Computer Physics Communications 52 (1989) 241–247
North-Holland, Amsterdam

MERLIN-2.0 – ENHANCED AND PROGRAMMABLE VERSION

D.G. PAPAGEORGIOU, C.S. CHASSAPIS and I.E. LAGARIS

Physics Department, University of Ioannina, Ioannina, Greece

Received 2 June 1988

Merlin 2.0 is a new version of the recently published optimization package MERLIN-1.0. Apart from several minor improvements and extensions, this new version offers a very important feature: Programmability. A high level language has been developed, that enables the user to easily write programs that control the new MERLIN-2.0 at run time.

NEW VERSION SUMMARY

Title of new version: MERLIN-2.0 programmable

Cataloque number: ABHB

Program obtainable from: CPC Program Library, Queen's University of Belfast, N. Ireland (see application form in this issue)

Reference to original program: MERLIN-1.0 [1]

Authors of original program: G.A. Evangelakis, J.P. Rizos, I.E. Lagaris and I.N. Demetropoulos

Computer: CDC CYBER-171; *Installation*: University of Ioannina, Ioannina, Greece

Operating system: NOS 2.5.2 678/670

Programming language used: ANSI FORTRAN 77

High speed storage required: depending upon the maximum number of variables the object can handle (36000 words for 30 variables, 58000 words for 150 variables)

Number of bits in a word: 60

Peripherals used: terminal, line printer, card reader, disc

Number of lines in combined program and test deck: 6956

Nature of physical problem
A lot of problems in physics, chemistry, applied mathematics as well as in engineering and in other fields, are quite often reduced to minimizing a function of several variables. MERLIN-2.0 is a programmable system designed to minimize a multi-dimensional function.

Method of solution
Six algorithms are implemented. Three of them make use of the function's gradient and hence are suitable for minimizing differentiable functions, the rest three do not use derivatives at all and so are applicable to non-differentiable functions as well [1].

Reasons for the new version
A new version of the recently published [1] program MERLIN 1.0, for multidimensional optimization, is presented. Apart from several minor improvements and extensions, the new version offers a very important feature: Programmability. A high level language [2] has been developed, that enables the user to easily write programs that control the new MERLIN-2.0 at run time.

Restriction of the complexity of the problem
Currently MERLIN is dimensioned to handle up to 150 variables. However, by redimensioning a few arrays it can easily be enhanced or reduced according to user's needs, as described in the provided manual of the original program.

Typical running time
Since houskeeping operations are quite fast, running time heavily depends on the complexity of the objective function. The provided test run, took 4.8 CPU s on a CDC CYBER-171.

Unusual features of the problem
Apart from being a minimization program, MERLIN is designed to be easily further developed by the user and evolve in different directions. Hence it can serve as a testing ground of different trial algorithms and can be seen as an optimization development system.

References
[1] G.A. Evangelakis, J.P. Rizos, I.E. Lagaris and I.N. Demetropoulos, Comput. Phys. Commun. 46 (1987) 401.
[2] C.S. Chassapis, D.G. Papageorgiou and I.E. Lagaris, Comput. Phys. Commun. 52 (1989) 223.

LONG WRITE-UP

1. Introduction

Minimization techniques have proved to be extremely useful and powerful tools, in tackling a wide variety of problems.

MERLIN-1.0 [1], is a program that was developed to ease the minimization process by offering efficient algorithms and a friendly environment. The philosophy followed during its development was to offer an integrated package of interconnected subprograms, rather than a subroutine library. The next step that came up naturally, was to construct a language [2], to drive and control this program at runtime, the same way an experienced user would from his terminal. Furthermore, using a language one can systematically develop strategies, that can be tested and improved as time goes by, for several different classes of functions. To achieve this we added an "interface" package that implements all the additional operations required by the language.

The main purpose of this paper is to describe this "interface" program that makes the new version of MERLIN programmable.

2. General description

In what follows we describe the additions, extensions and modifications that lead to the new version MERLIN-2.0. The nature of such a description requires familiarity with the old version. MERLIN-2.0 - ENHANCED AND PROGRAMMABLE VERSION, offers some new commands, has employed a few additional I/O units and has modified and/or extended some of the 1.0-version features. The programmability has been facilitated by inserting an "interface" package, as is shown in fig. 1.

There are now two kinds of commands: MERLIN commands and interface commands. All interface commands begin with a $ (dollar-sign). After a command is read in, if it begins with a $, program control is transferred to the interface module, otherwise command processing is done as in the 1.0-version. The interface tests whether the command is a valid one, and if so executes it. The user does not need to know the interface commands; however, for the sake of completeness and to ease possible further development, they are all listed in the MCL manual [2].

New commands

Command **VALDIS**. Displays the function's value alone.

Command **RUNMCL**. Prompts the user to enter the name of the file, where the MERLIN Object Code [2] resides.

