USER MANUAL
PANMIN: Sequential and Parallel Global Optimization Procedures with a variety of options for the Local Search Strategy.

F. V. Theos, I. E. Lagaris
Department of Computer Science

D. G. Papageorgiou
Department of Materials Science and Engineering

UNIVERSITY OF IOANNINA
P.O.Box 1186, Ioannina 45110 – GREECE

October 16, 2003

http://merlin.cs.uoi.gr/panmin
# Contents

1 **MERLIN Set–Up**
   1.1 **MERLIN** installation ........................................ 3
   1.2 Running **MERLIN** ........................................... 4

2 **Running and customizing the global optimization codes** 5
   2.1 Some general remarks ........................................ 5
   2.2 Running **PRICE** ........................................... 5
   2.3 Customizing **PRICE** ......................................... 7
   2.4 Running **TML** .............................................. 7
   2.5 Customizing **TML** ........................................... 9
   2.6 Running **Parallel–TML (PTML)** ............................. 9
   2.7 Customizing **PTML** .......................................... 10
   2.8 Files common to all methods .................................. 10
      2.8.1 The **MERLIN** input file .............................. 10
      2.8.2 Specifying parameter bounds ............................ 11

3 **Test Runs** 11
   3.1 Six Hump Camel-Back code ................................. 11
   3.2 **PRICE** - Test Run ....................................... 12
   3.3 **TML** - Test Run ........................................ 14
   3.4 **PTML** - Test Run ........................................ 16

4 **Examples** 19
   4.1 Polynomial Root-Finder ..................................... 19
   4.2 Sample code for multistart .................................. 24
   4.3 Molecular conformation problem ............................. 25

5 **Using MCL with the OPTIMA interface** 26

References 30
1 **MERLIN Set-Up**

Before running the global optimization codes, one should install properly the **MERLIN** optimization package [1, 2].

1.1 **MERLIN installation**

The following instructions are valid for installing **MERLIN** under a Unix system.

1. Download the latest **MERLIN** package from [http://merlin.cs.uoi.gr](http://merlin.cs.uoi.gr). The corresponding file will be of the form `merlin-x.y.tar.gz` where `x.y` is the **MERLIN** version.

2. Uncompress and untar the file
   ```
tar xvfz merlin-x.y.tar.gz
   ```
A directory named `merlin-x.y` will be created.

3. Change into the newly created directory
   ```
cd merlin-x.y
   ```

4. Edit `Makefile.inc` and provide appropriate values for the variables **DESTDIR** and **FOPTIONS** (found in the first few lines of `Makefile.inc`).

   **DESTDIR** is the destination directory where the **MERLIN** binaries and accompanying files will be installed. You must specify **DESTDIR** using a full path. Note that this directory must be different from the directory where you unpacked the **MERLIN** sources. Also note that if installing **MERLIN** outside your home directory you must have super-user privileges. Example:
   ```
   DESTDIR=/usr/local/merlin
   ```

   **FOPTIONS** is a string of options that must be passed to the Fortran compiler during the installation process. This is a good place to pass appropriate optimization options to the compiler. Example:
   ```
   FOPTIONS=-O3
   ```

   In addition the following options may be set:
   **LINKOPTIONS**: Options that must be passed to the linker during installation.
   **TYPE**: This specifies whether **MERLIN** will be compiled using REAL or DOUBLE PRECISION arithmetic.
   **MXX**: The maximum number of optimization parameters **MERLIN** will handle.
   **MXT**: The maximum number of terms in a sum-of-squares objective function.
   **MCLBUF**: The maximum size (in bytes) of a compiled **MCL** program.
   **MCLMEM**: The maximum size (in words) of memory available to **MCL** programs.

   The default values provided for **DESTDIR** and **FOPTIONS** are adequate for most systems. Note for Cygwin users: You must set `F77=g77` and uncomment `EXESUFFIX=.exe` in `Makefile.inc`. 

3
5. Build the package
   make

6. Install the package
   make install

1.2 Running MERLIN

In order to compile the user written subprograms with the rest of the MERLIN package one must use the run-merlin script. After installation, the run-merlin script as well as other binaries are located in DESTDIR/bin. Hence one needs to add this directory to its Unix PATH.

To add DESTDIR/bin to your Unix path and assuming that DESTDIR is set to /usr/local/merlin:

- Users of csh and tcsh add the following line in the file .cshrc
  
  path = ( $path /usr/local/merlin/bin )

- Users of sh add the following lines in the file .profile

  PATH=$PATH:/usr/local/merlin/bin
  EXPORT PATH

- Users of bash add the following lines in the file .bashrc

  PATH=$PATH:/usr/local/merlin/bin
  EXPORT PATH

Alternatively you can invoke the run-merlin script using a full path. For example:

/usr/local/merlin/bin/run-merlin funmin.f

The run-merlin script accepts one or more file names as arguments that can be of the following types:

- Files ending in .d
  These are files that must be processed by the MERLIN preprocessor before they can be compiled. The MERLIN preprocessor will use the definitions file DEFS in the current directory, or if no such file exists, the one used for the installation of the MERLIN package (found in DESTDIR/files). After preprocessing the files, the script will compile and link them with the rest of the MERLIN package.

- Files ending in .f
  These are files containing Fortran-77 code that must be compiled.

- Files ending in .o
  These are files containing object code (already compiled).

In addition the run-merlin script recognizes the following environment variables.

4
• **MERLIN_F77**
  This is the name of the compiler that will be used to compile the user written code.

• **MERLIN_FOPTIONS**
  Options that will be used when compiling the user written code. The default is to use `-c` (compile only-do not link) and any other flags that were specified when editing `Makefile.inc`. Note that if you set this environment variable you must include the `-c` compiler flag.

• **MERLIN_LDOPTIONS**
  Options that will be passed to the linker when building the final MERLIN executable. Here you can specify any libraries required by the user written subprograms.

## 2 Running and customizing the global optimization codes

### 2.1 Some general remarks

Before running any of the global optimization codes, make sure you have installed the latest version of the MERLIN optimization package as described in section 1.1.

