NAG Library Function Document

nag_glopt_bnd_mcs_optset_file (e05jcc)

1 Purpose

nag_glopt_bnd_mcs_optset_file (e05jcc) may be used to supply optional arguments to nag_glopt_bnd_mcs_solve (e05jbc) from an external file. The initialization function nag_glopt_bnd_mcs_init (e05jac) **must** have been called before calling nag_glopt_bnd_mcs_optset_file (e05jcc).

2 Specification

```
#include <nag.h>
#include <nage05.h>
```

```
void nag_glopt_bnd_mcs_optset_file (Nag_FileID fileid, Nag_E05State *state,
     NagError *fail)
```

3 Description

nag_glopt_bnd_mcs_optset_file (e05jcc) may be used to supply values for optional arguments to nag_glopt_bnd_mcs_solve (e05jbc). nag_glopt_bnd_mcs_optset_file (e05jcc) reads an external file which has been opened by a call to nag_open_file (x04acc). Each line of the file defines a single optional argument. It is only necessary to supply values for those arguments whose values are to be different from their default values.

Each optional argument is defined by a single character string, consisting of one or more items. The items associated with a given optional argument must be separated by spaces, or equals signs [=]. Alphabetic characters may be upper or lower case. The string

```
Static Limit = 100
```

is an example of a string used to set an optional argument. For each optional argument the string contains one or more of the following items:

- a mandatory keyword;
- a phrase that qualifies the keyword;
- a number that specifies an integer or real value. Such numbers may be up to 16 contiguous characters.

Blank strings and comments are ignored. A comment begins with an asterisk (*) and all subsequent characters in the string are regarded as part of the comment.

The implied data type (character, integer or real) of each value to set **must** match that expected by the corresponding optional argument.

The file containing the optional arguments must start with `Begin` and must finish with `End`. An example of a valid options file is:

```
Begin * Example options file
   Static Limit = 500
End
```

Optional argument settings are preserved following a call to nag_glopt_bnd_mcs_solve (e05jbc) and so the keyword **Defaults** is provided to allow you to reset all the optional arguments to their default values before a subsequent call to nag_glopt_bnd_mcs_solve (e05jbc).

A complete list of optional arguments, their symbolic names and default values is given in Section 12 in nag_glopt_bnd_mcs_solve (e05jbc).

4 References

None.

5 Arguments

1: **fileid** – Nag_FileID *Input*

On entry: the ID of the option file to be read, as returned by a call to nag_open_file (x04acc).

2: **state** – Nag_E05State * *Communication Structure*

state contains information required by other functions in this suite. You must not modify it directly in any way.

3: **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.

NE_BAD_PARAM

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

NE_FILE_NOT_READ

At least one optional argument from the options file could not be recognized. *All optional arguments that were set from the file before this error was encountered will remain set on exit.*

BEGIN found, but end-of-file found before END. *All optional arguments that were set from the file before this error was encountered will remain set on exit.*

Could not read options file.

End-of-file found before BEGIN.

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.

NE_NOT_INIT

Initialization function nag_glopt_bnd_mcs_init (e05jac) has not been called.

NE_NOT_PARSED

One of the numeric values to be set could not be parsed. Check that all such strings specify valid integer or real values.

NE_OUT_OF_RANGE

Attempt to assign an illegal value of **Local Searches** ($lcsrch$): $lcsrch =$ "$\langle$*value*$\rangle$".

Attempt to assign an illegal value of **Repeatability** ($repeat$): $repeat =$ "$\langle$*value*$\rangle$".

Attempt to assign a non-positive value of **Function Evaluations Limit** (nf): $nf = \langle$*value*$\rangle$.

Attempt to assign a non-positive value of **Local Searches Limit** ($loclim$): $loclim = \langle$*value*$\rangle$.

Attempt to assign a non-positive value of **Static Limit** ($stclim$): $stclim = \langle$*value*$\rangle$.