Extensions

The **AUTO** command can now be invoked directly from both options of the USERSO mode. It also appears in the catalog, produced by the LIST command.

The **INIT** command has been extended so as to initialize not only the point, but also the left and the right margins.

Modifications

If any of the panel cancel-buttons is set to 2, the panel prompts for additional input. If the point (i.e. all of the minimization variables) is not initialized, all commands that need to evaluate the objective function are disabled and informative messages are issued. In the beginning of the run, one now is prompt to specify the option (BATCH or IAF) of the PROCES-mode.

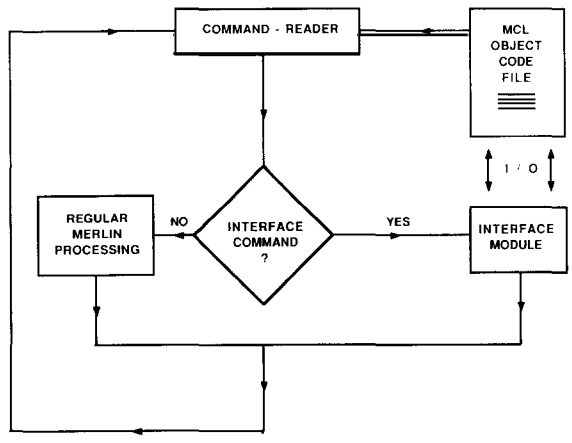

Fig. 1. Interface package.

The units IU37 and IU47 have been reserved, both serving the implementation of MERLIN's programmability. IU37 is associated with the Merlin object code file, while IU47 is connected to a work-file manipulated by the interface and used to input values for certain MERLIN parameters at runtime.

Four subroutines are added to implement the various minor improvements, while the interface is comprised of twelve subprograms. The description of all of the new subprograms is given in the next section.

3. Subprogram description

The interface is a collection of twelve subprograms (8 subroutines and 4 functions). The main work is carried out in subroutine USEMCL, while the rest of the routines take care of file positioning (subroutines FORWAR and BACWAR), or unit inquiry (subroutine DETUNT), arrange for the I/O formatting (subroutines STORIC, TREBUF and FLUSH), implement some of the several MCL intrinsic functions (functions ACOSH0, ASINH0, ATANH0 and FACT), supply debugging aids and upon errors terminate execution appropriately (subroutine STOPER).

The routines added for the several other improvements are the subroutines: SETMODE, UNDEF, AUTIDA and MENU. In what follows we describe every one of them.

3.1. Interface routines

Subroutine ***USEMCL*** (COM)

INPUT: COM

COM is a character * 10 variable containing an interface command. This routine performs a check to determine whether COM contains a valid command and if so, executes it. The valid interface commands are stored in the character * 10 array MCLCOM through a DATA statement. All commands are described in the MCL manual [2].

The COMMON block MCLVAR contains the arrays STACK, CMEM and the variables ISTACK, INFILE. The STACK array implements the stack structure, while the array CMEM implements the memory structure, both described in the MCL manual. ISTACK is the index of the element at the top of the stack. INFILE determines the input unit.

The DEBVAR integer variable of the COMMON block DEBUG, facilitates the issuing of debugging information upon errors. The character array BUFFER, is used as an OUTPUT buffer. When the buffer is filled, it is flushed automatically to the output unit.

The PARAMETER defined integers, MEM, LSTA, MB and MC, denote the maximum allowed storage for the arrays CMEM, STACK, BUFFER and MCLCOM, while the integer MR is the length for every element of the character array BUFFER. MEM is set equal to 2500 and LSTA to 500. If however an MCL program requires additional storage for either or both of the CMEM and STACK arrays, these values should be set to whatever appropriate.

Subroutine ***FORWAR*** (N, IU)

INPUT:N, IU

Repositions the file connected to the unit IU, by skipping N − 2 records forward.

Subroutine ***BACWAR*** (N, IU)

INPUT:N, IU

Repositions the file connected to the unit IU, by backspacing |N| + 2 records.

Subroutine ***STORIC*** (X, CH, L)

INPUT: X

OUTPUT: CH, L

Stores the real number X in the character variable CH(1:L − 1) and sets the element CH(L:L) equal to a blank space. An F-format is used if X is in the range permitted by the machine's precision, otherwise an E-format is used.

Subroutine ***TREBUF*** (BUFFER, IBUF, LENBUF, CARRY, L, TEMPO, IUOUT)

INPUT: CARRY, L, IUOUT

INPUT/OUTPUT: BUFFER, IBUF, LENBUF, TEMPO

Treats the I/O buffer. Adds to the BUFFER(IBUF), the contents of CARRY (1:L) only if it fits. In the opposite case, fills the BUFFER(IBUF)(LENBUF:) with blanks and stores CARRY (1:L) in the next buffer record BUFFER(IBUF +1), updating IBUF = IBUF + 1. If the buffer was completely filled, then the buffer is flushed to IUOUT, IBUF becomes equal to 1 and CARRY (1:L) is stored at the top of the buffer. TEMPO is the last filled (or partially filled) buffer record.

