How to Run the Continued Fraction Regression Program

Source Code

The source code of the program is written using C++. The program requires C++ 11 and make to compile and execute. You can access the source code from GitHub repository at: https://github.com/MohammadNHaque/ContinuedFractionRegression-V2

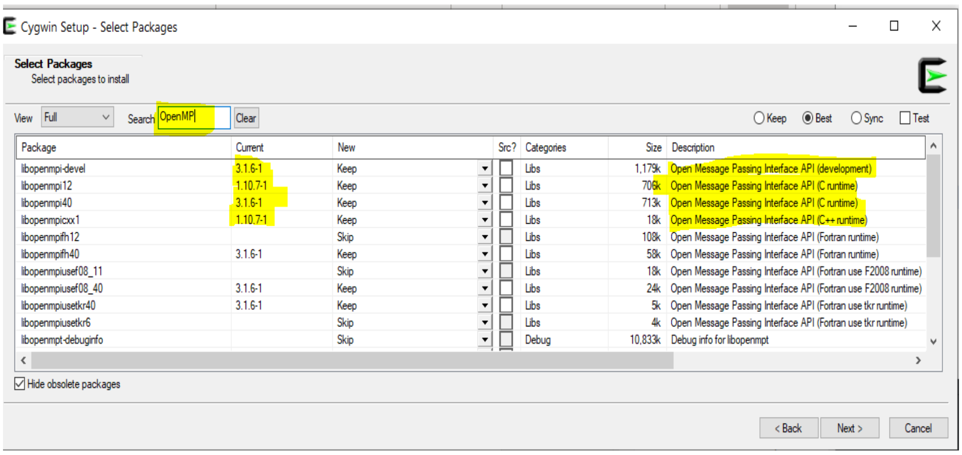
Setup Required Software and IDE

If you are using a Windows operating system, then you can use Cygwin (for C++11 and Make) and NetBeans to compile and execute the program. In Ubuntu/Unix like operating system, Netbeans, Make and C++ can be be used. Detail instruction about how to install the system in Unix OS can be found in the Readme of the project on GitHub.

Installing C++, OpenMP, gdb and Make from Cygwin for Windows

Cygwin is a Unix-like environment for Windows OS. It is a vast collection of Unix tools and Utilities. We will use GNU C++ (GCC) compiler, OpenMP, GNU Debugger (gdb) and Make form it. Download Cygwin from https://cygwin.com/install.html . Run the setup-x86_64.exe (for the 64-bit version of Windows) or setup-x86.exe for the 32-bit Windows.

- Run the Cygwin Installer.
- Select default options at the installer until the “Select Packages” option arrives.
- At the “Select Packages” Click the + next to `Devel`, to expose the developer packages/ search for following packages:
    - Scroll down until you see the `gcc–g++ C++ compiler` package and select it.
    - Scroll down until you see the `gdb GNU debugger` package and select it
    - Scroll down until you see the GNU version of the `make` package and select it
    - Scroll down until you see the `OpenMP` package and select it



Cygwin Setup - Select Packages

Select Packages
Select packages to install

View: Full   Search: OpenMP   Clear   ○ Keep ● Best ○ Sync □ Test

Package	Current	New	Src?	Categories	Size	Description
libopenmpi-devel	3.1.6-1	Keep	□	Libs	1,179k	Open Message Passing Interface API (development)
libopenmpi12	1.10.7-1	Keep	□	Libs	706k	Open Message Passing Interface API (C runtime)
libopenmpi40	3.1.6-1	Keep	□	Libs	713k	Open Message Passing Interface API (C runtime)
libopenmpicxx1	1.10.7-1	Keep	□	Libs	18k	Open Message Passing Interface API (C++ runtime)
libopenmpifh12		Skip	□	Libs	108k	Open Message Passing Interface API (Fortran runtime)
libopenmpifh40	3.1.6-1	Keep	□	Libs	58k	Open Message Passing Interface API (Fortran runtime)
libopenmpiusef08_11		Skip	□	Libs	18k	Open Message Passing Interface API (Fortran use F2008 runtime)
libopenmpiusef08_40	3.1.6-1	Keep	□	Libs	24k	Open Message Passing Interface API (Fortran use F2008 runtime)
libopenmpiusetkr40	3.1.6-1	Keep	□	Libs	5k	Open Message Passing Interface API (Fortran use tkr runtime)
libopenmpiusetkr6		Skip	□	Libs	4k	Open Message Passing Interface API (Fortran use tkr runtime)
libopenmpt-debuginfo		Skip	□	Debug	10,833k	Debug info for libopenmpt

☑ Hide obsolete packages

- We have selected our required packages, so click the `Next` button. The installer will then begin installing the packages.
- Congratulations! You've just installed `g++, OpenMP, gdb`, and `make`!