Attempt to assign an out-of-bounds value of **Infinite Bound Size** (*infbnd*): $infbnd = \langle value \rangle$.

Attempt to assign too small a value of **Local Searches Tolerance** (*loctol*): $loctol = \langle value \rangle$.

Attempt to assign too small a value of **Target Objective Error** (*objerr*): $objerr = \langle value \rangle$.

Attempt to assign too small a value of **Target Objective Safeguard** (*objsfg*): $objsfg = \langle value \rangle$.

7 Accuracy

Not applicable.

8 Parallelism and Performance

Not applicable.

9 Further Comments

nag_glopt_bnd_mcs_optset_string (e05jdc), nag_glopt_bnd_mcs_optset_char (e05jec), nag_glopt_bnd_mcs_optset_int (e05jfc) or nag_glopt_bnd_mcs_optset_real (e05jgc) may also be used to supply optional arguments to nag_glopt_bnd_mcs_solve (e05jbc).

10 Example

This example finds the global minimum of the 'peaks' function in two dimensions

$$F(x,y) = 3(1-x)^2 \exp\Big(-x^2-(y+1)^2\Big) - 10\Big(\frac{x}{5}-x^3-y^5\Big)\exp\big(-x^2-y^2\big) - \frac{1}{3}\exp\Big(-(x+1)^2-y^2\Big)$$

on the box $[-3,3] \times [-3,3]$.

The function F has several local minima and one global minimum in the given box. The global minimum is approximately located at $(0.23, -1.63)$, where the function value is approximately -6.55.

By specifying an initialization list via **list**, **numpts** and **initpt** we can start nag_glopt_bnd_mcs_solve (e05jbc) looking close to one of the local minima and check that it really does move away from that point to one of the global minima.

More precisely, we choose $(-1,0)$ as our initial point (see Section 10.3), and let the initialization list be

$$\begin{pmatrix} -3 & -1 & 3 \\ -3 & 0 & 3 \end{pmatrix}.$$

This example solves the optimization problem using some of the optional arguments described in Section 12 in nag_glopt_bnd_mcs_solve (e05jbc).