Subroutine ***FLUSH*** (BUFFER, IBUF, IUOUT)

INPUT: IUOUT

INPUT/OUTPUT: IBUF, BUFFER

Flushes the contents of the character array BUFFER to the unit IUOUT, and resets IBUF to 1.

Function ***ACOSH0*** (X)

INPUT: X

Implementation of the function: arccosh(x). The algorithm used is:

If $x \leq 1$, it is set equal to: $\ln(x + \sqrt{x^2 - 1})$

else, the routine aborts.

Function ***ASINH0*** (X)

INPUT: X

Implementation of the function: arcsinh(x). The formula used is:

$\operatorname{arcsinh}(x) = \ln(x + \sqrt{x^2 + 1})$.

Function ***ATANH0*** (X)

INPUT: X

Implementation of the function: arctanh(x). The algorithm used is:

if $|x| < 1$, is set equal to: $\frac{1}{2} \ln\left(\frac{1+x}{1-x}\right)$

else, the routine aborts.

Double precision function ***FACT*** (N)

INPUT: N

Calculates the N-factorial (N!). For $N \leq 30$ the factorials are stored via a DATA statement. For $N > 30$ they are calculated.

Subroutine ***DETUNT*** (IUOUT, INFILE, IU)

INPUT: IUOUT, INFILE

OUTPUT: IU

Reads a file name from the unit INFILE, and determines an available unit number IU, to connect it to. IUOUT is the output unit.

Subroutine ***STOPER*** (CHARI)

INPUT: CHARI

CHARI is a character variable containing an error message. It is called when an error occurs. Terminates execution of the MCL program and if in IAF option, returns control to MERLIN's operating system, otherwise aborts.

3.2. Other (non-interface) added routines

Subroutine ***SETMOD*** (IUINP, IUOUT, IPROC)

INPUT: IUINP, IUOUT

OUTPUT: IPROC

IUINP, IUOUT are the input, output units. IPROC defines the option of the PROCES mode: 1 for BATCH, 0 for IAF. This routine forces the user to define the desired PROCES option at the beginning of the run.

Subroutine ***UNDEF*** (INDEF, NOV, IUOUT, STRO, IGO)

INPUT: INDEF, NOV, IUOUT, STRO

OUTPUT: IGO

NOV is the dimensionality of the objective function. STRO is a logical variable, used to suppress or not output. IUOUT is the output unit. INDEF(NOV) is an array containing zeros and ones. If INDEF(I) is zero, it means that the variable POINT(I) is indefinite (not assigned a value). If IGO is zero upon return, indicates that there are still indefinite elements in the array POINT(1:NOV).

*Subroutine **AUTIDA*** (IOPEN, IUOUT, IU42)

INPUT: IOPEN, IUOUT, IU42

If IOPEN is 1 or 0, indicates that OPEN statements are used or not used. IUOUT is the output unit. IU42 is the unit associated with the MERLIN's HELP-file. This routine issues information about the program's title, version, implementation date and authors.

*Subroutine **MENU*** (ISET, IUOUT, IUINP)

INPUT: IUOUT, IUINP
OUTPUT: ISET

This routine implements the menu of the new INIT command, that assigns values (residing in a file) to the elements of the arrays POINT, XLL and XRL. IUOUT, IUINP are the output and input units. ISET is -1, 0 or 1, for initializing the XLL, POINT or the XRL array correspondingly.

4. Procedural instructions

A general procedure to be followed is:

a) Construct an MCL program as described in ref. [2].
b) Compile it, using the MCL-compiler and so produce the MCL object code.
c) Construct the input file and possibly files containing initial values.
d) Load and execute MERLIN-2.0 with the embedded objective function.
e) Upon program's termination inspect the output.

5. Test-run description

The test-run is organized in the following way: The objective function used, is the same as that in table 3 of ref. [1]. We list the following files:

1) File: MCLPRO that contains the source MCL program.
2) File: MOC that contains the MCL object code.
3) File: XSTART that contains initial values for the point variables.
4) File: TAPE31 which is the input deck to MERLIN-2.0.
5) File: TAPE32 which is the output produced by MERLIN-2.0.

References

[1] G.A. Evangelakis, J.P. Rizos, I.E. Lagaris and I.N. Demetropoulos, Comput. Phys. Commun. 46 (1987) 401.
[2] C.S. Chassapis, D.G. Papageorgiou and I.E. Lagaris, Comput. Phys. Commun. 52 (1989) 223.