Check the versions of your Cygwin compilers and tools:

1. After completing the installation, we have to check the version of Cygwin in the Windows command Prompt:

```
C:\Users\moham>cygcheck -c cygwin
Cygwin Package Information
Package                  Version         Status
cygwin                   3.1.4-1         OK
```

2. Now check the versions of the `g++`, `gdb` and `make` in the command prompt:

```
C:\Users\moham>g++ --version
g++ (tdm64-1) 9.2.0

C:\Users\moham>gdb --version
GNU gdb (GDB) 8.3.1

C:\Users\moham>make --version
GNU Make 3.82.90
```

If you have the correct versions, then no further setup is necessary.

Download and Install Netbeans IDE

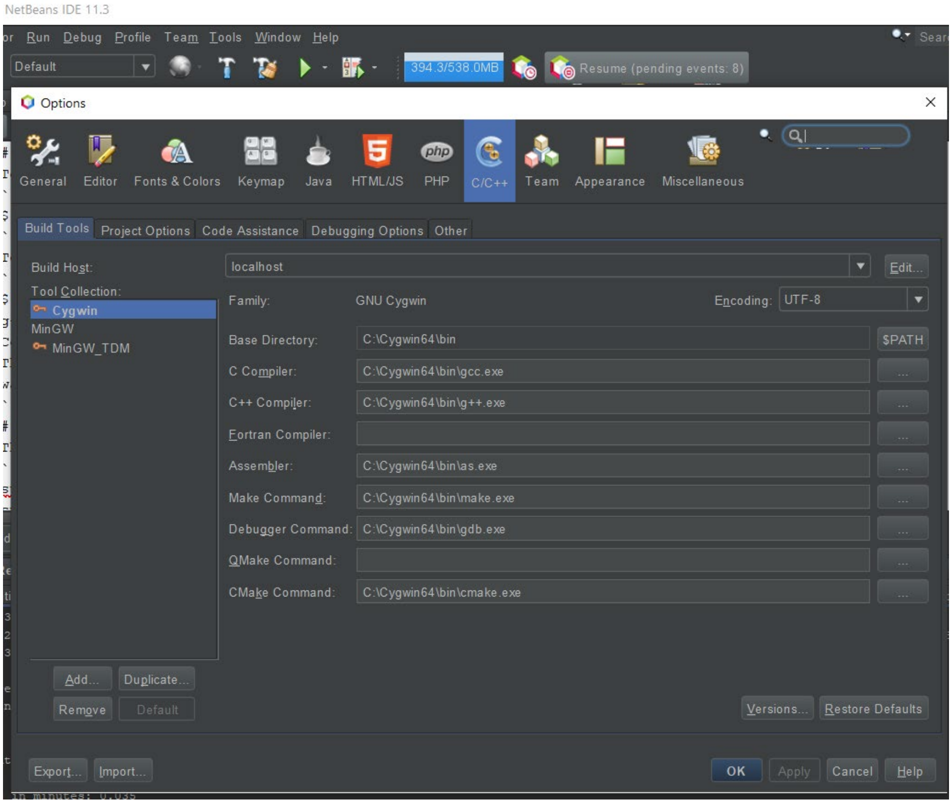
Apache NetBeans is a cross-platform Integrated Development Environment (IDE) for C++, Java, PHP and many other languages. We will use NetBeans as our IDE for the Progam. Download the Netbeans from https://netbeans.apache.org/download/index.html . I suggest using the **Apache NetBeans 12 LTS** version. After you install the Netbeans, we have to configure it for using C++.

Now go to Tools → Plugins → Settings and add "NetBeans 8.2 Plugin Portal" with url: http://updates.netbeans.org/netbeans/updates/8.2/uc/final/distribution/catalog.xml.gz

Form the available plugins install the one for C/C++.

