2 Installation and Getting Started

2.1 Prerequisites

Smooth Emulator software should run on UNIX, Mac OS or Linux, but is not supported for Windows OS. *Smooth Emulator* is largely written in C++. In addition to a C++ compiler, the user needs the following software installed.

- git
- CMake
- Eigen3 (Linear Algebra Package)
- GSL (Gnu Scientific Library)
- Python/Matplotlib (only for generating plots in the MCMC procedure)

CMake is an open-source, cross-platform build system that helps automate the process of compiling and linking for software projects. Hopefully, CMake will perform the needed gymnastics to find the Eigen3 and GSL installations. To install CMake, either visit the CMake website (https://cmake.org/), or use the system's package manager for the specific system. For example, on Mac OS, if one uses *homebrew* as a package manager, the command is

```
% brew install cmake
```

Eigen is a C++ template library for vector and matrix math, i.e. linear algebra. The user can visit the Eigen website (https://eigen.tuxfamily.org/dox/), or use their system's package manager. For example on Mac OS with *homeebrew*,

```
% brew install eigen
```

The GNU Scientific Library (GSL) is a numerical library for C and C++ programmers. The library provides a wide range of mathematical routines such as random number generators, special functions and least-squares fitting. There are over 1000 functions in total with an extensive test suite. One can either download the software from the GSL website (https://www.gnu.org/software/gsl/), or use a package manager. Again, for *homebrew* on Mac OS,

```
% brew install gsl
```

2.2 Making Home Directory and Setting Home Environment Variable

The software requires two repositories. They should be cloned into the same directory. For compilation purposes this directory needs to be accessed via an environmental variable. For example the directory might be named `/Users/CarlosSmith/git_msu`. The User needs to set an environmental variable, `GITHOME_BAND`, to the full path of the directory, e.g.

```
% export GITHOME_BAND="/Users/CarlosSmith/bandframework/software"
```

It is recommended to copy this command into the user's `.bashrc` (or equivalent) file to avoid re-defining it each time one needs to recompile.

2.3 Downloading

From within the `GITHOME_BAND/` directory,

```
GITHOME_BAND% git clone https://github.com/bandframework/bandframework.git
```

This creates 2 directories: `.../GITHOME_BAND/SmoothUtilities` and `.../GITHOME_BAND/SmoothEmulator`.

The User needs to create a project directory from which the User would perform most projects. This is easiest accomplished by copying a template from the *Smooth* distribution,

```
% cp -r GITHOME_BAND/templates/myproject MY_PROJECT
```

Hence forth, `MY_PROJECT` will refer to the directory, including the path, from which the User will perform most of the analysis. The User may wish to have several such directories. These directories should be outside the main distribution, i.e. outside the `bandframework/software/SmoothEmulator/` or `bandframework/software/SmoothEmulator/` paths.

Although the main source code, include files and libraries are all located in the software directory, the main programs and executables are not. The motivation for this decision is to allow the User to easily modify their own versions of the main programs. These tend to be very short programs. For that reason their is a separate directory to store the main programs and their executables. The User can easily set this up by copying a template directory,

```
% cp -r GITHOME_BAND/templates/mylocal MY_LOCAL
```

Here `MY_LOCAL` will hence forth refer to the path of this directory. This directory should be outside the main distribution, i.e. outside the `SmoothEmulator/` or `bandframework/software/SmoothEmulator/` paths.

For the remainder of this manual, `GITHOME_BAND`, `MY_LOCAL` and `MY_PROJECT` will be used to denote the location of these directories.

2.4 Directory and File Structure

Once compiled, the libraries in the `commonutils/` directory are used for a variety of tasks. These libaries are not particularly designed for *Smooth Emulator* or *Simplex Sampler*. The `SmoothEmulator/` directory contains codes that are used to create libraries specific to the sampler and emulator. The executables are stored in `MY_LOCAL/bin`. The short main program source files are located in `MY_LOCAL/main_programs/`. It is not envisioned that the User would edit files in the `SmoothEmulator/software` directory, but that the User may well wish to create custom versions of the short main programs in `MY_LOCAL/main_programs/`. The main programs are compiled using the CMake files in `MY_LOCAL/build/`. The User may find it convenient to add `MY_LOCAL/bin/` to their path.