10.1 Program Text

```
/* nag_glopt_bnd_mcs_optset_file (e05jcc) Example Program.
 *
 * Copyright 2013 Numerical Algorithms Group.
 *
 * Mark 24, 2013.
 */
#include <string.h>
#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nage05.h>

#ifdef __cplusplus
extern "C" {
#endif
static void NAG_CALL objfun(Integer n, const double x[], double *f,
                          Integer nstate, Nag_Comm *comm, Integer *inform);
static void NAG_CALL monit(Integer n, Integer ncall, const double xbest[],
                           const Integer icount[], Integer sdlist,
                           const double list[], const Integer numpts[],
                           const Integer initpt[], Integer nbaskt,
                           const double xbaskt[], const double boxl[],
                           const double boxu[], Integer nstate, Nag_Comm *comm,
                           Integer *inform);
static void NAG_CALL output_current_box(const double boxl[],
                                        const double boxu[]);
#ifdef __cplusplus
}
#endif

int main(void)
{
  /* Scalars */
  double            infbnd, obj;
  Integer           exit_status=0, i, ibdchk, n=2, nf, plot, sdlist, stclim;
  Nag_BoundType     boundenum;
  Nag_MCSInitMethod initmethodenum;
  /* Arrays */
  const char        *optionsfile = "e05jcce.opt";
  static double     ruser[2] = {-1.0, -1.0};
  char              bound[16], initmethod[18];
  double            *bl = 0, *bu = 0, *list = 0, *x = 0;
  char              lcsrch[3];
  Integer           *initpt = 0, *numpts = 0;
  Integer           iuser[1];
  /* Nag Types */
  Nag_E05State      state;
  NagError          fail;
  Nag_Comm          comm;
  Nag_FileID        fileid;

  INIT_FAIL(fail);

  printf("nag_glopt_bnd_mcs_optset_file (e05jcc) Example Program Results\n");

  /* For communication with user-supplied functions: */
  comm.iuser = iuser;
  comm.user = ruser;

  /* Skip heading in data file */
  scanf("%*[^\n] ");
  /* Read sdlist from data file */
  scanf("%ld%*[^\n] ", &sdlist);

  if (n <= 0 || sdlist <= 0)
    goto END;

  if (!(bl = NAG_ALLOC(n, double)) ||
      !(bu = NAG_ALLOC(n, double)) ||
      !(list = NAG_ALLOC(n*sdlist, double)) ||
      !(x = NAG_ALLOC(n, double)) ||
      !(initpt = NAG_ALLOC(n, Integer)) ||
      !(numpts = NAG_ALLOC(n, Integer)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Read in bound (and bl and bu if necessary) */
  scanf("%15s%*[^\n] ", bound);

  /* nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value
   */
  boundenum = (Nag_BoundType) nag_enum_name_to_value(bound);
  if (boundenum == Nag_Bounds)
    /* Read in the whole of each bound */
    {
      for (i = 0; i < n; ++i)
        scanf("%lf", &bl[i]);
      scanf("%*[^\n] ");

      for (i = 0; i < n; ++i)
        scanf("%lf", &bu[i]);
      scanf("%*[^\n] ");
    }
  else if (boundenum == Nag_BoundsEqual)
    /* Bounds are uniform: read in only the first entry of each */
    {
      scanf("%lf%*[^\n] ", &bl[0]);
      scanf("%lf%*[^\n] ", &bu[0]);
    }

  /* Read in initmethod (and list, numpts and initpt if necessary) */
  scanf("%17s%*[^\n] ", initmethod);

  /* nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value
   */
  initmethodenum = (Nag_MCSInitMethod) nag_enum_name_to_value(initmethod);

  if (initmethodenum == Nag_UserSet)
    {
      for (i = 0; i < n*sdlist; ++i)
        scanf("%lf", &list[i]);
      scanf("%*[^\n] ");

      for (i = 0; i < n; ++i)
        scanf("%ld", &numpts[i]);
      scanf("%*[^\n] ");

      for (i = 0; i < n; ++i)
        scanf("%ld", &initpt[i]);
      scanf("%*[^\n] ");
    }

  /* Read in plot. Its value determines whether monit displays
   * information on the current search box
   */
  scanf("%ld%*[^\n] ", &plot);

  /* Communicate plot through to monit */
  iuser[0] = plot;

  /* Call nag_glopt_bnd_mcs_init (e05jac) to initialize
   * nag_glopt_bnd_mcs_solve (e05jbc). */
  /* Its first argument is a legacy argument and has no significance. */
  nag_glopt_bnd_mcs_init(0, &state, &fail);

  if (fail.code != NE_NOERROR)
    {
      printf("Initialization of nag_glopt_bnd_mcs_solve (e05jbc) failed.\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }

  /* Use nag_glopt_bnd_mcs_optset_file (e05jcc) to read some options
   * from the end of the data file */

  /* Call nag_open_file (x04acc) to set the stdin fileid. */
