

4 References

None.

5 Arguments

1: **argc** – int *Input*

On entry: the number of command-line arguments.

2: **argv**[**argc**] – const char * *Input*

On entry: the argument vector.

3: **flag** – const char * *Input*

On entry: indicates which file pointer or file name will be returned by the function. nag_example_file_io (x04aec) searches the command-line arguments contained in **argv** for **flag**. If it is found, in say **argv**[$i-1$], the function examines **argv**[i] and if it is a filename the function returns either a file pointer or the filename as follows:

flag = "-data"
: If **argv**[i] is a filename it specifies the data file to be opened for reading and a FILE pointer to that file is returned. Otherwise a FILE pointer to `stdin` is returned.

flag = "-results"
: If **argv**[i] is a filename it specifies the results file to be opened for writing and a FILE pointer to that file is returned. Otherwise a FILE pointer to `stdout` is returned.

flag = "-options"
: If **argv**[i] is a filename it specifies a file containing optional arguments to be opened for reading and this filename is returned in **fname**. Otherwise a default filename comprising the stem of the program name with a file extension of .opt (e.g., e04ucce.opt) is returned in

fname. Note that memory is allocated internally to **fname** using NAG_ALLOC. It can be freed using NAG_FREE.

flag = "-nag_write"
If **argv**$[i]$ is a filename it specifies a file to be opened for library output and this filename is returned in **fname**. Otherwise a **NULL** is returned in **fname**.

flag = "-nag_read"
If **argv**$[i]$ is a filename it specifies a file to be opened for reading library input and this filename is returned in **fname**. Otherwise a **NULL** is returned in **fname**.

Constraint: **flag** = "-data", "-results", "-options", "-nag_write" or "-nag_read".

4: **fname** – char ** *Input*

On exit: if the **flag** specified is "-options", "-nag_write" or "-nag_read", **fname** contains the name of the file for reading or writing.

6 Error Indicators and Warnings

None.

7 Accuracy

Not applicable.

8 Parallelism and Performance

Not applicable.

9 Further Comments

None.

10 Example

This program illustrates the use of nag_example_file_io (x04aec) to check for input and output file names on the command line, before making a call to nag_1d_aitken_interp (e01aac).

10.1 Program Text

```
/* nag_example_file_io (x04aec) Example Program.
 *
 * Copyright 2011, Numerical Algorithms Group.
 *
 * Mark 23, 2011.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nage01.h>
#include <nagx04.h>
#include <nag_example_file_io.h>

int main(int argc, char *argv[])
{
  /* Scalars */
  FILE    *fpin = 0, *fpout = 0;
  Integer exit_status = 0;
  Integer i, j, k, n;
  double  x;
  NagError fail;
  /* Arrays */
  double  *a = 0, *b = 0, *c = 0;

  INIT_FAIL(fail);

  /* Check for command-line IO options */
  fpout = nag_example_file_io(argc, (const char **)argv, "-results", NULL);
  fpin = nag_example_file_io(argc, (const char **)argv, "-data", NULL);
  fprintf(fpout, "nag_example_file_io (x04aec) Example Program Results\n");

  /* Skip heading in data file*/
  fscanf(fpin, "%*[^\n] ");
  fscanf(fpin, "%"NAG_IFMT "", &n);
  fscanf(fpin, "%lf", &x);
  fscanf(fpin, "%*[^\n] ");

  /* Allocate memory */
  if (!(a = NAG_ALLOC(n + 1, double)) ||
      !(b = NAG_ALLOC(n + 1, double)) ||
      !(c = NAG_ALLOC(n * (n + 1)/2, double)))
    {
      fprintf(fpout, "Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  for (i = 0; i < n + 1; i++){
    fscanf(fpin, "%lf", &a[i]);
  }
  fscanf(fpin, "%*[^\n] ");
  for (i = 0; i < n + 1; i++){
    fscanf(fpin, "%lf", &b[i]);
  }
  fscanf(fpin, "%*[^\n] ");

  /* nag_1d_aitken_interp (e01aac).
   * Interpolated values, Aitken's technique,
   * unequally spaced data, one variable.
   */
  nag_1d_aitken_interp(n, a, b, c, x, &fail);
  if (fail.code != NE_NOERROR){
    fprintf(fpout, "Error from nag_1d_aitken_interp (e01aac).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  fprintf(fpout, "\nInterpolated values\n");
  k = 0;
  for (i = 1; i <= n - 1; i++){
    for (j = k; j <= k + n - i; j++){
      fprintf(fpout, "%12.5f", c[j]);
    }
    fprintf(fpout, "\n");
    k = j;
  }
  fprintf(fpout, "\nInterpolation point = %12.5f\n", x);
  fprintf(fpout, "\nFunction value at interpolation point = %12.5f\n",
         c[n * (n + 1)/2 - 1]);

 END:
  if (fpin != stdin) fclose(fpin);
  if (fpout != stdout) fclose(fpout);
  NAG_FREE(a);
  NAG_FREE(b);
  NAG_FREE(c);

  return exit_status;
}
```

10.2 Program Data

None.

10.3 Program Results

```
nag_example_file_io (x04aec) Example Program Results

Interpolated values
    -1.35680    -1.28000    -0.39253     1.28000     5.67808
    -1.23699    -0.60467     0.01434     1.38680
    -0.88289    -0.88662    -0.74722
    -0.88125    -0.91274

Interpolation point =      0.28000

Function value at interpolation point =     -0.83591
```