Configure C++ Build tools in NetBeans

Now go to NetBean's Tools→ Options window and select the C/C++ and Create "Build Tool" with the proper path in your computer, as shown in Figure 1.



NetBeans IDE 11.3
Run Debug Profile Team Tools Window Help
Default
394.3/538.0MB
Resume (pending events: 8)
Options
General Editor Fonts & Colors Keymap Java HTML/JS PHP C/C++ Team Appearance Miscellaneous
Build Tools Project Options Code Assistance Debugging Options Other
Edit...
$PATH
Add... Duplicate...
Remove Default
Versions... Restore Defaults
Export... Import...
OK Apply Cancel Help

Build Host: localhost

Tool Collection:
- Cygwin
- MinGW
- MinGW_TDM

Family: GNU Cygwin

Encoding: UTF-8

Base Directory: C:\Cygwin64\bin

C Compiler: C:\Cygwin64\bin\gcc.exe

C++ Compiler: C:\Cygwin64\bin\g++.exe

Fortran Compiler:

Assembler: C:\Cygwin64\bin\as.exe

Make Command: C:\Cygwin64\bin\make.exe

Debugger Command: C:\Cygwin64\bin\gdb.exe

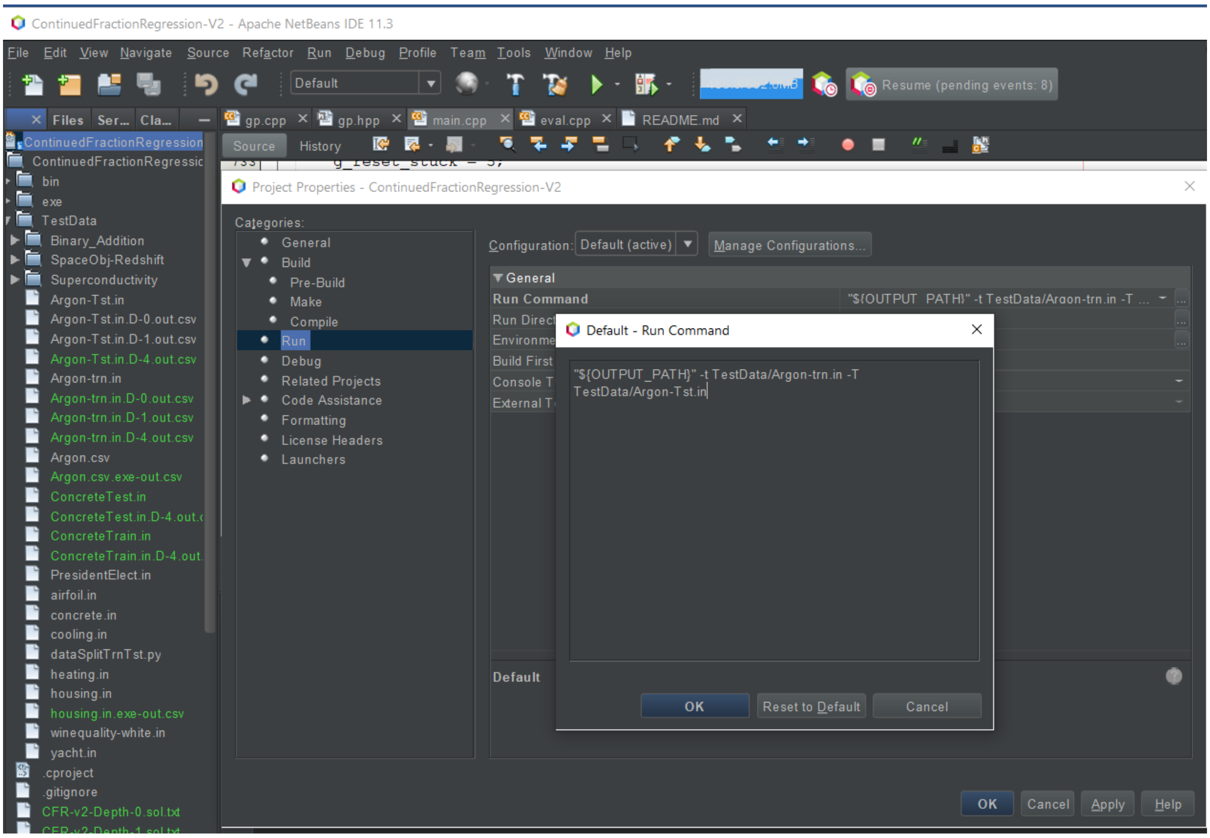
QMake Command:

