QUICK GUIDE ON HOW TO EVOLVE METAMODELS USING HyGP (sequential or parallel execution, on a single machine)

To evolve metamodels using HyGP the following steps you need to:

0. Compile HyGP source code
1. Create two input files with the building and test data sets
2. Launch a HyGP experiment, made of a given number of independent runs
3. Harvest the best metamodel evolved and experiment statistical data

Please mind that HyGP was developed **to be run on Linux machines**.

0 - HyGP source code compilation

Clean previous versions, in case:

```
>> make clean
```

Edit the makefile uncommenting the proper line in object *master.o* to compile either the sequential or parallel version:

- parallel execution with OpenMP:

```
# parallel compilation (OpenMP)
g++ -c -g parallel_master.cpp -o master.o -fopenmp
```

- sequential execution:

```
# normal compilation
g++ -c -g master.cpp -o master.o
```

Compile HyGP:

```
>> make
```

The following messages should appear:

```
# parallel compilation
g++ -c -g parallel_master.cpp -o master.o -fopenmp
# normal compilation
# g++ -c -g master.cpp -o master.o
In file included from master.cpp:376:0:
./genetic_code/classes/problem_definition.cpp: In constructor
'ProblemDefinition::ProblemDefinition()':
./genetic_code/classes/problem_definition.cpp:38:15: warning: deprecated conversion
from string constant to 'char*' [-Wwrite-strings]
  constraints0 = ""; //previous version
               ^
./genetic_code/classes/problem_definition.cpp:43:15: warning: deprecated conversion
from string constant to 'char*' [-Wwrite-strings]
  constraints1 = ""; //previous version
```

```
                ^
gfortran -fsecond-underscore -c ./genetic_code/SQP/MI0L2_c/TI0L2.FOR  -o T.o
gfortran -std='legacy' -c ./genetic_code/SQP/MI0L2_c/MI0L2.FOR -o M.o
gfortran -std='legacy' -o gp M.o T.o master.o -L/usr/lib/gcc/x86_64-redhat-linux/4.0.2/
-lstdc++ -fopenmp
```

Being some of them just warnings, the code should compile with no specific issues.

1 – Input file formatting

HyGP needs three kind of data sets to work:

- hyperparameters defining the properties of the evolution
- building data set
- test data set

The hyperparameters and the building data set are defined in the primary input file, whereas the test data set is stored in the secondary input file. Both are simple .txt files. The primary input file has the following structure:

```
# hyperparameters
SEED= -1
NVAR= 8
MINRAND= -200
MAXRAND= 200
MAX_N_PERIODS= 1.5
STEP= .001
NFITCASES= 400
METHOD= 4
DEPTH_MAX= 4
DEPTH_MIN= 2
DEPTH_LIM= 30
p_FULL= .5
REPR_RATE= .2
CROSS_RATE= .4
MUT_RATE= .4
COMP_RATE= .0
NEW_RATE= .0
M= 300
G= 50
NORMALISED_ERR= 0
MINMAX= 0
W_COMPLEXITY= 0.0001
W_N_CORRECTIONS= 0.1
W_SIZE= 0.000001
W_FACTORISATION= 1
N_GUESSES= 2
CROSSVALIDATION= 0
FOLDS_N= 0
THRESHOLD= 1.0E-6
N_INEQUALITY0= 0
W_PEN_ORD0= 0.0
N_INEQUALITY1= 0
W_PEN_ORD1= 0.0
# operators (unary and binary)
# shift,square,cube,exp,sin,cos,log,sinh,cosh,tanh
# add,sub,mult,sdiv
BINARY_FUN= add,sub,mult,sdiv
UNARY_FUN= shift,square,cube,sin,cos,exp,sinh,cosh,tanh
# building data matrix (number of records x number of variables + 1)
#      Z1     Z2     Z3     Z4     Z5     Z6     Z7     Z8     TARGET
0.11   0.22   0.33   0.44   0.55   0.66   0.77   0.88   999.999999  (first record or building point)
… (second record or building point)
```

The basic parameters to set up to launch an experiment are:

`NVAR= :` number of variables
`NFITCASES= :` number of records in the building data matrix
`M= :` number of individuals (trees) in a population (population size, which is constant for each run)
`G= :` number of generations (upper bound)

The building data matrix should be formatted so that the number of rows is equal to the number of building points and a number of columns equal to the number of variables plus one. Each row therefore contains the coordinates of each building point and the corresponding output.

Further details about all other hyperparameters are provided in my PhD thesis, Appendix B.

The secondary input file has instead the following structure:

```
# test data set: 8 inputs, 1 output
# no of variables    no of test cases
8 2000
#
# test data matrix
#      Z1     Z2     Z3     Z4     Z5     Z6     Z7     Z8     TARGET
0.12 0.23 0.34 0.45 0.56 0.67 0.78 0.89 101101
```

… (second record or test point)
…

2 - Launching a HyGP experiment

A HyGP experiment is made of a given number if independent evolutions or *runs*.
To launch an experiment using the parallel implementation (OpenMP), type:

```
>> ./gp primary_input_file_location secondary_input_file_location output_folder_name
number_of_runs p
```

For example:

```
>> ./gp input/Rosenbrock/Rosenbrock_high_prec.txt
input/Rosenbrock/Rosenbrock_high_prec_TEST.txt output/test/Rosenbrock_3 3
```

Instead for the sequential version type:

```
>> ./experiment_new primary_input_file_location secondary_input_file_location
output_folder_name number_of_runs
```

For example:

```
>> ./experiment_new input/Rosenbrock/Rosenbrock_high_prec.txt
input/Rosenbrock/Rosenbrock_high_prec_TEST.txt output/test/Rosenbrock_3 3
```

3 – Harvesting the results

HyGP copies in the experiment folder two metamodels, the one performing best (minimum RMS error) on the building data set and the one performing best on the test data set. These metamodels can be found in the text files:

Parallel version:
indications provided in file *best_run_TEST.txt*

Sequential version:
best_tree.txt -> best metamodel on the building data set
best_tree_TEST.txt -> best metamodel on the test data set

These files are to be found in the main experiment directory (`output_folder_name`).