  /* nag_open_file (x04acc).
   * Open unit number for reading, writing or appending, and
   * associate unit with named file
   */
  nag_open_file(optionsfile, 0, &fileid, &fail);
```

```
if (fail.code != NE_NOERROR)
  {
    printf("Fileid could not be obtained.\n");
    exit_status = 1;
    goto END;
  }

/* nag_glopt_bnd_mcs_optset_file (e05jcc).
 * Supply optional parameter values for nag_glopt_bnd_mcs_solve (e05jbc)
 * from external file */
nag_glopt_bnd_mcs_optset_file(fileid, &state, &fail);

if (fail.code != NE_NOERROR)
  {
    printf("nag_glopt_bnd_mcs_optset_file (e05jcc) failed.\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

/* Use nag_glopt_bnd_mcs_optget_int (e05jkc) to find the value of
 * the Integer-valued option 'Function Evaluations Limit' */
/* nag_glopt_bnd_mcs_optget_int (e05jkc).
 * Get the setting of an Integer-valued option of
 * nag_glopt_bnd_mcs_solve (e05jbc) */
nag_glopt_bnd_mcs_optget_int("Function Evaluations Limit", &nf, &state,
                             &fail);

if (fail.code != NE_NOERROR)
  {
    printf("nag_glopt_bnd_mcs_optget_int (e05jkc) failed.\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

printf("\nOption 'Function Evaluations Limit' has the value %ld.\n",
       nf);

/* Use nag_glopt_bnd_mcs_optset_int (e05jfc) to set the value of the
 * Integer-valued option 'Static Limit' */

stclim = 4*n;

/* nag_glopt_bnd_mcs_optset_int (e05jfc).
 * Set a single Integer-valued option of
 * nag_glopt_bnd_mcs_solve (e05jbc) */
nag_glopt_bnd_mcs_optset_int("Static Limit", stclim, &state, &fail);

if (fail.code != NE_NOERROR)
  {
    printf("nag_glopt_bnd_mcs_optset_int (e05jfc) failed.\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

/* Use nag_glopt_bnd_mcs_option_check (e05jhc) to determine whether
 * the real-valued option 'Infinite Bound Size' has been set by us
 * (in which case 1 is returned) or whether it holds its default
 * value (in which case 0 is returned) */
/* nag_glopt_bnd_mcs_option_check (e05jhc).
 * Determine whether the user has set a single option of
 * nag_glopt_bnd_mcs_solve (e05jbc) */
ibdchk = nag_glopt_bnd_mcs_option_check("Infinite Bound Size", &state, &fail);

if (fail.code != NE_NOERROR)
  {
    printf("nag_glopt_bnd_mcs_option_check (e05jhc) failed.\n%s\n",
           fail.message);
```

```
      exit_status = 1;
      goto END;
    }

  printf("Option ’Infinite Bound Size’ ");
  printf((ibdchk == 1 ? "has been set by us" : "holds its default value"));
  printf(".\n");

  /* Use nag_glopt_bnd_mcs_optget_real (e05jlc) to find the value of
   * the real-valued option ’Infinite Bound Size’ */
  /* nag_glopt_bnd_mcs_optget_real (e05jlc).
   * Get the setting of a real-valued option of
   * nag_glopt_bnd_mcs_solve (e05jbc) */
  nag_glopt_bnd_mcs_optget_real("Infinite Bound Size", &infbnd, &state, &fail);

  if (fail.code != NE_NOERROR)
    {
      printf("nag_glopt_bnd_mcs_optget_real (e05jlc) failed.\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }

  printf("Option ’Infinite Bound Size’ has the value %14.5e\n", infbnd);

  /* Use nag_glopt_bnd_mcs_optset_real (e05jgc) to increase the value of
   * the real-valued option ’Infinite Bound Size’ tenfold */

  infbnd = 10.0*infbnd;

  /* nag_glopt_bnd_mcs_optset_real (e05jgc).
   * Set the value of a real-valued option of
   * nag_glopt_bnd_mcs_solve (e05jbc) */
  nag_glopt_bnd_mcs_optset_real("Infinite Bound Size", infbnd, &state, &fail);

  if (fail.code != NE_NOERROR)
    {
      printf("nag_glopt_bnd_mcs_optset_real (e05jgc) failed.\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }

  /* Use nag_glopt_bnd_mcs_optset_string (e05jdc) to set the value of
   * the Integer-valued option ’Local Searches Limit’ */
  /* nag_glopt_bnd_mcs_optset_string (e05jdc).
   * Set the value of an option of nag_glopt_bnd_mcs_solve (e05jbc)
   * from a string */
  nag_glopt_bnd_mcs_optset_string("Local Searches Limit = 40", &state, &fail);

  if (fail.code != NE_NOERROR)
    {
      printf("nag_glopt_bnd_mcs_optset_string (e05jdc) failed.\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }

  /* Use nag_glopt_bnd_mcs_optset_char (e05jec) to set the value of
   * the ’On’/’Off’-valued character option ’Local Searches’ */
  strcpy(&lcsrch[0], "On");

  /* nag_glopt_bnd_mcs_optset_char (e05jec).
   * Set the value of an ’On’/’Off’-valued character option of
   * nag_glopt_bnd_mcs_solve (e05jbc) */
