FEM Code (Python) Getting Started

Louis Remes

1 Prerequisites

Before you begin, let's ensure that your development environment is ready for Python coding. Most of the installation will be performed through **command lines**.

1.1 A code editor or IDE

We recommend **Visual Studio Code (VS Code)** for this guide, as all examples will be demonstrated using VS Code. However, you may use any editor or IDE that supports Python development (e.g., PyCharm, Sublime Text, Atom, or even a terminal-based editor like Vim).

☐ **Install VS Code.**

1.2 Python installed on your system

This project was developed and tested with Python 3.13. You can verify your installation by running commands:

```
python --version
```

or

```
python3 --version
```

If Python is not installed, follow the instructions below for your operating system:

- Windows (PowerShell, maybe run as admin, maybe run as admin):

  ```
  winget install --id Python.Python.3
  ```

  Or download the installer from python.org. Please **relaunch** PowerShell.

- macOS:

  ```
  brew install python3
  ```

  (Requires Homebrew to be installed.)

- Linux (Debian/Ubuntu):

  ```
  sudo apt update
  sudo apt install -y python3 python3.12-venv
  ```

- Linux (Fedora/RHEL/CentOS):

  ```
  sudo dnf install -y python3 python3-venv
  ```

☐ **Install Python 3 or newer.**

☐ **Verify the version with –version.**

1.3 Git installed on your system

Git is required to download the project repository. You can verify your installation by running:

```
git --version
```

If Git is not installed, use one of the following commands:

- Windows (PowerShell, maybe run as admin):

  ```
  winget install --id Git.Git
  ```

  Or download the installer from git-scm.com. Please **relaunch** PowerShell.

- macOS:

  ```
  brew install git
  ```

  (Requires Homebrew to be installed.)

- Linux (Debian/Ubuntu):

  ```
  sudo apt update
  sudo apt install git
  ```

- Linux (Fedora/RHEL/CentOS):

  ```
  sudo dnf install git
  ```

☐ **Install git.**

☐ **Verify the version with –version.**

1.4 Download the project repository

Once Git is installed, you can download the project repository using the `git clone` command. First, open a terminal or command prompt and navigate to the directory where you want the project to be saved. For example:

```
cd path/to/your/desired/directory
```

Then, run:

```
git clone https://github.com/LouisRemes-95/FEM-Project.git
```

This will create a new directory containing the project files inside your chosen directory, similar to:

```
fem_project/
|
+-- input_files/              # .py files defining problem data
|   \-- input_example.py
|
+-- provided_code/            # helper functions, FEM core routines already written
|   +-- solver.py
|   \-- plotting.py
|
+-- code_to_be_implemented/ # templates where students fill in missing parts
|   +-- assembly.py
|   +-- boundcond.py
|   +-- elements.py
|   \-- __init__.py
|
+-- main.py                   # main driver script to run a given input file
+-- requirements.txt
+-- setup_env.ps1
+-- setup_env.sh
\-- ...
```

☐ **Download the repository.**

1.5 Set up a Python virtual environment

Windows (PowerShell, maybe run as admin) note:

After cloning the repository and moving into its directory, you can set up your Python environment with the following script: `setup_env.ps1`.

To use the script:

1. Go to the project directory (the directory created by 'git clone')

   ```
   cd path/to/your/project (cd FEM_Project/)
   ```

2. Allows local scripts to run for this session only (first time only)

   ```
   Set-ExecutionPolicy RemoteSigned
   ```

3. Run the script

   ```
   ./setup_env.ps1
   ```

Unix/macOS/Linux or Git Bash/WSL on Windows:

After cloning the repository and moving into its directory, you can set up your Python environment with the following script: `setup_env.sh`.

