Feat_FloWer Manual

Raphael Münster

April 19, 2018

Table Of Contents

1 Installation **2**
1.1 Prerequisites . . . 2
1.2 Checking out the Code from the Repository . . . 2
1.3 Building the Code from Source . . . 3
1.3.1 Linux Systems . . . 3
1.3.2 Windows Systems . . . 4

2 Running a FeatFlower Application **5**
2.1 Introduction to the Application Framework . . . 5
2.1.1 Preprocessing and Mesh Generation . . . 5
2.1.1.1 Using Meshes from the Mesh Repository . . . 5
2.2 Appendix . . . 6

Chapter 1

Installation

1.1 Prerequisites

The Feat_FloWer software package is accessible through the version control system *Git*. The main repository of the software is hosted on the servers of the TU Dortmund. In order to checkout a copy of the code an account that provides access to the LS3 servers is required, so it is recommended to get such an account. Furthermore, the code repository does not include the meshes used in the example applications. The meshes are stored in a different repository. So, at first we will get this repository and set up an environment variable so that the configuration mechanism of Feat_FloWer can find the mesh repository. The mesh repo can be checked out by:

```
> git clone ssh://username@lannister/home/user/git/mesh_repo.git
```

After you have checked out the mesh repository, it is time to set up the environment variable for Feat_FloWer. For the `BASH` or `ZSH` shells this can be done by adding the following line to your *.bashrc* or respectively *.zsh*:

```
> export Q2P1_MESH_DIR=/path/to/meshrepo/mesh_repo
```

1.2 Checking out the Code from the Repository

When you have acquired an appropriate account you can clone the Feat_FloWer repository by opening a terminal and entering the following sequence of commands:

```
> mkdir FeatFlower && cd FeatFlower
> git clone --recursive ssh://username@lannister/home/user/git/Feat_FloWer.git
```

This will create a parent folder called FeatFlower and within this folder the source

code will be contained. This is a preparation for building the code from source. The Feat_Flower code supports only out-of-source builds, meaning the binary files are not built in the same directory as the source code. This is done in order to prevent that binary files or object files litter the source directory or that these files show up in the version control system.

1.3 Building the Code from Source

The Feat_FloWer code can be build on Linux and Windows operating systems (and probably on MacOS, but this is not tested yet). On Linux systems an installation of the GNU compiler is needed, including the gfortran compiler and the matching OpenMPI libraries. The minimum requirement for the GCC is version 4.9.2. On Linux systems the Intel compiler can be used as an alternative compiler. On Windows systems the Intel compiler is a necessary requirement.

1.3.1 Linux Systems

After you have cloned the repository, navigate to the *FeatFlower* folder that you have created in the previous step and create another folder that will contain the binaries:
> mkdir bin
Verify with the command **pwd** that your FeatFlower folder now contains two subfolders *bin* and *Feat_FloWer*. At the core of the Feat_FloWer build system is the multi-platform build files generator CMake. On the servers of the TU Dortmund CMake is available as a loadable module. In order to successfully compile the code with the GNU compiler the following components are needed:

- GCC the GNU Compiler Collection including the gfortran compiler
- A matching OpenMPI installation
- CMake with a minimum version of 2.8

On the servers of the TU Dortmund load for example the following modules or newer versions of them:

- > module load gcc/6.1.0
- > module load openmpi/gcc6.1.x/1.10.2/non-threaded/no-cuda/ethernet
- > module load cmake/3.5.2-ssl

After you have typed these commands verify that these module are loaded by checking the output of the command **module list**. The build of the basic software package is initiated by navigating into the *bin* folder that you have created before and evoking CMake from there:

```
> cd bin
> cmake -DBUILD_METIS=True ../Feat_FloWer
```

The build process is then started by the command:

```
> make -j 5
```

The option <-j 5> starts a parallel build using 5 processes.

1.3.2 Windows Systems

Under construction

Chapter 2

Running a FeatFlower Application

2.1 Introduction to the Application Framework

A FeatFlower application is a particular simulation case that is based on the underlying Finite-Element-Method framework. The possibility of a FeatFloWer application range from very simple CFD-Setups to well-known benchmark configurations to complex realistic configurations that have an industrial or scientific background. The FeatFlower code base comes with a number of preconfigured applications that demonstrate the capabilities of the software. The user can then change and adapt these preconfigured applications to his needs or add completely new applications based on the Featflower framework. A FeatFloWer applications roughly consists of a preprocessing step, a solving procedure and an output of the solution. At first we will introduce the tools used in the preprocessing step.

2.1.1 Preprocessing and Mesh Generation

The main task in the preprocessing step is the creation of the mesh used for the FeatFloWer application. A mesh can be acquired by an external mesh generator, by manual construction or by taking a mesh from the FeatFloWer mesh repository.

2.1.1.1 Using Meshes from the Mesh Repository

Taking a mesh from the mesh repository is the easiest scenario for a new user. Usually this means that a benchmark configuration or a preconfigured application is run by the user. All the user has to do in this case is to partition the mesh according to the

number of MPI processes that the user has to his disposal. The preferred partitioning tool for meshes from the mesh repository is the Python command line program *PyPartitioner.py*. In order to demonstrate the use of the python partitioning tool, we will use the application *q2p1_fc_ext*. The application can be found in the *Feat_FloWer/applications* directory. The python partitioner tool can be found in the *Feat_FloWer/tools* directory. For our first contact with the tool, we will just copy the python files from the *Feat_FloWer/tools* directory to the *bin/applications/q2p1_fc_ext*. So start by navigating to the *bin/applications/q2p1_fc_ext* and copying the python files by:

```
> cp ../../../Feat_FloWer/tools/*.py .
```

Additionally, we need to copy the metis library so that the PyPartitioner can interface with that library:

```
> cp ../../../bin/extern/libraries/metis-4.0.3/Lib/libmetis.so .
```

We can now invoke the PyPartitioner by typing:

```
> python PyPartitioner.py 4 1 1 NEWFAC _adc/2D_FAC/2Dbench.prj
```

which will take the mesh referenced in the project file *_adc/2D_FAC/2Dbench.prj* and decompose the domain into 4 partitions which are then stored in the *NEWFAC* directory. The additional 1 1 parameters denote the type of the partitioning method. As the simulation requires a master node to control some processes the simulation should be launched with 4+1 processes by:

```
> mpirun -np 5 ./q2p1_fc_ext
```

2.2 Appendix

TODO