  nag_glopt_bnd_mcs_optset_char("Local Searches", lcsrch, &state, &fail);

  if (fail.code != NE_NOERROR)
    {
      printf("nag_glopt_bnd_mcs_optset_char (e05jec) failed.\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }

  /* Solve the problem. */
  /* nag_glopt_bnd_mcs_solve (e05jbc).
   * Global optimization by multilevel coordinate search, simple bounds.
   */
  nag_glopt_bnd_mcs_solve(n, objfun, boundenum, initmethodenum, bl, bu,
                          sdlist, list, numpts, initpt, monit, x, &obj,
                          &state, &comm, &fail);

  if (fail.code != NE_NOERROR)
    {
      printf("Error message from nag_glopt_bnd_mcs_solve (e05jbc).\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }

  printf("Final objective value = %11.5f\n", obj);
  printf("Global optimum x = ");
  for (i = 0; i < n; ++i)
    printf("%9.5f", x[i]);
  printf("\n");

 END:
  NAG_FREE(bl);
  NAG_FREE(bu);
  NAG_FREE(list);
  NAG_FREE(x);
  NAG_FREE(initpt);
  NAG_FREE(numpts);

  return exit_status;
}

static void NAG_CALL objfun(Integer n, const double x[], double *f,
                            Integer nstate, Nag_Comm *comm, Integer *inform)
{
  /* Routine to evaluate objective function */

  if (comm->user[0] == -1.0)
    {
      printf("(User-supplied callback objfun, first invocation.)\n");
      comm->user[0] = 0.0;
    }

  /* This is a two-dimensional objective function.
   * As an example of using the inform mechanism,
   * terminate if any other problem size is supplied.
   */
  if (n!=2)
    {
      *inform = -1;
      return;
    }

  *inform = 0;

  if (*inform >= 0)
  /* Here we’re prepared to evaluate objfun at the current x */
    {
      if (nstate == 1)
      /* This is the first call to objfun */
        {
          printf("\n(objfun was just called for the first time)\n");
        }

      *f = (
        3.0*pow((1.0-x[0]), 2)*exp(-pow(x[0], 2)-pow((x[1]+1), 2))
```

```
         - (10.0*(x[0]/5.0-pow(x[0], 3)-pow(x[1], 5))*
            exp(-pow(x[0], 2)-pow(x[1], 2)))
         - 1.0/3.0*exp(-pow((x[0]+1.0), 2)-pow(x[1], 2))
         );
    }
}

static void NAG_CALL monit(Integer n, Integer ncall, const double xbest[],
                           const Integer icount[], Integer sdlist,
                           const double list[], const Integer numpts[],
                           const Integer initpt[], Integer nbaskt,
                           const double xbaskt[], const double boxl[],
                           const double boxu[], Integer nstate, Nag_Comm *comm,
                           Integer *inform)
{
  /* Scalars */
  Integer i, j;
  Integer plot;

#define LIST(I, J) list[(I-1)*sdlist + (J-1)]
#define XBASKT(I, J) xbaskt[(I-1)*nbaskt + (J-1)]

  if (comm->user[1] == -1.0)
    {
      printf("(User-supplied callback monit, first invocation.)\n");
      comm->user[1] = 0.0;
    }

  *inform = 0;

  if (*inform >= 0)
  /* We are going to allow the iterations to continue */
    {
      /* Extract plot from the communication structure */
      plot = comm->iuser[0];

      if (nstate == 0 || nstate == 1)
      /* When nstate == 1, monit is called for the first time.
       * When nstate == 0, monit is called for the first AND last time.
       * Display a welcome message */
        {
          printf("\n*** Begin monitoring information ***\n\n");

          printf("Values controlling initial splitting of a box:\n");
          for (i = 1; i <= n; ++i)
            {
              printf("**\n");
              printf("In dimension %5ld\n", i);
              printf("Extent of initialization list in this dimension ="
                     "%5ld\n", numpts[i - 1]);
              printf("Initialization points in this dimension:\n");
              printf("LIST(i, 1:numpts[i - 1]) =");
              for (j = 1; j <= numpts[i - 1]; ++j)
                printf("%9.5f", LIST(i, j));
              printf("\n");
              printf("Initial point in this dimension: LIST(i,%5ld)\n",
                     initpt[i - 1]);
            }

          if (plot != 0 && n == 2)
            printf("<Begin displaying search boxes>\n\n");
        }

      if (plot != 0 && n == 2)
        {
          /* Display the coordinates of the edges of the current search box */
          output_current_box(boxl, boxu);
        }

      if (nstate <= 0)
      /* monit is called for the last time */
```