In addition, you should prepare the objective function following the instructions given in the MERLIN Users Manual. You may prepare either `FUNCTION FUNMIN` or `SUBROUTINE SUBSUM`. Optionally the gradient vector, Hessian and Jacobian matrices may be programmed as well.

### 2.2 Running PRICE

1. Prepare the input file. This file must be named `PRICE.DAT`. A sample file is displayed below.

```
  2        NOD
  0        NOF
10000    NOC
1.00000000000000E-04  EPS
10000.00000000   OMEGA
  20       NF
   1     IPRINT
   1    NFORM
  PRICE.OUT  OUTFIL
  POIMAR.DAT  POIFIL
    in.dat   FINP
    out.dat  FOUT
```
Only the first column is required by the software. The second column contains only code-names that serve as a reminder to the user to ease the data entry.

(a) NOD is the dimensionality of the problem.
(b) NOF is the number of terms when the SUBROUTINE SUBSUM has been prepared.
   If NOF is set to zero, FUNCTION FUNMIN is used.
(c) NOC is an upper bound for the number of function evaluations.
(d) EPS is a small tolerance, used to terminate the search.
(e) OMEGA is the value of the $\omega$ parameter.
(f) NF is an integer used to set the sample size $M$ via the relation: $M = NF \times NOD$
(g) IPRINT is a flag. If it is set to zero, no intermediate printout is produced.
   If it is set to one, informative printout is issued at every iteration.
(h) NFORM controls the format of the final output. Takes on the values 0,1,2 and
   the corresponding formats are described at the end of this section.
(i) OUTFIL is a character string that specifies the name of the output file.
(j) POIFIL is a character string that specifies the name of the file containing an
   initial point, parameter bounds and the fix-status as described in section 2.8.2.
(k) FINP is a character string that specifies the name of the Merlin instructions file
   as described in section 2.8.1. If FINP is left blank then the final polishing is not
   performed.
(l) FOUT is a character string that specifies the name of the Merlin output file.

2. Prepare the MERLIN input file as described in section 2.8.1.

3. Prepare a file containing the parameter bounds as described in section 2.8.2.

4. Compile the program and the objective function, link with the rest of the MERLIN
   package and run (assuming the objective function is in file funmin.f):
      run-merlin price.d funmin.f

5. After the run completes, the results are disposed in the output file. Its file name is
   specified in the input file PRICE.DAT (The default being PRICE.OUT). The output file
   contains the values of the minimization parameters along with the corresponding value
   of the objective function. Three output formats are supported and may be specified
   in the input file (NFORM).
   
- **NFORM=0** Raw unformatted output.
  Output is written using unformatted write statements as in the following program
  segment:

      DO 10,I=1,N
          WRITE (1) X(I)
  10 CONTINUE
      WRITE (1) VALUE
• **NFORM=1** Raw formatted output.
  Output is written using formatted write statements as in the following program segment:
  
  ```fortran
  DO 10, I=1, N
      WRITE (1,20) X(I)
  10   CONTINUE
      WRITE (1,20) VALUE
  20   FORMAT (1PG21.14)
  ```

• **NFORM=2** Output is written as a MERLIN record.
  This is the default.

### 2.3 Customizing PRICE

The following parameter statement is used to customize the code, in accord with the problem at hand.

• **PARAMETER (MAXFAC = 100)**
  This sets the maximum value for the NF factor.

### 2.4 Running TML

1. Prepare the input file. This file must be named **TML.DAT**. A sample file is displayed below.

```
2           NOD
0           NOF
10          ITTHR
5.0000000000000000000 HEAL
2           NEARN
100         NSAMPL
4.0000000000000000000 SIGMA
0.9950000000000000000 COVTHR
0.1000000000000000000E-01 XDEPS
0.1000000000000000000E-03 FDEPS
0.1000000000000000000E-02 GDEPS
0           IREST
100         NDUMP
in.dat      FINP
out.dat     FOUT
MINIMA      FMIN
POIMAR.DAT  POIFIL
input       INDIR
```
Only the first column is required by the software. The second column contains only
code-names that serve as a reminder to the user to ease the data entry.

(a) NOD is the dimensionality of the problem.
(b) NOF is the number of terms, in case where the SUBROUTINE SUBSUM is being
   prepared.
   If NOF is set to zero, FUNCTION FUNMIN is used.
(c) ITTHR is the enforced minimum number of iterations of the algorithm.
(d) HEAL is the healing parameter.
(e) NEARN specifies the number of nearest neighbors
(f) NSAMPL specifies the sample size
(g) SIGMA specifies the σ parameter involved in the calculation of the critical dis-
tance.
(h) COVTHR specifies the required relative domain coverage. (Default value: 0.995)
(i) XDEPS specifies the minimum distance between two minima to be considered as
different.
(j) FDEPS specifies the minimum relative difference in the values of two minima, in
   order to be considered as different.
(k) GDEPS specifies the maximum value for the RMS gradient, in order that a point
   is accepted as a local minimum.
(l) IREST If set to a non-zero value, starts the procedure from a previously saved
dump.
(m) NDUMP Specifies the iteration period between dumps.
(n) FINP is a character string that specifies the name of the Merlin instructions file
   as described in section 2.8.1.
(o) FOUT is a character string that specifies the name of the Merlin output file.
(p) FMIN is a character string that specifies the name of the file containing the
discovered local minima.
(q) POIFIL is a character string that specifies the name of the file containing an
    initial point, parameter bounds and the fix-status as described in section 2.8.2.
(r) INDIR is a character string that specifies the name of a Unix directory, where
    auxiliary files may reside as described in section 2.6. This applies only to the
    parallel version of the software. The sequential version ignores it.

2. Prepare a file containing the parameter bounds as described in section 2.8.2.

3. Prepare the MERLIN input file as described in section 2.8.1.

4. Compile the program and the objective function, link with the rest of the MERLIN
   package and run (assuming the objective function is in file funmin.f): run-merlin
   tml.d funmin.f
5. After the run completes, the results are disposed in the output file. Its file name is specified in the input file TML.DAT, the default being MINIMA). The output file contains the values of the minimization parameters along with the corresponding value of the objective function, and is written as a series of MERLIN records (see the MERLIN Users Manual for a description of MERLIN records).

