User Manual: High Performance Control C++ Library

Written by David Dos Santos

This manual explains the installing and use of the High-Performance Control C++ library.

Currently the library only contains an implementation of the Markov Approximation Method and uses Euler's method for Differential equation solving. It also has the Continuous Decomposition Method and can utilise it (http://www.alexgorodetsky.com/index.html).

Installation

The following installations have been tested only on specific operating systems.

All platforms need the following packages:

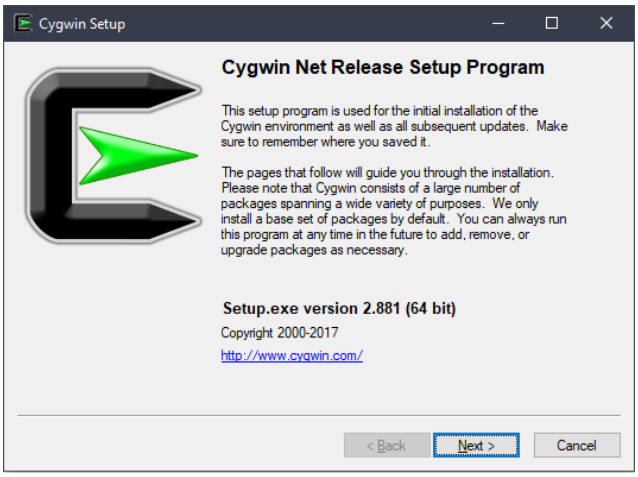
- Git (https://git-scm.com/downloads)
- CMake (https://git-scm.com/downloads)
- BLAS (http://www.netlib.org/blas/)
- LAPACK (http://www.netlib.org/lapack/)

If using Windows, install the Git package only as the other 3 will be covered in the following section.

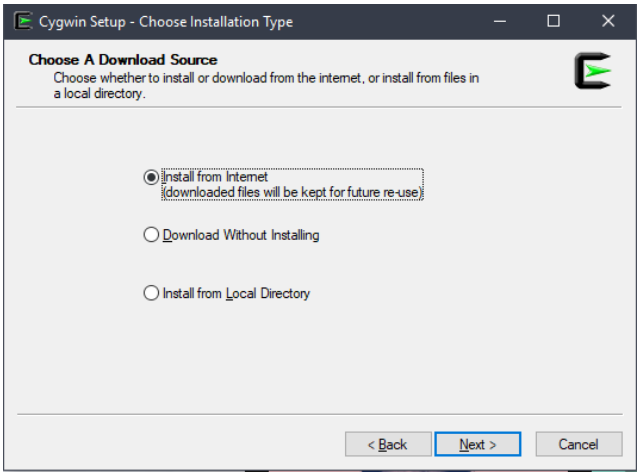
For Windows 10:

To install the necessary files, you will need Cygwin (https://www.cygwin.com/install.html)

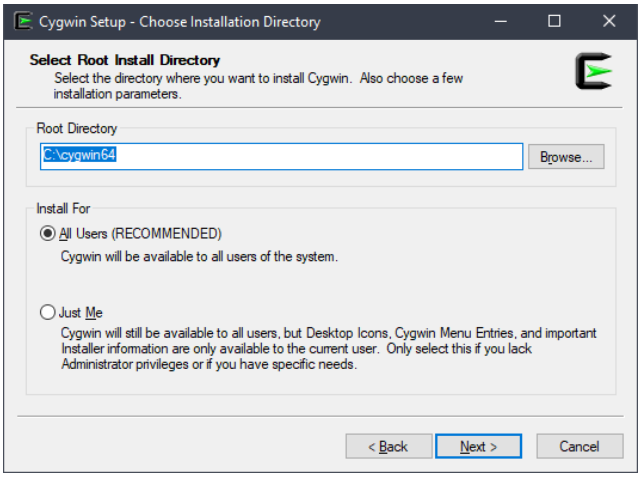
- When you open the setup you will have something like this. Click Next.



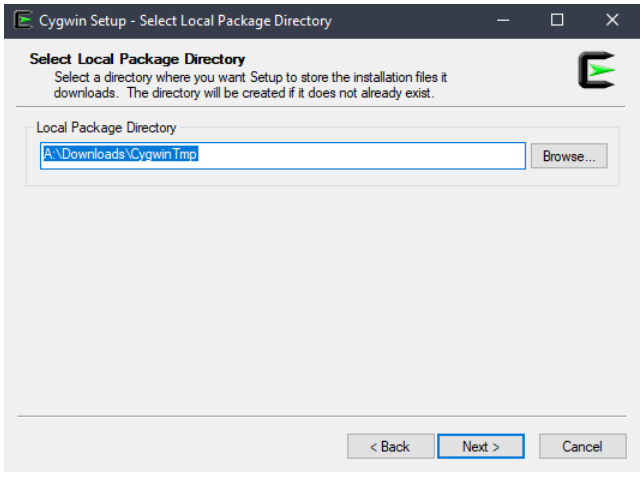
- In the Cygwin installation, choose the download source 'Install from Internet'. Click Next.