TEST RUN OUTPUT

FILE : TAPE31

```
3
IAF
INIT
0
XSTART
VALDIS
RUNMCL
MOC
STOP
```

FILE : XSTART

```
-112  44  107
```

FILE : MCLPRO

```
> PROGRAM TRIVIAL_ILLUSTRATION
VAR SUM
NOPRINT
BFGS(PRI=0;NOC=1000;CANCEL=0)
SIMPLEX(NOC=1000;CANCEL=0)
FULLPRINT
SIMPLEX
BFGS
LAB:
GRADDIS
SUM=ABS[GRAD[1]]+ABS[GRAD[2]]+ABS[GRAD[3]]
WHEN SUM <1.E-3 JUST MOVE TO ENDIT
SIMPLEX
BFGS
MOVE TO LAB
ENDIT:
SHORTDIS
```

FILE : MOC

```
$LENGTH
 1
NOPRINT
$PUSHC
 6
$PUSHC
 0.
$PUSHC
 1
$PUSHC
 1000.
$PUSHC
 8
$PUSHC
 0.
$WMULTI
  8
BFGS
  3
$SWAP
$PUSHC
 5
$PUSHC
 1000.
$PUSHC
 7
$PUSHC
 0.
$WMULTI
  7
SIMPLEX
  2
$SWAP
FULLPRINT
$WMULTI
  7
SIMPLEX
  0
$SWAP
$WMULTI
  8
BFGS
  0
$SWAP
$CONTINUE
GRADDIS
  0.000
$PUSHC
 1
$PUSHC
 1.
$GRAD
$ABS
$PUSHC
 2.
$GRAD
$ABS
$ADD
$PUSHC
 3.
$GRAD
$ABS
$ADD
$POPCONT
$PUSH
 1
$PUSHC
 .001
$<
$NOT
$TESTMOVE
 3
$MOVE
 13
$WMULTI
  7
SIMPLEX
  0
$SWAP
$WMULTI
  8
BFGS
  0
$SWAP
$MOVE
 -40
$CONTINUE
SHORTDIS
$FINISH
```

FILE : TAPE32

```
 ENTER # OF VARIABLES
  ------------------------------------------------------------------------------

  ........................      MERLIN 2.0 ENHANCED AND PROGRAMMABLE VERSION
  ........................                    IMPLEMENTED BY :
  ........................ C.S. CHASSAPIS, D.G. PAPAGEORGIOU & I.E. LAGARIS
  ........................                    PHYSICS DEPARTMENT
  ........................                UNIVERSITY OF IOANNINA
  ........................            IOANNINA - G R E E C E , 45332

  M E R L I N  2.0
  IOANNINA,  MARCH   1988
  ------------------------------------------------------------------------------
  TO OBTAIN ON-LINE INFORMATION, MAKE SURE THAT
                     FILE:  HELP  , IS PRESENT
  ------------------------------------------------------------------------------
 ESTIMATED MACHINE'S ACCURACY: 1.E-14

------------------------------------------------------------------------------
 ... ENTER PROCESSING MODE OPTION. ( BATCH / IAF )

------------------------------------------------------------------------------


  ....  W A R N I N G  ....
 ...  INITIALIZE VARIABLES  ...


    /\/\/\/\/\/\     MERLIN   IS AT YOUR COMMAND !!!
 INIT
  ------------------------------------------------------------------------
  :                            SELECTION MENU                            :
  ------------------------------------------------------------------------
  :                                  -1 ... FOR LEFT - MARGINS           :
  :                        ENTER :    0 ... FOR    POINT                 :
  :                                  +1 ... FOR RIGHT - MARGINS          :
  ------------------------------------------------------------------------
                 ENTER SELECTION : >>
  ...   ENTER FILE NAME   ...


    /\/\/\/\/\/\     MERLIN   IS AT YOUR COMMAND !!!
 VALDIS
------------------------------------------------------------------------

                 VALUE =   38984672497945.

------------------------------------------------------------------------


    /\/\/\/\/\/\     MERLIN   IS AT YOUR COMMAND !!!
```

```
RUNMCL
  ENTER FILE-NAME
 NOPRINT
 SIMPLEX
        NUMBER OF   SIMPLEX  CALLS:        997
 PANEL-ON
 BFGS
        NUMBER OF      BFGS  CALLS:        132
 PANEL-ON
 GRADDIS
  ERROR BOUND
   1)              -.28421456E-13       0
   2)               .00000000E+00       0
   3)              -.36113928E-12      -1
```

```
SHORTDIS
TOTAL NU. OF CALLS = 1130
NU. OF CALLS SINCE LAST RESET  1130
  1)           .......   3.0000000000000
  2)           .......  -.73101490598310E-19
  3)           .......   .13246762855732E-20
       VALUE.......      .20194839407630E-27


   /\/\/\/\/\/\     MERLIN   IS AT YOUR COMMAND !!!
STOP
```