2.5 Customizing TML

The following parameter statements are used to customize the code, in accord with the problem at hand.

- **PARAMETER (MXNM = 10000)**
  Sets the maximum number of local minima that will be stored. (The program stops when the number of discovered minima becomes equal to $\text{MXNM}$).

- **PARAMETER (MXNS = 500)**
  The maximum number of points to sample at each iteration. (Maximum allowed value for the NSAMPL input parameter).

- **PARAMETER (MXNNN = 8)**
  The maximum number of nearest neighbors considered. (Maximum allowed value for the NEARN input parameter).

2.6 Running Parallel-TML (PTML)

1. Prepare the input file. The file name and contents are the same as for the plain TML code. (section 2.4).

2. Prepare a file containing the parameter bounds as described in section 2.8.2.

3. Prepare the MERLIN input file as described in section 2.8.1.

4. In parallel TML, it is necessary for every processor to operate on a separate directory of the file system, so that intermediate files created by MERLIN and the objective function do not get mixed up. The user must create a directory and copy all input files required by MERLIN and the objective function in this directory. Note that the MERLIN panel description file PDESC must be copied as well.

   ```bash
   mkdir input
   cp PDESC input
   cp an_input_file input
   ```

5. Create temporary directories for all processors and copy the necessary input files (assuming 4 processors).

   ```bash
   create-ptml-dirs 4 input
   ```
6. Compile the program and the objective function and link with MERLIN and the MPI environment (assuming the objective function is in file funmin.f):
   setenv MERLIN_F77 mpif77
   compile-merlin ptml.d funmin.f

7. Run the program (assuming 4 processors):
   mpirun -np 4 merlin.executable

8. The results are disposed in the output file (its file name is specified in the input file TML.DAT, the default being MINIMA).

### 2.7 Customizing PTML

The following parameter statements are used to customize the code, in accord with the problem at hand. These statements reside in the include file ptml.h.

- **PARAMETER (MXPROC = 100)**
  The maximum number of processors that can be utilized.

- **PARAMETER (MXNM = 10000)**
  Sets the maximum number of local minima that will be stored. (The program stops when the number of discovered minima becomes equal to MXNM).

- **PARAMETER (MXNS = 500)**
  The maximum number of points to sample at each iteration. (Maximum allowed value for the NSAMPL input parameter).

- **PARAMETER (MXNNN = 8)**
  The maximum number of nearest neighbors considered. (Maximum allowed value for the NEARN input parameter).

### 2.8 Files common to all methods

#### 2.8.1 The MERLIN input file

The MERLIN input file contains commands that are to be executed by MERLIN in order to perform a local minimization. The name of this file is specified in files PRICE.DAT or TML.DAT according to the method, the default being in.dat. A sample is presented below:

```
anal
bfgs noc 5000
```

Note that it not allowed to change the current point using commands such as point, init etc. Note also that this may be the object file of an MCL program.
2.8.2 Specifying parameter bounds

The name of this file is specified in files PRICE.DAT or TML.DAT according to the method, the default being POIMAR.DAT. Each line in the file corresponds to one of the minimization parameters and must contain 4 values: An initial value for the parameter, the lower and upper bound and an integer specifying whether the parameter is fixed (0) or not (1).

Example (for a two parameter objective function):

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>-5.0</td>
<td>5.0</td>
<td>1</td>
</tr>
<tr>
<td>2.5</td>
<td>-8.0</td>
<td>17.0</td>
<td>1</td>
</tr>
</tbody>
</table>

Note that if you attempt to run any of the global optimization codes without preparing this file, the program will create a sample file with default values and stop. You must then edit the file, specify the correct values for your problem, and rerun the program.

3 Test Runs

We employ for the test runs the “Six-hump Camel-Back” function:

\[ f(x, y) = (4 - 2.1x^2 + \frac{1}{3}x^4)x^2 - 4(1 - y^2)y^2 + xy \]

This function has six local minima, two of which are global. The code for the function and its gradient is given in section 3.1. The PRICE and the TML results are presented in sections 3.2 and 3.3 correspondingly. In section 4.2 we list a code that makes use of the subroutine OPTIMA to search for the global minimum of the same test function. The simplistic multistart approach is being used, i.e. we generate a number of points at random and from each one we start a local search.

3.1 Six Hump Camel-Back code

The code for the “Six-hump Camel-Back”, its gradient and its Hessian, is given below:

```c
C -----------------------------------------------------------------------------
C FUNCTION FUNMIN ( X,N )
C -----------------------------------------------------------------------------
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
DIMENSION X(N)
X1 = X(1)
X2 = X(2)
FUNMIN = (4 - 2.1*X1**2 + X1**4/3)*X1**2 +
& X1*X2 + (-4 + 4*X2**2)*X2**2
END
```
C SUBROUTINE GRANAL ( N, X, GRAD )
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
DIMENSION X(N), GRAD(N)
GRAD(1) = 8*X(1) - 8.4*X(1)**3 + 2*X(1)**5 + X(2)
GRAD(2) = X(1) - 8*X(2) + 16*X(2)**3
END

C SUBROUTINE HANAL ( H, LD, N, X )
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
DIMENSION H(LD,N), X(N)
H(1,1) = 8 - 25.2*X(1)**2 + 10*X(1)**4
H(2,1) = 1
H(2,2) = 48*X(2)**2 - 8
END

3.2 PRICE - Test Run

To execute the test run, issue the following command:
run-merlin price.d 6hump.f
The following output is produced on the standard output.

