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)
- 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 installation. 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 *homebrew*,

```
% brew install eigen
```

2.2 Downloading the Repository

The software requires downloading the BAND framework software repository into some directory. Should that be in the User's home directory, the User might enter

```
    /Users/CarlosSmith% git clone https://github.com/bandframework/bandframework.git
```

Within the repository, there will be a directory `/Users/CarlosSmith/bandframework/software/SmoothEmulator/`. All the relevant functionality of *Smooth Emulator* is contained within this directory. Throughout the manual the phrase `${GITHOME_SMOOTH}` will refer to this directory.

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_SMOOTH}/templates/myproject ${MY_PROJECT}
```

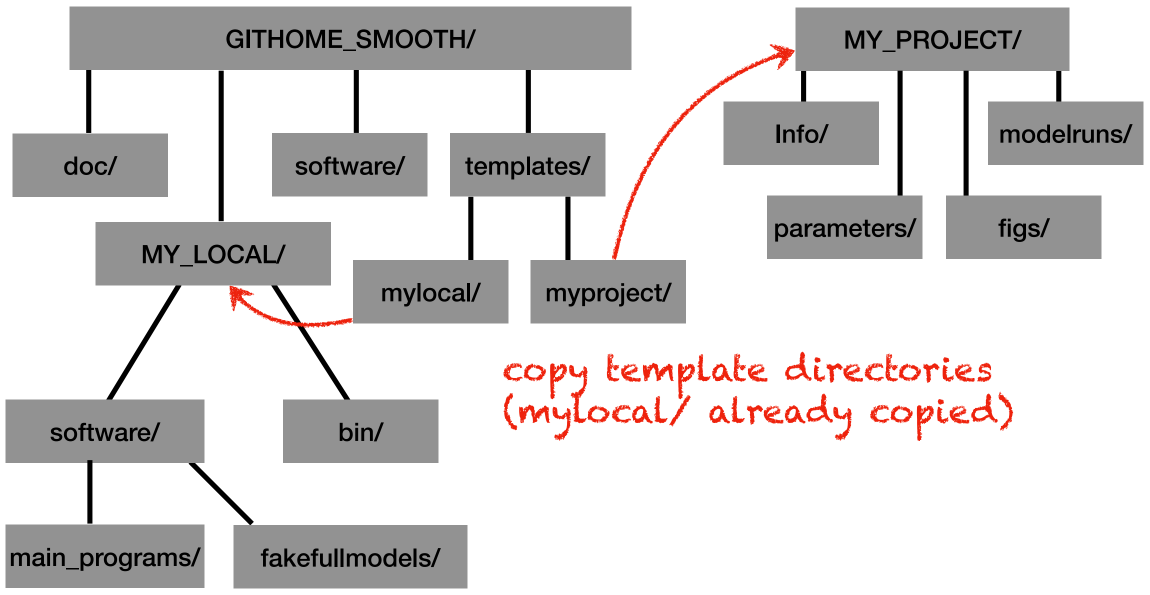

Figure 2.1: The directory structure: The User clones the repository into some location, which will be referred to as `${GITHOME_SMOOTH}`. The User can then copy the `${MY_PROJECT}` directory to the User's choices of locations outside the path of the BAND repository. The programs are designed to be run from within the `MY_PROJECT/` directory, and the User can make multiple copies of `${MY_PROJECT}` whenever they wish to run and save different instances of *Smooth Emulator*

.

Hence forth, `${MY_PROJECT}` is a name chosen by the User, and will refer to the directory, including the path, from which the User will perform most of the analysis. The User may wish to have several such directories, located according to the User's preference. Any time a new analysis is performed with new parameters, and if the User wishes to save the previous analysis, a new `${MY_PROJECT}` directory should be created. These directories should be outside the main distribution, i.e. outside the `bandframework/` path so that the User's work does not interfere with the repository.

Within the `templates/` directory, there is also a directory `../templates/mylocal/`. The `mylocal/` directory contains the main programs and the *Smooth Emulator* executables which will be stored in `${GITHOME_SMOOTH}/mylocal/bin`. Because the User may well wish to edit the main programs, or to add similar programs, this provides the User a space to make such edits, all while leaving an original copy of the directory in the `templates/` directory.

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

2.3 Directory and File Structure

The `${GITHOME_SMOOTH}/software/` directory contains codes that are used to create libraries specific to the sampler and emulator. The User can change to The executables are stored in `${MY_LOCAL}/bin`. The short main program source files are located in `${MY_LOCAL}/software/main_programs`. It is not envisioned that the User would edit files in the `SmoothEmulator/software/` directory, but that the User may well wish to create custom versions of the short main programs in `${MY_LOCAL}/software/main_programs/`. The main programs are compiled using the CMake files in `${MY_LOCAL}/software/main_programs/`. The User may find it convenient to add `${MY_LOCAL}/bin/` to their path.

2.4 Compiling Libraries

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

