Installation of FEniCSx on a Mac using Docker and VSCode

Eric M. Stewart and Jorge Nin

February 10, 2025

This document provides a brief set of instructions for

- installing FEniCSx on a Mac computer using Docker, and
- setting up Visual Studio Code (VSCode) to run and write FEniCSx codes as Jupyter notebooks.

We note that:

- This document provides specific instructions for installing the **v0.8.0 release of FEniCSx** — however, simple modifications to the commands presented permit installation of other versions.
- In general, these instructions work equally well for machines using an Intel / AMD processor or an M-series / Apple Silicon / ARM processor — however, we will point out some small modifications which are necessary for ARM machines.
- The basic installation process is essentially identical for Mac and Windows machines, but the screenshots in this version will be more directly helpful for Mac users.

1 Installing Docker

We will install FEniCSx using the container management software Docker. For background information about Docker and the role it plays in running FEniCSx codes, see Appendix A. Install the appropriate version of Docker Desktop for your machine as listed on the Docker website:

- `https://www.docker.com/products/docker-desktop/`

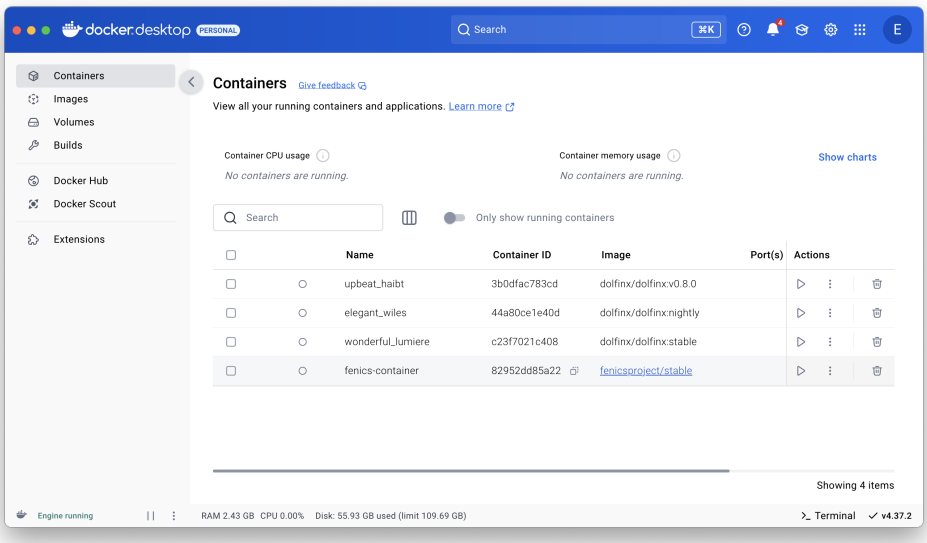

Figure 1: The Docker dashboard.

The Docker Dashboard shows all your **containers**, and will be empty the first time you open Docker.

- A **container** is a virtual machine with desired software installed, and is built from a certain **image**.[1]
- An **image** is a collection of files necessary to run specific softwares (such as FEniCSx).

Note that

- You can use the "recommended" settings when initially setting up Docker.
- Macs with an M-series / Apple Silicon / ARM processor need to install Rosetta before using Docker. You will be prompted to install Rosetta when opening Docker for the first time — click `install`.
  - Alternatively, you can manually install Rosetta using the terminal command `softwareupdate --install-rosetta`.
- You may be asked to sign in, but an account is not necessary for the features we will use. The free "Personal" Docker plan is sufficient for running and writing FEniCSx codes.
- If Docker is running but you can't find the dashboard, right-click the Docker icon in the menu bar at the top of your screen and select `"Go to the Dashboard"`

Remark. Recently, for Mac users, there have been some issues with MacOS mis-interpreting Docker as containing malware. To prevent this issue, Docker suggests making sure your Docker installation is up-to-date each time before updating your MacOS. See e.g. `https://www.docker.com/blog/incident-update-docker-desktop-for-mac/` for guidance.

- If you have trouble getting MacOS to stop treating Docker as malware, try the instructions at `https://docs.docker.com/desktop/cert-revoke-solution/`.