-----------------------------------
PRICE running with ...
Number of parameters: 2
Maximum function calls: 10000
Termination criterion: 1.0000E-04
Weighting factor: 1.0000E+04
Sample size factor: 20
Printout option: 1
Output file: PRICE.OUT
Bounds file: P0IMAR.DAT
Merlin input file: in.dat
Merlin output file: out.dat
Output format: 1
-----------------------------------
Iter: 30 Lower value: -0.14170040690117 Calls: 82 of 10000
Iter: 35 Lower value: -0.22434743776021 Calls: 92 of 10000
Iter: 48 Lower value: -0.64718662239411 Calls: 111 of 10000
Iter: 115 Lower value: -0.72788366844541 Calls: 199 of 10000
Iter: 140 Lower value: -0.83495615401031 Calls: 239 of 10000
Iter: 141 Lower value: -1.0275395980524 Calls: 241 of 10000

12
Iter: 211  Lower value: -1.0277942972752  Calls: 352 of 10000
Iter: 239  Lower value: -1.0284090174387  Calls: 390 of 10000
Iter: 257  Lower value: -1.0313613560203  Calls: 408 of 10000
Iter: 312  Lower value: -1.0313833436910  Calls: 463 of 10000
Iter: 335  Lower value: -1.0316284204941  Calls: 486 of 10000

The termination criterion has been satisfied
Function value: -1.0316284534837
Total function evaluations: 661
PRICE Iterations: 480
The minimum has been refined by Merlin, GRMS = 3.92365572601884E-10
Output parameters in file: PRICE.OUT

The relevant input files are listed:
File: POIMAR.DAT

0.0  -5.0  5.0  1
0.0  -5.0  5.0  1

File: PRICE.DAT

2       NOD
10000   NOC
1.00000000000000E-04  EPS
10000.00000000   OMEGA
20      NF
1       IPRINT
1       NFOM
PRICE.OUT  OUTFIL
POIMAR.DAT  POIFIL
in.dat     in.dat
out.dat    out.dat

File: in.dat

ANAL
BFGS NOC 2000

File: PRICE.OUT (Contains the global minimizer).

8.98420130599726E-02
-0.71265640298440
-1.0316284534837
3.3 TML - Test Run

To execute the test run, issue the following command:
run-merlin tml.d 6hump.f
The following output is produced on the standard output.

TML running with ...
Iteration threshold: 10
Healing parameter: 5.
Number of nearest neighbors: 2
Sample size: 100
Sigma parameter: 4.
Coverage threshold: 0.995000005
X distance criterion: 0.01
F distance criterion: 0.0001
Gradient convergence criterion: 0.001
Saving every 100 iterations
Merlin input file: in.dat
Merlin output file: out.dat

... The first minimum is found ...  F: 136  G: 10
Iterations: 1  Coverage: 0.79865  Est. minima: 3.9
Iterations: 2  Coverage: 0.96460  Est. minima: 1.4
... Number of minima found: 2  F: 361  G: 32
... Number of minima found: 3  F: 375  G: 46
Iterations: 3  Coverage: 0.95274  Est. minima: 4.0
... Number of minima found: 4  F: 489  G: 60
Iterations: 4  Coverage: 0.97160  Est. minima: 5.0
... Number of minima found: 5  F: 656  G: 124
Iterations: 5  Coverage: 0.98355  Est. minima: 5.8
Iterations: 6  Coverage: 0.99071  Est. minima: 5.6
... Number of minima found: 6  F: 1042  G: 299
Iterations: 7  Coverage: 0.99201  Est. minima: 6.6
Iterations: 8  Coverage: 0.99516  Est. minima: 6.5
Iterations: 9  Coverage: 0.99674  Est. minima: 6.4
Iterations: 10  Coverage: 0.99769  Est. minima: 6.3

TML run completed.
Number of local minimizers found: 6
The minimizers are disposed to file: MINIMA
Total number of Function calls: 1914
Total number of Gradient calls: 838
Total number of Jacobian calls: 0
Total number of Hessian calls: 0
Total number of local optimizations : 51
Maximum number of starting points: 13
Current state has been saved.

The relevant input files are listed:

File: POIMAR.DAT

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>-5.0</td>
<td>5.0</td>
<td>1</td>
</tr>
<tr>
<td>0.0</td>
<td>-5.0</td>
<td>5.0</td>
<td>1</td>
</tr>
</tbody>
</table>

File: TML.DAT

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>NOD</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>NOF</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>ITTHR</td>
<td></td>
</tr>
<tr>
<td>5.00000000000000E+00</td>
<td>HEAL</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>NEARN</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>NSAMPL</td>
<td></td>
</tr>
<tr>
<td>4.00000000000000E+00</td>
<td>SIGMA</td>
<td></td>
</tr>
<tr>
<td>0.995000000476837</td>
<td>COVTHR</td>
<td></td>
</tr>
<tr>
<td>0.10000000000000E-01</td>
<td>XDEPS</td>
<td></td>
</tr>
<tr>
<td>0.10000000000000E-03</td>
<td>FDEPS</td>
<td></td>
</tr>
<tr>
<td>0.10000000000000E-02</td>
<td>GDEPS</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>IREST</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>NDUMP</td>
<td></td>
</tr>
</tbody>
</table>

in.dat
out.dat
MINIMA
POIMAR.DAT
input

File: in.dat

ANAL
BFGS NOC 2000

File: MINIMA (Contains the minimizers found).

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1)</td>
<td>.......</td>
<td>8.9842012988325989E-02</td>
</tr>
<tr>
<td>2)</td>
<td>.......</td>
<td>-0.7126564030053476</td>
</tr>
</tbody>
</table>