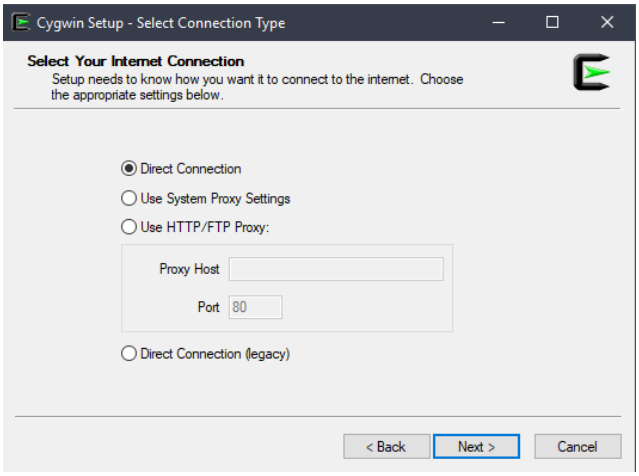
- Select the root directory where you want to install it (and remember this as you may have to add it to your PATH variable later). Click Next.



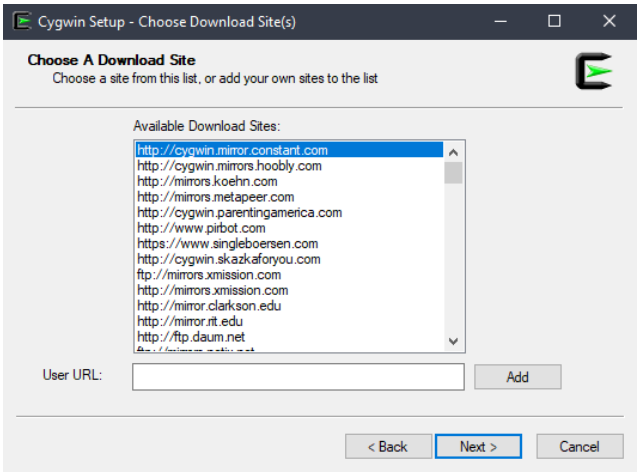
- Select a location to place the temporary downloads (the temporary downloads can be deleted after installation is complete, best to pick somewhere where you will remember to clear it later)



- If you have a proxy to use you can enter those details now, else just leave it as Direct Connection.

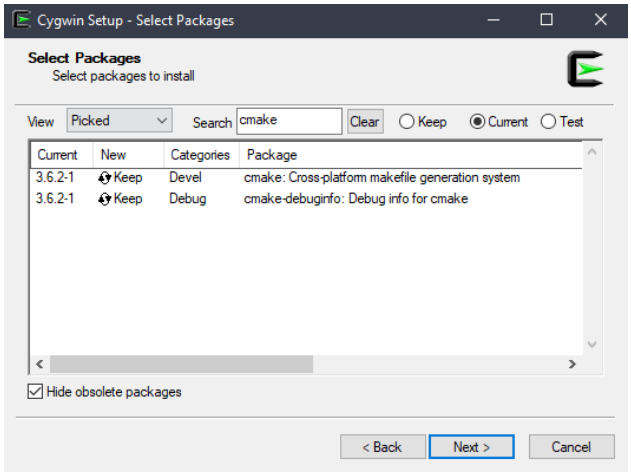


- There are a bunch of different mirrors (or download locations) to choose from, but the only real difference between them is speed and security (due to protocols being used). The most secure are obviously the https, and least secure are ftp. The first one on the list is probably the most reliable.