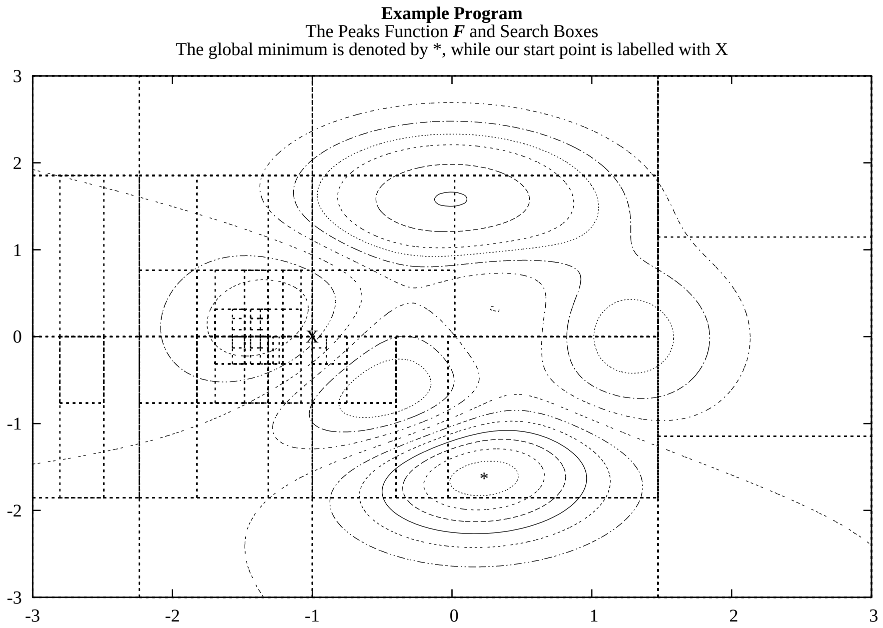
```
        {
          if (plot != 0 && n == 2)
            printf("<End displaying search boxes>\n\n");
          printf("Total sub-boxes = %5ld\n", icount[0]);
          printf("Total function evaluations = %5ld\n", ncall);
          printf("Total function evaluations used in local search = "
                 "%5ld\n", icount[1]);
          printf("Total points used in local search = %5ld\n",
                  icount[2]);
          printf("Total sweeps through levels = %5ld\n", icount[3]);
          printf("Total splits by init. list = %5ld\n", icount[4]);
          printf("Lowest level with nonsplit boxes = %5ld\n",
                  icount[5]);
          printf("Number of candidate minima in the 'shopping basket'"
                  " = %5ld\n", nbaskt);
          printf("Shopping basket:\n");

          for (i = 1; i <= n; ++i)
            {
              printf("xbaskt(%3ld,:) =", i);
              for (j = 1; j <= nbaskt; ++j)
                printf("%9.5f", XBASKT(i, j));
              printf("\n");
            }

          printf("Best point:\n");
          printf("xbest =");
          for (i = 0; i < n; ++i)
            printf("%9.5f", xbest[i]);
          printf("\n");

          printf("\n*** End monitoring information ***\n\n");
        }
    }
}

static void NAG_CALL output_current_box(const double boxl[],
                                        const double boxu[])
{
  printf("%20.15f %20.15f\n", boxl[0], boxl[1]);
  printf("%20.15f %20.15f\n\n", boxl[0], boxu[1]);
  printf("%20.15f %20.15f\n", boxl[0], boxl[1]);
  printf("%20.15f %20.15f\n\n", boxu[0], boxl[1]);
  printf("%20.15f %20.15f\n", boxl[0], boxu[1]);
  printf("%20.15f %20.15f\n\n", boxu[0], boxu[1]);
  printf("%20.15f %20.15f\n", boxu[0], boxl[1]);
  printf("%20.15f %20.15f\n\n", boxu[0], boxu[1]);
}
```