Value ....... -1.031628453483664
1) ....... -8.9842001740361430E-02 | -5.000 | 5.000 |
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2)</td>
<td>.......</td>
<td>0.7126564016386691</td>
<td>-5.000</td>
<td>5.000</td>
</tr>
<tr>
<td>Value</td>
<td>.......</td>
<td>-1.031628453483664</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1)</td>
<td>.......</td>
<td>1.703606615756318</td>
<td>-5.000</td>
<td>5.000</td>
</tr>
<tr>
<td>2)</td>
<td>.......</td>
<td>-0.796083641969248</td>
<td>-5.000</td>
<td>5.000</td>
</tr>
<tr>
<td>Value</td>
<td>.......</td>
<td>-0.2154630210843795</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1)</td>
<td>.......</td>
<td>-1.703606616107386</td>
<td>-5.000</td>
<td>5.000</td>
</tr>
<tr>
<td>2)</td>
<td>.......</td>
<td>0.796083644085253</td>
<td>-5.000</td>
<td>5.000</td>
</tr>
<tr>
<td>Value</td>
<td>.......</td>
<td>-0.2154630210843795</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1)</td>
<td>.......</td>
<td>-1.607104586736009</td>
<td>-5.000</td>
<td>5.000</td>
</tr>
<tr>
<td>2)</td>
<td>.......</td>
<td>-0.5686514769482340</td>
<td>-5.000</td>
<td>5.000</td>
</tr>
<tr>
<td>Value</td>
<td>.......</td>
<td>2.104250946486463</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1)</td>
<td>.......</td>
<td>1.607104589746949</td>
<td>-5.000</td>
<td>5.000</td>
</tr>
<tr>
<td>2)</td>
<td>.......</td>
<td>0.5686514782893214</td>
<td>-5.000</td>
<td>5.000</td>
</tr>
<tr>
<td>Value</td>
<td>.......</td>
<td>2.104250946486463</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 3.4 PTML - Test Run

The test run assumes that LAM-MPI is installed and operational. The test run uses 4 processors. To execute the test run, issue the following commands:

```bash
../src/create-ptml-dirs 4 indir
setenv MERLIN_F77 mpif77
compile-merlin ptml.d 6hump.f
lamboot
mpirun -np 4 merlin.executable
wipe
```

The following output is produced on the standard output.

```
Parallel TML running with ... 
Iteration threshold: 10
Healing parameter: 5.00000000000000 
Number of nearest neighbors: 2
Sample size: 100
Sigma parameter: 4.00000000000000
Coverage threshold: 0.995000000476837
X distance criterion: 1.00000000000000D-02
F distance criterion: 1.00000000000000D-04
Gradient convergence criterion: 1.00000000000000D-03
Saving every 100 iterations
Merlin input file: in.dat
Merlin output file: out.dat
Directory containing input files: indir
```

16
... The first minimum is found ...  F:  136  G:  9
Iterations: 1  Coverage: 0.6350206  Est. minima: 2.0
Iterations: 2  Coverage: 0.9556767  Est. minima: 1.5
... Number of minima found:  2  F:  379  G:  31
... Number of minima found:  3  F:  407  G:  47
Iterations: 3  Coverage: 0.9476247  Est. minima: 4.1
... Number of minima found:  4  F:  546  G:  75
Iterations: 4  Coverage: 0.9706831  Est. minima: 5.0
... Number of minima found:  5  F:  723  G:  130
Iterations: 5  Coverage: 0.9820669  Est. minima: 5.9
... Number of minima found:  6  F:  1039  G:  282
Iterations: 6  Coverage: 0.9870537  Est. minima: 6.8
Iterations: 7  Coverage: 0.9917658  Est. minima: 6.7
Iterations: 8  Coverage: 0.9953168  Est. minima: 6.5
Iterations: 9  Coverage: 0.9968641  Est. minima: 6.4
Iterations: 10  Coverage: 0.9978384  Est. minima: 6.3

PTML run completed.
Number of local minimizers found:  6
The minimizers are disposed to file: MINIMA
Total number of Function calls:  2400
Total number of Gradient calls:  986
Total number of Jacobian calls:  0
Total number of Hessian calls :  0
Total number of local optimizations :  0
Maximum number of starting points:  13
Current state has been saved.

Processor Utilization

<table>
<thead>
<tr>
<th>Proc</th>
<th>Minim</th>
<th>Graph</th>
<th>Fcalls</th>
<th>Gcalls</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1246</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>346</td>
<td>375</td>
<td>318</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>346</td>
<td>374</td>
<td>320</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>346</td>
<td>405</td>
<td>348</td>
</tr>
</tbody>
</table>

The relevant input files are listed:

File: FOIMAR.DAT

0.0  -5.0  5.0  1
0.0  -5.0  5.0  1

File: TML.DAT

2  NO D
0            NOF
10           ITTHR
5.0000000000000000 HEAL
  2           NEARN
100          NSAMPLE
4.0000000000000000 SIGMA
0.99500000476837    COVTHR
0.1000000000000000E-01    XDEPS
0.1000000000000000E-03    FDEPS
0.1000000000000000E-02    GDEPS
  0           IREST
100          NDUMP
in.dat            FINP
out.dat           FOUT
MINIMA           FMIN
POIMAR.DAT        POIFIL
indir            INDIR

File: in.dat

ANAL
BFGS NOC 2000

File: MINIMA (Contains the minimizers found).

2

1)       8.9842012988325989E-02  -5.000  5.000
2)       -0.7126564030053476  -5.000  5.000

Value       -1.031628453483664
1)       -8.9842001740361430E-02  -5.000  5.000
2)       0.7126564016386691  -5.000  5.000

Value       -1.031628453483664
1)       1.703606615756318  -5.000  5.000
2)       -0.7960835641969248  -5.000  5.000

Value       -0.2154630210843795
1)       -1.703606616107386  -5.000  5.000
2)       0.7960835644085253  -5.000  5.000

Value       -0.2154630210843795
1)       1.607104589746949  -5.000  5.000
2)       0.5686514782893214  -5.000  5.000

Value       2.104250946486463
1)       -1.607104586736009  -5.000  5.000
2)       -0.5686514769482340  -5.000  5.000

Value       2.104250946486463
4 Examples

4.1 Polynomial Root-Finder

As an example of an application of `OPTIMA`, we present a polynomial root finder. The method we use is based on Bairstow’s algorithm and is briefly described here. Let \( P_n(x) = \sum_{k=0}^{n} a_k x^k \) be a polynomial of degree \( n \) with real coefficients \( a_k, \ k = 0, 1, 2, \ldots, n \). The division with a quadratic polynomial

\[
P_n(x) = (x^2 + p_1 x + p_2) B_{n-2}(x, p_1, p_2) + R(p_1, p_2)x + Q(p_1, p_2)
\]