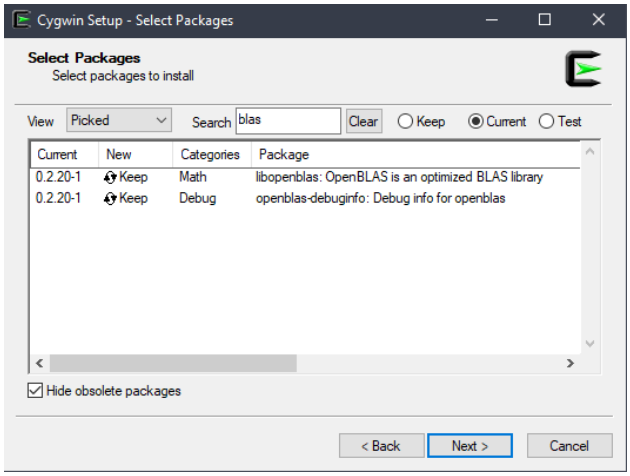


- At the next section, you will be selecting different packages, but don't click the Next button until all packages have been selected. If you forget a package just repeat all processes above up to this point (because this is how you install new packages to Cygwin anyways)

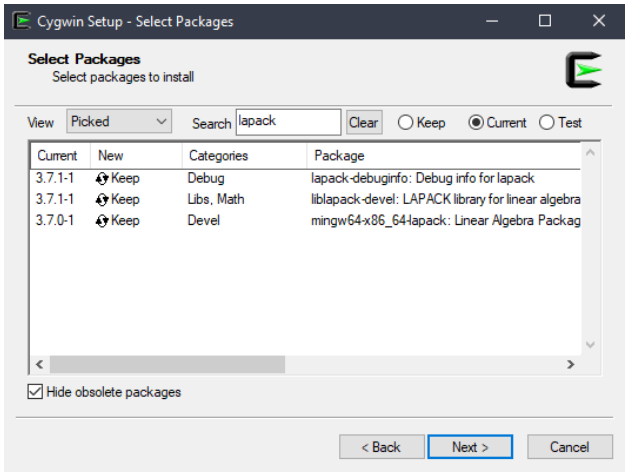
- You need cmake (the cmake-debuginfo is pretty handy too but not necessary). The version you get of cmake may be different, just get the latest version as cmake is backwards compatible. Also note that mine all say “Keep” under the New tab since I already had these packages installed at the time of making this user manual.



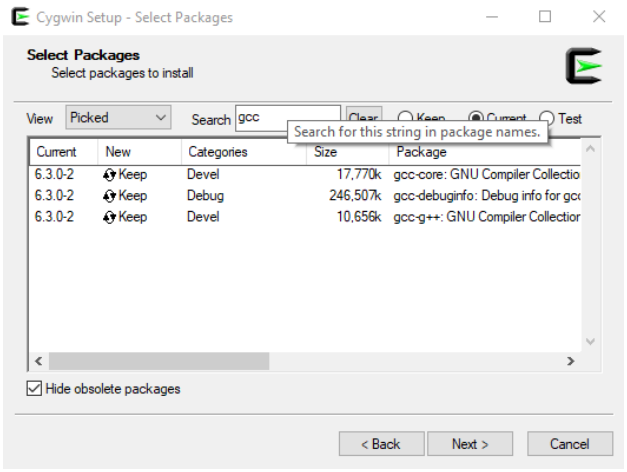
- You need the libopenblas package



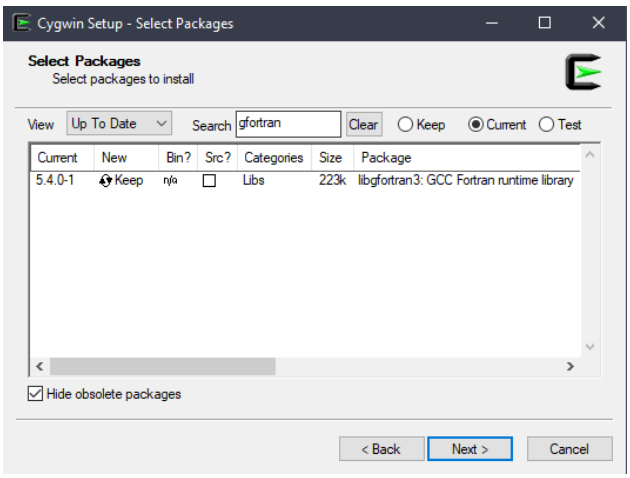
- You need the liblapack package. You don’t need the mingw version unless you are using MinGW (which is not covered by this user manual)



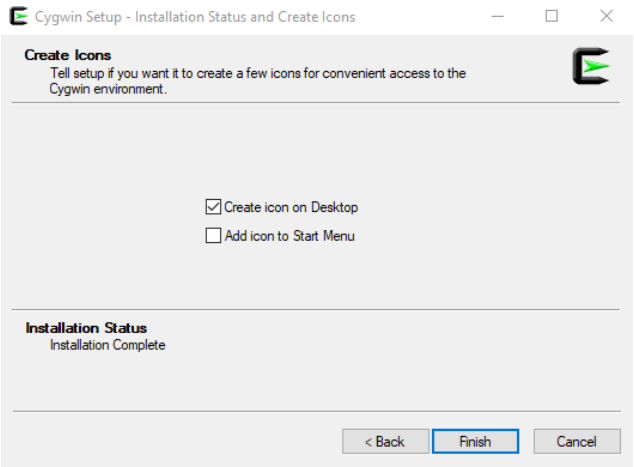
- You need the gcc, gcc-g++ and gcc-debuginfo packages



- The final package will be the libgfortran package, as the blas and lapack are originally Fortran based



- Now click Next and it will have another screen summarising the components to be installed (including their prerequisite parts), click Next.