10.2 Program Data

```
nag_glopt_bnd_mcs_optset_file (e05jcc) Example Program Data
  3                                     : sdlist
  Nag_Bounds                            : bound
  -3.0   -3.0                           : Lower bounds bl
  3.0   3.0                             : Upper bounds bu
  Nag_UserSet                           : initmethod
  -3.0   -1.0   3.0   -3.0   0.0   3.0  : Matrix list
  3   3   3                             : numpts
  2   2   2                             : Initial-point pointer initpt
  0                                     : plot

Begin example options file
* Comment lines like this begin with an asterisk
* Set the maximum number of function evaluations
Function Evaluations Limit = 100000
* Set the local search termination tolerance
Local Searches Tolerance = 1.0D-10
* Set the maximum number of times a given box may be split
Splits Limit = 20
End
```

10.3 Program Results

```
nag_glopt_bnd_mcs_optset_file (e05jcc) Example Program Results

Option ’Function Evaluations Limit’ has the value 100000.
Option ’Infinite Bound Size’ holds its default value.
Option ’Infinite Bound Size’ has the value    1.15792e+77
(User-supplied callback objfun, first invocation.)

(objfun was just called for the first time)
(User-supplied callback monit, first invocation.)

*** Begin monitoring information ***

Values controlling initial splitting of a box:
**
In dimension     1
Extent of initialization list in this dimension =    3
Initialization points in this dimension:
LIST(i, 1:numpts[i - 1]) = -3.00000 -1.00000  3.00000
Initial point in this dimension: LIST(i,    2)
**
In dimension     2
Extent of initialization list in this dimension =    3
Initialization points in this dimension:
LIST(i, 1:numpts[i - 1]) = -3.00000  0.00000  3.00000
Initial point in this dimension: LIST(i,    2)
Total sub-boxes =   180
Total function evaluations =   185
Total function evaluations used in local search =   102
Total points used in local search =     9
Total sweeps through levels =     9
Total splits by init. list =     5
Lowest level with nonsplit boxes =     6
Number of candidate minima in the ’shopping basket’ =     2
Shopping basket:
xbaskt(  1,:) =  0.22828 -1.34740
xbaskt(  2,:) = -1.62553  0.20452
Best point:
xbest =  0.22828 -1.62553

*** End monitoring information ***

Final objective value =    -6.55113
Global optimum x =   0.22828 -1.62553
```


Example Program
The Peaks Function ***F*** and Search Boxes
The global minimum is denoted by *, while our start point is labelled with X
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3
X
*