(1)
can be made perfect, by choosing properly \( p_1 \) and \( p_2 \) so as to make the remainder terms \( R \) and \( Q \) vanish. (The coefficients \( b_k, k = 0, 1, \cdots, n - 2 \) of \( B_{n-2} \) as well as \( R \) and \( Q \), are obtained by synthetic division, i.e. via a recursion relation). If we determine such values for \( p_1 \) and \( p_2 \) then \( P_n(x) \) has two roots that coincide with the roots of the quadratic. The same procedure is then applied to the quotient polynomial \( B_{n-2} \) and so on so forth. One way to search for proper values of \( p_1 \) and \( p_2 \) is by optimizing the quantity: \( R^2 + Q^2 \). It is here where `OPTIMA` can be used. We list the source code in Fortran 77.

```
PROGRAM ROOTF
  *
  * This is a polynomial root finder program.
  * the polynomials must have real coefficients.
  * The method applied is a modification of bairstow’s
  * i.e. division by a quadratic polynomial x**2 +p1*x +p2
  * determining p1 & p2 to be such that the remainder is vanishing.
  * Then a deflation is applied by synthetic division to reduce
  * the polynomial degree by two, and the procedure is repeated.
  * A check is always made if the degree is reduced to either
  * one or two to use special finishing procedure.
  *
  IMPLICIT DOUBLE PRECISION (A-H,O-Z)
  COMMON /TROOT/ A(0:100), B(0:100), KCD
  COMPLEX ROOT(100), DISC
  PARAMETER ( N = 2, M = 2 )
  DIMENSION ICODE(4), XP(2), XLL(2), XRL(2), IXAT(2)
  CHARACTER LINE*78
  DATA ICODE / 1, 0, 0, 0 /
  *
  DO 100 I=1,78
    LINE(I:I) = '='
  100  CONTINUE
  *
  * Read in, tolerance, polynomial degree, polynomial coefficients
  WRITE (*,*) 'Enter a termination tolerance: '
```

19
READ (*,*) EPS
WRITE (*,*) 'Enter the degree of the polynomial:'
READ (*,*) KCO
WRITE (*,*) 'Enter the polynomial coefficients: (a(0),a(1),...)
READ (*,*) (A(I),I=0,KCO)

* Type the above input for reference.
WRITE (*,'(A)') LINE
WRITE (*,' Running with the following input ')
WRITE (*,'Tolerance: ', eps
WRITE (*,' Degree: ', kco
WRITE (*,' Coefficients a(i), i=0,1,... ')
WRITE (*,'(A),I=0,KCO)
WRITE (*,'(A)') LINE
NOR = 0
30 CONTINUE
* Check if the degree is one, and if so solve a linear equation.
IF (KCO.EQ.1) THEN
  WRITE (*,*)' The roots of the polynomial are:'
  NOR = NOR + 1
  ROOT(NOR) = CMPLX(-A(0)/A(1),0)
  DO 1 I=1,NOR
    WRITE (*,19) I, ROOT(I)
 1 CONTINUE
  FORMAT (2X,'Root #: ',I4,T17,'(','G14.7,',',G14.7,')')
STOP
END IF
* Check if the degree is two, and if so solve a quadratic equation.
IF (KCO.EQ.2) THEN
  WRITE (*,*)' The roots of the polynomial are:'
  NOR = NOR + 1
  DISC = CMPLX(A(1)**2 - 4*A(2)*A(0),0.D0)
  ROOT(NOR) = (-A(1)+CSQRT(DISC))/2./A(2)
  NOR = NOR + 1
  ROOT(NOR) = (-A(1)-CSQRT(DISC))/2./A(2)
  DO 2 I=1,NOR
    WRITE (*,19) I, ROOT(I)
 2 CONTINUE
STOP
END IF

* Initialize randomly in (-1,1)
P1 = 2*RANM() - 1
P2 = 2*RANM() - 1
XP(1) = P1
XP(2) = P2

* Minimize the remainder down to zero !!!
CALL OPTIMA(N,M,XP,XV,XLL,XRL,IXAT,ICODE,
&        'in.dat','/dev/null',GRMS,NF,NG,NHE,NJA)
IF (SQRT(XV).LE.EPS) THEN
  P1 = XP(1)
  P2 = XP(2)
  DISC = CMPLX( P1**2-4*P2, 0.D0)
  NOR = NOR + 1
  ROOT(NOR) = (-P1 + CSQRT(DISC))/2
  NOR = NOR + 1
  ROOT(NOR) = (-P1 - CSQRT(DISC))/2
  CALL SYNDIV(A,KCO,P1,P2,B,R,Q)
  KCO = KCO - 2
  DO 3 I = 0,KCO
     A(I) = B(I)
  3 CONTINUE
END IF
GO TO 30
END

* SUBROUTINE SYNDIV ( A, N, P1, P2, B, R, Q )
*------------------------------------------------------------------
* IMPLICIT DOUBLE PRECISION (A-H,O-Z)
*
* Given a polynomial: a(n)*x**n + ...+a(1)*x +a(0)
* and a quadratic: x**2 + p1*x + p2
* calculate the coefficients of the quotient polynomial: b(k),k=0,1,...,n-2
* and the remainder terms r and q.
*
* DIMENSION A(0:*), B(0:*)
*
* B(N-2) = A(N)
* B(N-3) = A(N-1)-P1*B(N-2)
* DO 1 K=N-2,2,-1
*     B(K-2) = A(K)-P1*B(K-1)-P2*B(K)
  1 CONTINUE
* R = A(1) - P1*B(0)-P2*B(1)
* Q = A(0) - P2*B(0)
END
*------------------------------------------------------------------
* SUBROUTINE SYNDER (A, N, P1, P2, B, R, Q, B1, B2, R1, R2, Q1, Q2)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)

* Given a polynomial: \( a(n)*x^n + \ldots + a(1)*x + a(0) \)
* and a quadratic: \( x^2 + p1*x + p2 \)
* calculate the coefficients of the quotient polynomial: \( b(k), k=0,1,\ldots,n-2 \)
* and the remainder terms \( r \) and \( q \).
* Calculate the derivatives: \( b1(k), r1, q1 \) of \( b(k) \), \( r \) and \( q \) with respect to \( p1 \),
* and the derivatives: \( b2(k), r2, q2 \) of \( b(k) \), \( r \) and \( q \) with respect to \( p2 \).