- Assuming you didn't have any firewall or permission issues, you should finish with this

Now you are ready to copy the Git repository and begin installing the High-Performance-Control library. Run the following commands in a terminal (e.g. Powershell, Command-prompt, or your shiny new Cygwin terminal):

```
git clone https://github.com/revolutionized/High-Performance-Control.git High-Performance-Control

cd High-Performance-Control
```

Once this completes, you can type in a Command-prompt or PowerShell terminal (But not the Cygwin terminal):

```
Install_Windows.bat
```

Or double click the Install_Windows.bat file. Be prepared as this will take several minutes to install (depending on your download speed and computational power).

The Install_Windows.bat just calls the bash/shell script executable – Install_Linux.sh. This will download all the packages provided by Alex Gorodetsky (https://github.com/goroda), build and install them in the correct locations, and then build the High-Performance-Control code.

Ubuntu 16.04

To install CMake, in a terminal type:

```
sudo apt install cmake
```

To install BLAS, in a terminal type:

```
sudo apt install libblas-dec
```

To install LAPACK, in a terminal type:

```
sudo apt install liblapack
```

You will also need the GFortran package, type:

```
sudo apt install gfortran
```

Finally, you can copy the clone repository:

```
git clone https://github.com/revolutionized/High-Performance-Control.git High-Performance-Control

cd High-Performance-Control
```

And run the installation script (this will take several minutes):

```
sudo /.Install_Linux.sh
```

MacOS

To Install CMake, you can download it from their website, or using Homebrew (which allows for installations similar to Linux). I personally found that my installed CMake wasn't recognised on the command line until I used Homebrew.

First install XCode Command line tools (requires you to have XCode which you can get from the App Store).

```
xcode-select --install
```

Then you can install Homebrew with the following command:

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Now you can install CMake using:

```
brew install cmake
```

If typing cmake from this point on doesn't work then make sure your PATH variable is set:

```
PATH="/usr/local/bin:$PATH"
```

Reset the shell session and try again.

Depending on your version of MacOS you may need to install LAPACK and BLAS from the www.netlib.com website (exact links to each library given at the beginning of this Installation section). Also note that the latest version of LAPACK contains BLAS in its download packet.

More modern versions of MacOS have LAPACK and BLAS contained in the Accelerate-Framework.

Finally, you can copy the clone repository:

```
git clone https://github.com/revolutionized/High-Performance-Control.git High-Performance-Control

cd High-Performance-Control
```

And run the installation script (this will take several minutes):

```
sudo /.Install_Linux.sh
```

Creating your own examples or editing the code

You have the option of editing the code and contributing to the Github repository by forking (or branching) and creating a pull-request when you are ready to merge with the master branch (if these terms don't make sense, I recommend you take an introduction to source control or Git tutorial). Alternatively, you can create your own repository.

The code base has been set up so that you place examples in the examples folder, under their own directory. Ideally you need only have a single header and single cpp file (see the examples currently in there). You then add the execution of that to the main.cpp, and then have access to it using command line arguments.

Collecting and Viewing Results

The results from the examples are stored in '.dat' files which can be opened and viewed using Gnuplot[1].

An example of how to use Gnuplot is given in hpc/Project_C++_Files/viewplots.gla:

```
#!/bin/bash

gnuplot -p << EOF

# Open a new window to plot the figure (similar to MATLABs figure(0))

set term x11 0

# Here we plot the results of the ODE

set xlabel "Time (seconds)"

set ylabel "Velocity"

plot 'build/ExactEulerResult.dat' using 1:2 with linespoint

# Replot is like MATLABs hold on

replot 'cmake-build-debug/MarkovEulerResult.dat' using 1:2 with linespoint

# Open a new window to plot the figure (similar to MATLABs figure(1))

set term x11 1

# Here we plot the optimal control values for the ODE

set xlabel "Time (seconds)"

set ylabel "Control value"

plot 'build/ExactControl.dat' using 1:2

replot 'build/MarkovControl.dat' using 1:2

EOF
```

[1] http://www.gnuplot.info/download.html