XtalOpt Installation Guide

Release 12.0

Generated by Doxygen 1.8.13

Contents

1 Installation for Linux

Pre-compiled binaries of XtalOpt may be obtained from here. However, the Linux binaries may not work on every Linux computer. You may continue reading if you wish to compile XtalOpt on your own.

1.1 Dependencies

First and foremost, a C++ compiler with C++11 is required to compile XtalOpt. GCC >= 4.8.4 and Clang >= 3.5 will both work.

The following dependencies can be installed from the package manager of most linux distributions:

- `git`, any version
- `cmake` (and optionally the ncurses GUI, ccmake) >= 3.0
- `Qt` >= 5.2.1
- `Qwt` >= 6.1.3 (compiled with Qt5)
- `Eigen` >= 3

If you wish to enable SSH (default), you will also need:

- `openssh`, any version
- `libssh`, any version

Using the package manager for your linux distribution, install the dependencies listed above. For example, on Arch Linux, one can use the command

```
pacman -S git cmake glu qt5-base qwt eigen libssh openssh
```

On Ubuntu 16.04, one can use the command

```
sudo apt-get install git cmake qt5-default libqwt-qt5-dev libeigen3-dev libssh-dev
```

Note: if you are using Ubuntu 14.04, libqwt-qt5-dev may not be available. Installing Qwt that depends on Qt4 will not work. Therefore, you can either use the .deb file I have created here, or you can compile it yourself.

1.2 XtalOpt

Once the dependencies are installed, installing XtalOpt is simple. First, change to your source directory:

```
cd $HOME/src
```

Check out the master branch of the XtalOpt sources:

```
git clone git://github.com/xtalopt/XtalOpt.git xtalopt
```

Make a build directory and enter it:

```
mkdir xtalopt/build
cd xtalopt/build
```

Configure, build, and install (set the CMAKE_INSTALL_PREFIX to wherever you'd like):

```
cmake .. -DCMAKE_INSTALL_PREFIX=<wherever>
make -j3
make install
```

Note: if you need to use Kerberos authentication to establish a SSH connection to the remote server, the `libssh` library used by XtalOpt will not work. There is a workaround for Linux (and possibly Mac) users, which will call the command line `ssh`/`scp` commands to communicate with the remote cluster. This can be enabled by adding `-DUSE_CLI_SSH=true` to the cmake command above. This `ssh` backend will not prompt for a password, and expects the `ssh` commands to "just work". Passwordless `ssh` logins can be enabled using the `ssh-copy-id` command.

XtalOpt is now installed.

1.3 Test the installation

Try running the resulting XtalOpt executable (either in the bin directory of your install directory or in the build directory). If a window opens and you are able to select options, then XtalOpt was successfully installed.

2 Installation for Mac OS X

We encourage users to download and use XtalOpt pre-compiled from `here`. However, you may continue reading if you wish to compile XtalOpt on your own.

2.1 Dependencies

First, and foremost, Mac OS X users must install Xcode to begin compiling the following programs. Xcode 8 or later is required because Apple did not start including some of the C++11 features that XtalOpt uses until then.

The following dependencies can be installed from the package manager MacPorts or Homebrew:

- git, any version
- cmake (and optionally the ncurses GUI, ccmake) >= 3.0
- Qt >= 5.2.1
- Qwt >= 6.1.3 (compiled with Qt5)
- Eigen >= 3

If you wish to enable SSH (default), you will also need:

- openssh, any version
- libssh, any version

Using the package manager for your mac distribution, install the dependencies listed above. For example, using MacPorts,

```
port install git cmake qt5-mac qwt61 eigen libssh
```

or using Homebrew,

```
brew install git cmake qt5 qwt eigen libssh
```

2.2 XtalOpt

```
cd $HOME/src
```

Check out the master branch of the XtalOpt sources:

```
git clone git://github.com/dlonie/XtalOpt.git xtalopt
```

Make a build directory and enter it:

```
mkdir xtalopt/build
cd xtalopt/build
```