DIMENSION A(0:*), B(0:*), B1(0:*), B2(0:*)

B(N-2) = A(N)
B(N-3) = A(N-1) - P1*B(N-2)
DO 1 K=N-2,2,-1
   B(K-2) = A(K) - P1*B(K-1) - P2*B(K)
1 CONTINUE
R = A(1) - P1*B(0) - P2*B(1)
Q = A(0) - P2*B(0)

B1(N-2) = 0
B1(N-3) = -A(N)
DO 2 K=N-2,2,-1
   B1(K-2) = -B(K-1) - P1*B1(K-1) - P2*B1(K)
2 CONTINUE
R1 = -B(0) - P1*B1(0) - P2*B1(1)
Q1 = -P2*B1(0)

B2(N-2) = 0
B2(N-3) = 0
DO 3 K=N-2,2,-1
   B2(K-2) = -P1*B2(K-1) - B(K) - P2*B2(K)
3 CONTINUE
R2 = -P1*B2(0) - B(1) - P2*B2(1)
Q2 = -B(0) - P2*B2(0)
END

SUBROUTINE SUBSUM ( M, N, X, F )
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
DIMENSION X(N), F(M)
COMMON /TOROOT/ A(0:100), B(0:100), KCO
P1 = X(1)
P2 = X(2)
CALL SYNDIV(A,KCO,P1,P2,B,R,Q)
F(1) = R
F(2) = Q
END

* --------------------------------------------------

SUBROUTINE GRANAL ( N, X, GRAD )
* --------------------------------------------------

IMPLICIT DOUBLE PRECISION (A-H,0-Z)
DIMENSION X(N), GRAD(N)
COMMON /TOROOT/ A(0:100), B(0:100), KCO
DIMENSION B1(0:100), B2(0:100)
P1 = X(1)
P2 = X(2)
CALL SYNDER(A,KCO,P1,P2,B,R,Q,B1,B2,R1,R2,Q1,Q2)
GRAD(1) = 2*(R*R1 + Q*Q1)
GRAD(2) = 2*(R*R2 + Q*Q2)
END

* --------------------------------------------------

SUBROUTINE JANAL ( M, N, X, FJ, LD )
* --------------------------------------------------

IMPLICIT DOUBLE PRECISION (A-H,0-Z)
DIMENSION X(N), FJ(LD,N)
COMMON /TOROOT/ A(0:100), B(0:100), KCO
DIMENSION B1(0:100), B2(0:100)
P1 = X(1)
P2 = X(2)
CALL SYNDER(A,KCO,P1,P2,B,R,Q,B1,B2,R1,R2,Q1,Q2)
FJ(1,1) = R1
FJ(1,2) = R2
FJ(2,1) = Q1
FJ(2,2) = Q2
END

We list the Input/Output of a run for obtaining the roots of the polynomial: \( x^6 - 4x^5 + 2x^4 + 8x^3 - 7x^2 - 4x + 4 \) which is an expansion of: \( (x + 1)^2(x - 1)^2(x - 2)^2 \)

Enter a termination tolerance:
1.e-12
Enter the degree of the polynomial:
6
Enter the polynomial coefficients: \( a(0), a(1), \ldots \)

Running with the following input
Tolerance:  1.E-12
Degree:  6
   Coefficients a(i), i=0,1,...
=================================================================

The roots of the polynomial are:
   root #  1  ( 1.000000 ,  0.000000 )
   root #  2  ( -1.000000 ,  0.000000 )
   root #  3  (  1.000000 ,  0.000000 )
   root #  4  ( -1.000000 ,  0.000000 )
   root #  5  (  2.000000 ,  0.000000 )
   root #  6  (  2.000000 ,  0.000000 )

The input instructions to OPTIMA that reside in the file in.dat are as:

ANAL
JANAL
LEVE NOC 1000
TOLMIN NOC 1000

4.2 Sample code for multistart

   PROGRAM MSTART
   *
   *  Sample code for multistart.
   *  Illustrates the use of subroutine optima.
   *
   IMPLICIT DOUBLE PRECISION (A-H,O-Z)
   PARAMETER ( N = 2, M = 0 )
   DIMENSION XP(N), XLL(N), XRL(N), IXAT(N), ICODE(4)
   CHARACTER*10 FINP, FOUT
   *
   * The input Merlin-instructions reside in file: in.dat
   * Merlin's output will be disposed to: /dev/null    (Trash file)
   DATA FINP / 'in.dat' /
   DATA FOUT / '/dev/null' /
   *
   * Instruct Merlin to initialize the parameters from array XP,
   * and to ignore the input for lower bounds, upper bounds and fix-status.
   DATA ICODE / 1, 0, 0, 0 /
   *
   DO 2 J=1,10
   * Fill the XP array with random numbers in the range (-7,7)
   * Routine RANM() is provided by Merlin.