CMake Command: C:\Cygwin64\bin\cmake.exe

Figure 1: Build Tool for C++ Projects in NetBeans 12

Run The Program

This section will illustrate how to run the program. The parameters of the program can be specified in the Project’s Run Property, as shown in Figure 2.



```
"${OUTPUT_PATH}" -t TestData/Argon-trn.in -T TestData/Argon-Tst.in
```

Figure 2: Setting the Command Line Parameter of the program

Run Example 1:

An Example of how to run the program for Argon data:

```
"${OUTPUT_PATH}" -t TestData/Argon-trn.in -T TestData/Argon-Tst.in
```

Run in NetBeans

```
./bin/main -t TestData/Argon-trn.in -T TestData/Argon-Tst.in
```

Unix Environment

The output shows the values of all parameters in the program

```
======================== PARAMETER VALUE =====================
-t --train           TestData/Argon-trn.in
-T --Test            TestData/Argon-Tst.in
-o --obj             mse
-d --delta           0.35
-g --num-gen         200
-m --mut-rate        0.2
-f --frac-depth      4
-r --reset-root      5
-p --pop-init        Random
-l --ls-gen          1
-nm --nelder-mead    3:250:5
-n --norm            0
-S --is-serial       true
-s --seed            random_device
-ld --ls-data        100
-sw --samp-weight    Not Present
-loo --loo-cv        false
-prec --out-prec     6
```

```
-v --verbose        0
============================================================
```

Then it will start showing the best fitness obtained (so far) for each of the generations. Finally, it shows the best fitness achieved for the solution. Then it shows the solution in LaTeX, Mathematica (in verbose mode >=2) and Excel format. The error metric for train and test are then displayed. The error metric used in the fitness function is highlighted with an * mark.

```
==================================================
BestFitness       6.86101        Solution_Size    1

${10.9815+76.5324v0}+\cfrac{{-1-32.1863v0}}{{0.990555-0.0892841v0}+\cfrac{{-
2.52729+0.775669v0}}{{-0.0326047+1.38879v0}+\cfrac{{-
0.588104+13.702v0}}{{3.62593+2.91453v0}+\cfrac{{2.52588+70.2236v0}}{{-
0.102985-3.10377v0}}}}}$
(10.9815+76.5324*B2)+(-1-32.1863*B2)/((0.990555-0.089284
2.52729+0.775669*B2)/((-0.0326047+1.38879*B2)+(-
0.588104+13.702*B2)/((3.62593+2.91453*B2)+(2.52588+70.2236*B2)/((-0.102985-
3.10377*B2)))))

Score    *mse    nmse    pearson theil   med.err med.ae  mase
Train    5.08223 0.000757177     0.999609        0.994379        2.
2.15799 0.11278
Test     1.15378e+07     1.81891 -0.969125       -0.0286502      2030.19
2030.19 1.07619

Solution is written at: CFR-v2-Depth-4.sol.txt
```

Solution in LaTeX format

Solution in Excel format

Error Metric

Run Example 2 : with some Parameters

Another Example: running the program with parameters used in CEC 2020 paper: [obj=mse, gen=200, muRate=0.10 penalty=0.1 nelder-mead:4:250:10 depth=4]:

```
"${OUTPUT_PATH}" TestData/Argon-trn.in -T TestData/Argon-Tst.in -g 200 -m
0.1 -d 0.1 -nm 4:250:10 -f 4 -v 2
```

Run in NetBeans

```
./bin/main TestData/Argon-trn.in -T TestData/Argon-Tst.in -g 200 -m 0.1 -d
0.1 -nm 4:250:10 -f 4 -v 2
```

Unix Environment

More Detail about the program, data format and some data for testing it can be found in the Readme file at the Github Page:
https://github.com/MohammadNHaque/ContinuedFractionRegression-V2/