1.1 Recommended Settings

We recommend using the following custom settings, using the gear icon in the upper-right:

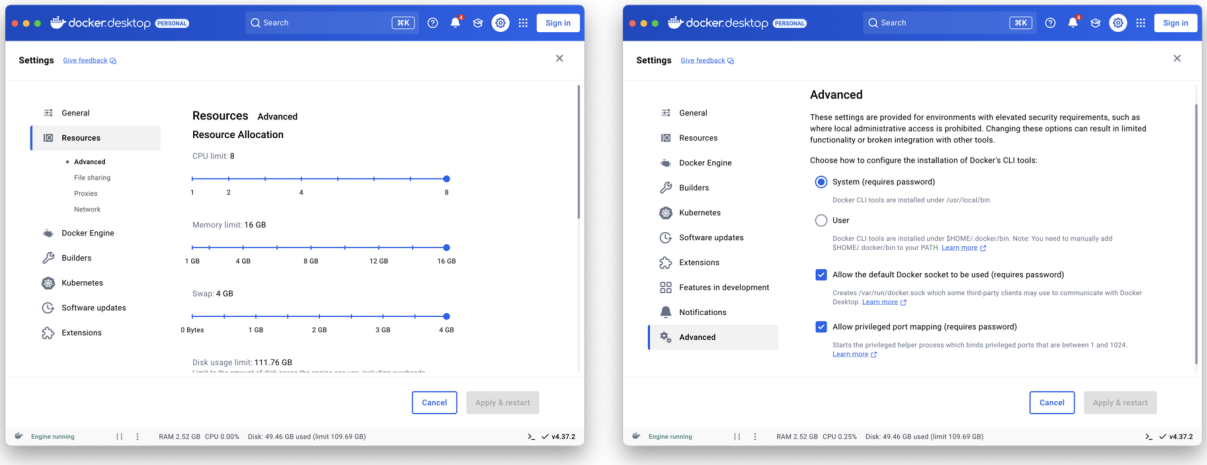

Figure 2: The Docker Settings tab. (Left) Resources tab. (Right) Advanced settings tab.

[1] A container is not actually a virtual machine in the traditional sense, but the term is helpful for understanding. See, e.g. `https://www.docker.com/blog/containers-are-not-vms/`

That is,

- In the **Resources** tab, maximize the number of CPUs, memory, and swap available to containers.
  - Finite-element simulations will benefit from having the maximum compute and memory resources available. Increased memory in particular can be highly impactful to improving solution time of large problems.
- In the **Advanced** tab, match the option selections shown in the right panel of Fig. 2.
  - These settings are needed to ensure compatibility of VSCode with Docker on some machines.

2 FEniCSx Container Setup

In this section we:

- Set up a FEniCSx image and container so that it runs properly.
- Mount a folder in the container so that it is shared between the host machine and Docker container.

2.1 Creating a working folder for FEniCSx codes

Before installing FEniCSx,

1. Create a working folder for FEniCSx codes called `FEniCSWrk`.
   - Place `FEniCSWrk` anywhere convenient for you; for example I use `/Users/eric/Desktop/FEniCSWrk`
2. To prepare to run an example code, create sub-folders in `FEniCSWrk` called:
   - `/FEniCSWrk/example`
   - `/FEniCSWrk/example/results`

2.2 Building a FEniCSx v0.8.0 installation

1. Pull the Docker image of the v0.8.0 release of dolfinx using this command in the MacOS terminal:

```
> Docker pull dolfinx/dolfinx:v0.8.0
```

2. Create a Docker container with a shared volume with the following command:

```
> Docker run -it -v /Users/eric/Desktop/FEniCSWrk:/home/shared dolfinx/dolfinx:v0.8.0
```

   - Replace the example filepath `/Users/eric/Desktop/FEniCSWrk` with the location of the working folder you created in Section 2.1.
   - After running this command, the container will be running in the terminal. Type in `exit` to leave MacOS terminal.
3. Open up the Docker Desktop dashboard. Make sure the new environment is running (the one with the `dolfinx/dolfinx:v0.8.0` image), as shown in Fig. 3. Open up the command line interface for this container by clicking the three-dots icon and selecting "Open in terminal" as shown in Fig. 3.