24
DO 1 I=1,N
   XP(I) = 7*(2*RANM()-1)
1 CONTINUE
* Perform local optimization.
   CALL OPTIMA(N,M,XP,XV,XLL,XRL,IXAT,ICODE,
   & FINP,FOUT,GRMS,NF,NG,NH,NJ)
   IF (J.EQ.1) WRITE (*,70)
      WRITE (*,71) J, NF, NG, XV, (XP(I),I=1,N), GRMS
2 CONTINUE
70 FORMAT (3X,'J',4X,'FE',4X,'GE',7X,'VAL',11X,'.... XP ....',
   & 10X,'GRMS')
71 FORMAT (1X,I3,1X,I5,1X,I5,2X,D12.5,1X,10(2X,D12.5))
END

The produced output is displayed below:

<table>
<thead>
<tr>
<th></th>
<th>J</th>
<th>FE</th>
<th>GE</th>
<th>VAL</th>
<th>.... XP ....</th>
<th>GRMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>47</td>
<td>13</td>
<td>-0.10316E+01</td>
<td>0.89842E-01</td>
<td>-0.71266E+00</td>
<td>0.89234E-08</td>
</tr>
<tr>
<td>2</td>
<td>28</td>
<td>19</td>
<td>-0.21546E+00</td>
<td>-0.17036E+01</td>
<td>0.79608E+00</td>
<td>0.49642E-08</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>10</td>
<td>-0.10316E+01</td>
<td>0.89842E-01</td>
<td>-0.71266E+00</td>
<td>0.81464E-09</td>
</tr>
<tr>
<td>4</td>
<td>22</td>
<td>17</td>
<td>-0.10316E+01</td>
<td>-0.89842E-01</td>
<td>0.71266E+00</td>
<td>0.34503E-12</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>24</td>
<td>-0.89842E-01</td>
<td>0.71266E+00</td>
<td>0.43907E-07</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>26</td>
<td>18</td>
<td>-0.21546E+00</td>
<td>0.17036E+01</td>
<td>-0.79608E+00</td>
<td>0.36890E-07</td>
</tr>
<tr>
<td>7</td>
<td>33</td>
<td>19</td>
<td>-0.10316E+01</td>
<td>-0.89842E-01</td>
<td>0.71266E+00</td>
<td>0.36883E-07</td>
</tr>
<tr>
<td>8</td>
<td>37</td>
<td>25</td>
<td>0.21043E+01</td>
<td>0.16071E+01</td>
<td>0.56865E+00</td>
<td>0.23094E-07</td>
</tr>
<tr>
<td>9</td>
<td>28</td>
<td>21</td>
<td>0.21043E+01</td>
<td>-0.16071E+01</td>
<td>-0.56865E+00</td>
<td>0.27245E-08</td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>12</td>
<td>-0.10316E+01</td>
<td>-0.89842E-01</td>
<td>0.71266E+00</td>
<td>0.24530E-07</td>
</tr>
</tbody>
</table>

The file: in.dat that contains the Merlin instructions and specifies the method(s) used for the local search is displayed below.

ANAL
DFP NOC 2000

The above is produced with the DFP method (Davidon, Fletcher, Powell). To change the minimization method to another, say to the BFGS method, edit the in.dat file and change the second line to:

BFGS NOC 2000

4.3 Molecular conformation problem

The geometrical structure is an important property for understanding and predicting the behavior of any molecular system. It is the necessary starting point for the derivation of
structural features, the estimation of steric requirements, and the calculation of electronic properties. For systems with many degrees of freedom, the potential energy hypersurface may have a substantial number of local minima that correspond to stable molecular configurations. Triglycerides are important biological compounds. Among other functions they serve as structural components of the cell membranes and as a source of carbon atoms for biosynthetic reactions. Glycerol triacetate also known as triacetin is a highly flexible molecule, hence its properties do not depend only on the globally optimal conformational state.

Triacetin consists of 29 atoms which form three branches named α, β and γ as shown in fig. 1. The 15 heavy atoms form a backbone structure with 8 rotational degrees of freedom, while the rotation of the terminal methyl groups add 3 more. For the potential function we used the MM2 [3] force field with 1991 parameters, as implemented in the Tinker [4] molecular modeling package. Triacetin was modelled using internal coordinates. Bond lengths and angles were fixed to their equilibrium values while the 11 dihedral angles were allowed to vary. We applied the parallel TML algorithm using 4 processors on a SUN Enterprise 450, with the following default parameter values:

```
ITTHR=10  HEAL=5  NEARN=2  NSAMPL=100  SIGMA=4
COVTHR=0.995  XDEPS=1.e-2  FDEPS=1.e-4  GDEPS=1.e-5
```

For the local optimization we used the TOLMIN method with a maximum of 5000 function calls. The TML algorithm performed 29780 iterations, 3982011 function and 993272 gradient calls. A total of 2181 local minima were found. After TML completed all minima were further refined using the full set of internal coordinates. Using this procedure we recovered all 109 low energy conformers described in [5]. The global minimum is shown in fig. 2, while the next lowest is shown in fig. 3.

## 5 Using MCL with the OPTIMA interface

We present a general strategy, coded in MCL, and then show how it can be used with the OPTIMA interface.

```
program
var geps;nocalls
geps = 1.e-5
nocalls = 1000

call local(geps;nocalls)
end
% ====================================================================
  sub local(geps;nocalls)
% ====================================================================
% It is a local minimization strategy.
```
Figure 1: The triacetin molecule

Figure 2: Lowest energy conformer of triacetin
Figure 3: The second lowest energy triacetin conformer

% GEPS is input and is a tolerance for the rms gradient.
% NOCALLS is input and adjusts the number of calls to the objective
% function.
% Care is taken to use proper methods depending on the existance
% of analytic derivatives, jacobian and the functional form.

var ic;z
ic = 0

REDO:
if funmode == 0 then  % General form (FUNCTION FUNMIN)
  if deriva == 1 then  % Analytic gradient exists.
    tolmin(NOC=nocalls)
    when grms[z] > geps just simplex(NOC=nocalls)
  %
  else          % Analytic gradient does not exist
    simplex(NOC=nocalls)
    when grms[z] > geps just bfgs(NOC=nocalls)
  endif
else               % Sum of Squares form (SUBSUM)
  if jacomo == 1 then % Analytic Jacobian exists


```plaintext
leve(NOC=nocalls)
    when grms[z] > geps just tolmin(NOC=nocalls)
    
else
    % Numerically estimated Jacobian
    leve(NOC=nocalls)
    when grms[z] > geps just simplex(NOC=nocalls)
    endif
endif

ic = ic + 1

if grms[z] > geps then
    when ic <= 10 just move to REDO
endif
end

Compile the above MCL program and let the object code be the input file (FINP) to subroutine optima.

```

29
References