```
% cd ${GITHOME_SMOOTH}/software
${GITHOME_SMOOTH}/software% cmake .
${GITHOME_SMOOTH}/software% make
```

There seems to be a common problem that `cmake` misreports the path of the `Eigen` installation. If the User should get an error stating that the Eigen header files cannot be found, the User can set the environmental variable,

```
% export EIGEN3_INCLUDE_DIR=/usr/local/include/eigen3
```

The final arguments may need to be changed depending on the User's location of the packages. If the User wishes to choose a specific C++ compiler, the `cmake` command should be replaced with `cmake -D YOUR_PREFERRED_COMPILER`.

At this point all the libraries are built, but this does not include the main programs. The main programs are short, and are meant to serve as examples which the User might copy and edit at will. Before compiling, one needs to set an environmental variable so that the compilation can find the libraries needed for compilation.

```
% export GITHOME_BAND_SMOOTH=/Users/CarlosSmith/.../bandframework/software/SmoothEmulat
```

The first part of the path needs to be replaced with the location of the `bandframework` git repository. If one prefers not to set an environmental variable, one can instead edit `${MY_LOCAL}/software/CMakeLists.tx` and replace the line where the variable `GITHOME_BAND_SMOOTH` is set to the shell's environmental variable with one where it is set explicitly.

Below, this illustrates how to compile the programs used for generating training points with Simplex and for tuning the emulator with *Smooth Emulator*:

```
% cd ${MY_LOCAL}/software
${MY_LOCAL}/software% cmake .
${MY_LOCAL}/software% make
.
.
```

The `cmake` command will also recompile the main libraries in `${GITHOME_SMOOTH}/software/` if necessary. Several source codes for main programs can be found in `${MY_LOCAL}/software/main_programs/`. If you build your own main programs (probably using these as examples), you can edit the `CMakeList.txt` file in `${MY_LOCAL}/software/main_programs/`, using the existing entries as an example. The executables should appear in `${MY_LOCAL}/bin/`.

2.5 The MY_PROJECT Directory

Within `${MY_PROJECT}/` there are four sub-directories (assuming it was created from the template). The first is `${MY_PROJECT}/smooth_data/Info/`. Information about the model parameters, and their priors is stored in `${MY_PROJECT}/smooth_data/Info/modelpar_info.txt`, and information about the observables is stored in `${MY_PROJECT}/observable_info.txt`.
The `${MY_PROJECT}/smooth_data/parameters` directory stores user-defined parameter files used by *Simplex Sampler*,
`${MY_PROJECT}/smooth_data/parameters/simplex_parameters.txt`, and by *Smooth Emulator*
`${MY_PROJECT}/smooth_data/parameters/emulator_parameters.txt`.
The `${MY_PROJECT}/modelruns` directory will store information for each full-model run.

The directories `${MY_PROJECT}smooth_data/modelruns/run0/`, `.../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 `${MY_PROJECT}/smooth_data/modelruns/run1/` directory has the files `mod_parameters.txt` and `obs.txt`. The first file stores the model parameter values for that particular training run. The User then runs their full model based on those parameters and stores the corresponding observables in `obs.txt`. The User may generate the `mod_parameters.txt` files using *Simplex Sampler*, or the user might generate them according to some other prescription. Once the User has then generated the `obs.txt` files, *Smooth Emulator* can then build and tune the emulator. Finally the `${MY_PROJECT}/figs` directory contains python scripts for plotting results.