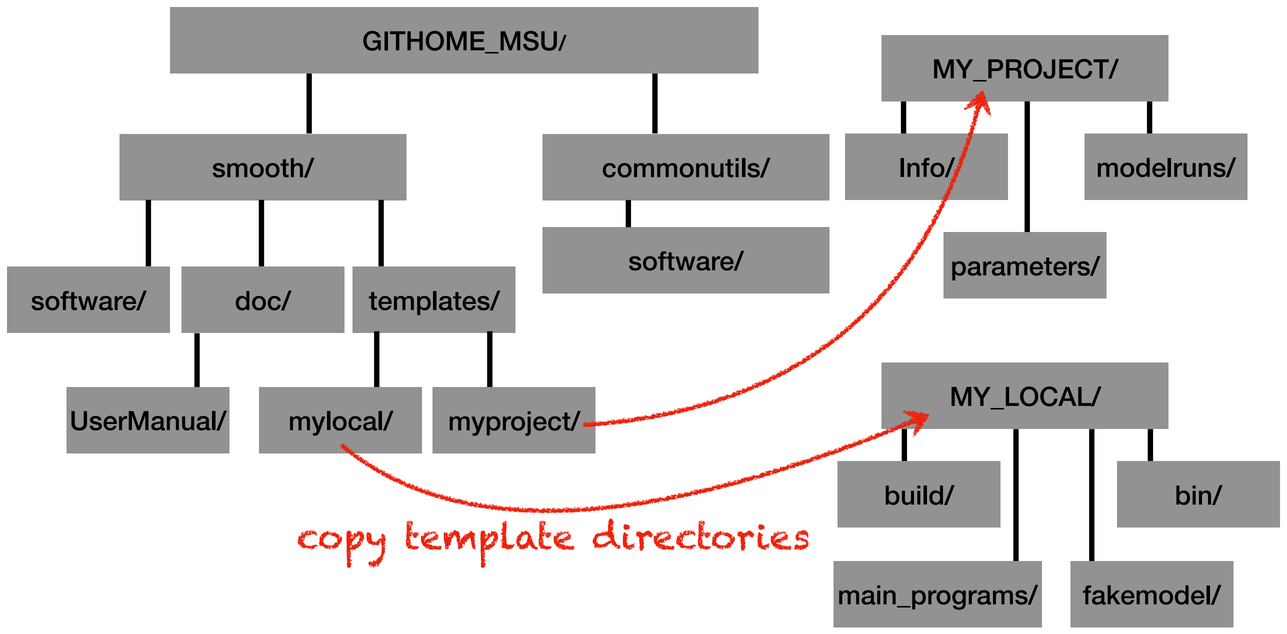

Figure 2.1: The directory structure: The User clones two repositories into some location, which will be referred to as `GITHOME_BAND`. The User can then copy two template directories into the User's choices of locations outside the path of the repositories. The name of those two directories will be referred to as `MY_LOCAL`, which will contain the main programs and executables, and `MY_PROJECT` which contains the data files related to a given project. The programs are designed to be run from within the `MY_PROJECT/` directory.

2.5 Compiling Libraries

First, change into software directories, then create the makefiles with cmake, then compile them.

```
% cd GITHOME_BAND/SmoothUtilities/software
GITHOME_BAND/SmoothUtilities/software% cmake .
GITHOME_BAND/SmoothUtilities/software% make
GITHOME_BAND/SmoothUtilities/software% cd ../GITHOME_BAND/SmoothEmulator/software
GITHOME_BAND/SmoothEmulator/software% cmake .
GITHOME_BAND/SmoothEmulator/software% make
```

At this point all the libraries are built, but this does not include the main programs. The main programs are short, and are located in a separate location, as they are meant to serve as examples which the User might copy and edit at will.

Finally, compile the main programs. Below, this illustrates how to build the programs used for generating training points with Simplex and for tuning the emulator with Smoothy:

```
% cd MY_LOCAL/build
MY_LOCAL/build% cmake .
MY_LOCAL/build% make simplex
MY_LOCAL/build% make smoothy_tune
.
.
```

Other source codes for main programs can be found in `MY_LOCAL/main_programs/`. If you build your own main programs (probably using these as examples), you can edit the `CMakeList.txt` file in `GITHOME_BAND/SmoothEmulator/local/build`, using the existing entries as an example. The executables should appear in `MY_LOCAL/bin/`.

2.6 The Project Directory

Within `MY_PROJECT/` there are three sub-directories (assuming it was created from the template). The first is `MY_PROJECT/Info/`. Information about the model parameters, and their priors is stored in `MY_PROJECT/Info/modelpar_info.txt`, and information about the observables is stored in `MY_PROJECT/observable_info.txt`. The `MY_PROJECT/parameters` directory stores user-defined parameter files used by *Simplex Sampler*, `MY_PROJECT/parameters/simplex_parameters.txt`, and by *Smooth Emulator* `MY_PROJECT/parameters/emulator_parameters.txt`. The `MY_PROJECT/modelruns` directory will store information for each full-model run. The directories `MY_PROJECT/modelruns/run0/`, `MY_PROJECT/modelruns/run1/`, $\cdots$, have files describing the model parameters for each run, along with the output required by the emulator for each specific full-model run. For example, the `MY_PROJECT/modelruns/run1/` directory has the files `mod_parameters.txt` and `obs.txt`. The first file stores the model parameter values for that particular training run. The User then runs their full model based on those parameters and stores the corresponding observables in `obs.txt`. The User may generate the `mod_parameters.txt` files using *Simplex Sampler*, or the user might generate them according to some other prescription. Once the User has then generated the `obs.txt` files, *Smooth Emulator* can then build and tune the emulator.