To use the script:

1. Go to the project directory (the directory created by 'git clone')

   ```
   cd path/to/your/project (cd FEM_Project/)
   ```

2. Make the script executable (first time only)

   ```
   chmod +x setup_env.sh
   ```

3. Run the script

   ```
   ./setup_env.sh
   ```

Prefer a different setup? This script is provided for convenience. You can set up the environment however you like (e.g., *conda*, *pipenv*, or *poetry*), but it will be your responsibility to ensure it is configured correctly for this project.

☐ **Set up a virtual environment.**

1.6 Opening the Project in Visual Studio Code

Once you have cloned the repository and set up your Python environment, you can open the project in Visual Studio Code. Install the python extension on VSCode (Python language support with extension access points for IntelliSense).

Option 1: From the terminal

1. Navigate to the project directory (the directory created by 'git clone'):

   ```
   cd path/to/your/project
   ```

2. Launch VS Code in the current directory:

   ```
   code .
   ```

3. VS Code will open with the project files visible in the Explorer panel.

Option 2: From within VS Code

1. Open VS Code.
2. From the menu, select `File` → `Open directory...`.
3. Navigate to the directory containing your cloned repository and click `Select directory`.
4. The project files will appear in the Explorer panel on the left.

Ready to code You are now ready to run and debug the project directly from VS Code. From the terminal in VS Code, you should now be able to use:

`python -m provided_code.plotting input_example --what mesh` (this command only works in the .venv, to launch it in VSCode, just open a new terminal in VSCode, at the beginning of the line you should see (.venv))

A simple mesh should appear, and you're all set!

FEM Plotting Commands

This project provides several plotting utilities accessible via the command line or directly in Python scripts. All plots are based on data stored in the `outputs/` directory after running the solver, except for –what mesh wich is based on the `inputs/` directory data.

1. Command-Line Usage

The general command format is:

```
python -m provided_code.plotting <case_name> [OPTIONS]
```

Where:

- `<case_name>` = name of the case (directory in `outputs/`), e.g. `input_example`.
- `[OPTIONS]` are optional arguments controlling what is plotted.

Options

- `--what` : Selects the plot type. Choices:
  - `mesh` (default) undeformed mesh
  - `displ` displacement field
  - `strain` strain field
  - `stress` stress field
- `--component` : Field component to plot.
  - Displacement: `ux`, `uy`, `uz`, `mag` (default)
  - Strain: `exx`, `eyy`, `ezz`, `gyz`, `gxz`, `gxy`
  - Stress: `vm` (von Mises), `sxx`, `syy`, `szz`, `tyz`, `txz`, `txy`
- `--scale <float>` : Deformation scale factor (applied to displacements).
- `--no-show` : Do not open an interactive window (useful for saving).
- `--save <path>` : Save figure as PNG to given path.

Examples

- Plot undeformed mesh:

  ```
  python -m provided_code.plotting input_example --what mesh
  ```

- Plot deformed mesh with displacement magnitude, scaled by factor 10:

  ```
  python -m provided_code.plotting input_example --what displ --component mag --scale 10
  ```

- Plot strain component `exx`:

  ```
  python -m provided_code.plotting input_example --what strain --component exx --scale 5
  ```

- Plot von Mises stress without showing, save to file:

  ```
  python -m provided_code.plotting input_example --what stress --component vm \
         --scale 3 --no-show --save stress_plot.png
  ```

2. Python Script Usage

You can also call the plotting functions directly from a script or Jupyter notebook:

```
from provided_code.plotting import (
    plot_mesh, plot_displacement, plot_strain, plot_stress
)

# Plot undeformed mesh
plot_mesh("input_example")

# Plot deformed displacement magnitude (scale factor = 10)
plot_displacement("input_example", component="mag", scale=10)

# Plot strain component exx
plot_strain("input_example", component="exx", scale=5)

# Plot von Mises stress
plot_stress("input_example", component="vm", scale=3)
```

3. Notes

- The plotting functions automatically load:
  - Input mesh from `outputs/<case>/<case>.py`
  - Results from `outputs/<case>/<case>_results.npz`
- Displacement, strain, and stress plots require a completed solver run that generated `<case>_results.npz`.