For Homebrew and at the time of writing this, Qt5 and Qwt headers are not included in the $PATH, and cmake will not find them. So we must show cmake where they are. Thus, we need to add the next few lines of code before running cmake. Set the CMAKE_INSTALL_PREFIX to wherever you'd like:

```
export CMAKE_FLAGS="$CMAKE_FLAGS -DCMAKE_PREFIX_PATH=/usr/local/opt/qt/"
      export CMAKE_FLAGS="$CMAKE_FLAGS -DQWT_INCLUDE_DIR=/usr/local/opt/qwt/lib/qwt.framework/Headers"
      export CMAKE_FLAGS="$CMAKE_FLAGS -DCMAKE_INSTALL_PREFIX=<wherever>"
```

Configure, build, and install:

```
cmake ${CMAKE_FLAGS} ..
make -j3
sudo make install
```

Note: if you need to use Kerberos authentication to establish a SSH connection to the remote server, the libssh library used by XtalOpt will not work. There is a workaround for Linux (and possibly Mac) users, which will call the command line ssh/scp commands to communicate with the remote cluster. This can be enabled by adding -DUSE_CLI_SSH=true to the cmake command above. This ssh backend will not prompt for a password, and expects the ssh commands to "just work". Passwordless ssh logins can be enabled using the ssh-copy-id command.

XtalOpt is now installed.

2.3 Test the installation

An xtalopt.app directory will be created wherever your cmake install prefix pointed to. You can test the program by opening this .app file. If it opens successfully and the dialog box appears, you have successfully installed XtalOpt.

3 Installation for Windows

We encourage users to download and use XtalOpt pre-compiled from here. In fact, this guide does not provide a complete set of instructions for compiling it on Windows. However, you may continue reading if you wish to attempt compile XtalOpt on your own.

3.1 Dependencies

First and foremost, a C++ compiler with C++11 is required to compile XtalOpt. MSVC >=2015 will work. One important point to remember when compiling for Windows is that you must use the same compiler for every dependency. So if MSVC 2017 64-bit Release is to be used for XtalOpt, all of the dependencies must be compiled with MSVC 2017 64-bit Release.

Similar to the compilations for the other operating systems, the following dependencies are required:

- `git`, any version
- `cmake` (and optionally the ncurses GUI, ccmake) >= 3.0
- `Qt` >= 5.2.1
- `Qwt` >= 6.1.3 (compiled with Qt5)
- `Eigen` >= 3

If you wish to enable SSH (default), you will also need:

- `openssh`, any version
- `libssh`, any version

For git and cmake, only the executables are used, so you may download them from their respective websites.

For Qt, you may either download pre-compiled binaries for your specific compiler or compile it yourself. Qt typically has compiled versions for just about every compiler, so if you choose to download it, be sure to pick the compiled version that matches your compiler.

Eigen is a header-only library, so it does not need to be compiled separately.

OpenSSH, LibSSH, and Qwt need to all be compiled with your compiler of choice. If you wish to use MSVC 2015 (32-bit) or MSVC 2017 (64-bit), pre-compiled binaries here. Note that LibSSH has OpenSSH statically compiled within it in these binaries.

You may also compile OpenSSH, LibSSH, and Qwt on your own, but we do not provide instructions for that. You must use their instructions to do so.

3.2 XtalOpt

Once you have compiled all of the binaries, you need to run cmake with the locations of all of the dependencies defined. An example of this for MSVC can be found in the batch file in `scripts/cmake-msvc.bat` file in the XtalOpt repository. Afterwards, you must run "make" (or "nmake" in the case of MSVC) in the build directory. If the cmake option "-DBUILD_INDEPENDENT_PACKAGE" was used, a "make install" (or "nmake install" in the case of MSVC) will install XtalOpt and all of its dependencies into your cmake install directory.

3.3 Test the installation

If the compilation was successful, the installed xtalopt.exe executable may be ran. It will only run successfully if all of the .dll dependencies are found, though, so you may need to manually find and place all of the .dll dependencies into the directory with XtalOpt.