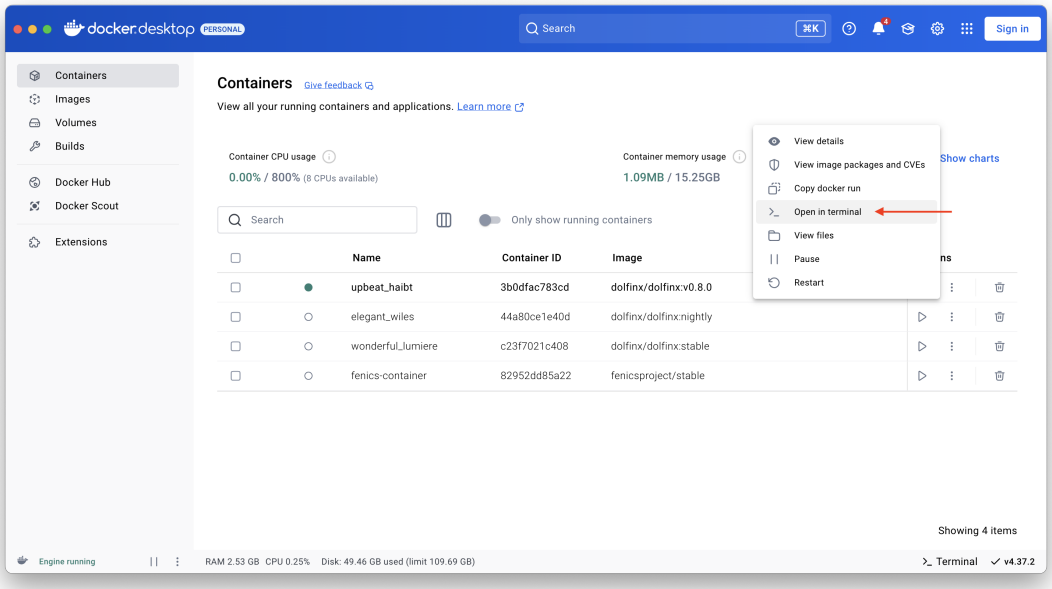

Figure 3: Opening the command line interface in Docker Desktop.

4. In the container's terminal window, run the following commands. First,

```
apt-get update && apt-get clean && apt-get install -y xvfb vim ffmpeg
```

   then, and this one line is only necessary if you are using an M-series / Apple silicon Mac,

```
pip3 install "https://github.com/finsberg/vtk-aarch64/releases/download/vtk-9.2.6-cp310
    /vtk-9.2.6.dev0-cp310-cp310-linux_aarch64.whl"
```

   Finally, (for both M-series and Intel Macs),

```
pip3 install "pyvista[all,trame]" jupyterlab ipython ipywidgets
```

5. Your Docker container is now properly set up for running FEniCSx (version 0.8.0) codes. Congratulations!

You can connect to this container in an instance of VSCode using the instructions below.

3 Editing and running code in Visual Studio Code

We use Visual Studio Code (VSCode) to edit and run FEniCSx codes in the form of Jupyter notebooks (.ipynb) and Python scripts (.py). If you do not have VScode, install it on your system:

- `https://code.visualstudio.com/`

Open up VScode, click the extensions tab (icon on the far left with three blocks and a fourth block clicking into place). Search for the **Docker** extension published by Microsoft shown in the Figure below, and install it. Also, search for and install the **Dev Containers** extension which allows us to use the full suite of VSCode development tools within a container.

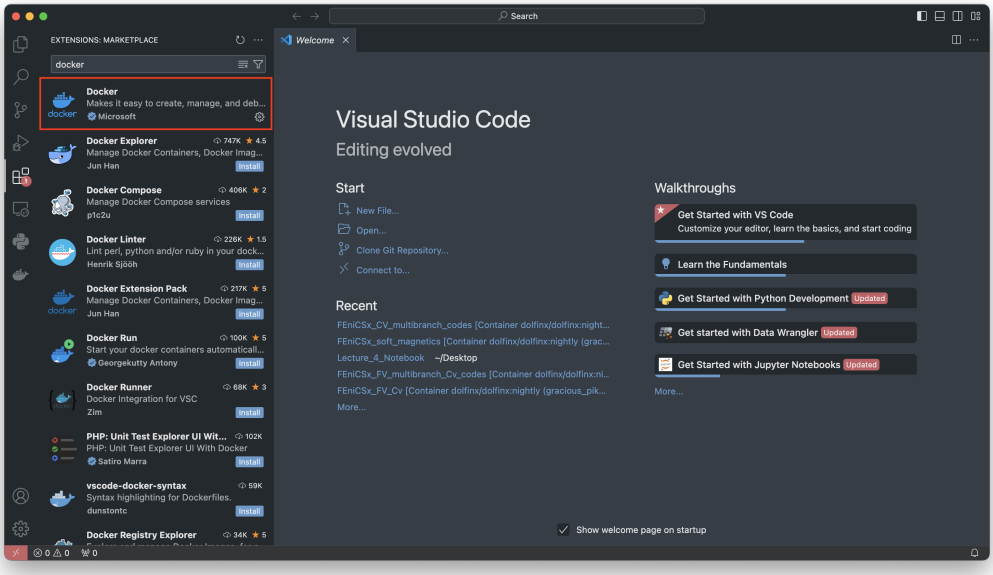

Figure 4: Installing the Docker extension.

3.1 Connecting to the Container through VSCode

We now launch VSCode and connect to the container we just created.

1. We can go to the Docker extension, and find the container we just launched. Right clicking on any of the containers will allow us to attach a VSCode instance.

2. Right click on the "v0.8.0" FEniCSx container and then click `Attach Visual Studio Code` as shown in Fig. 5. This will open a new window of VSCode.

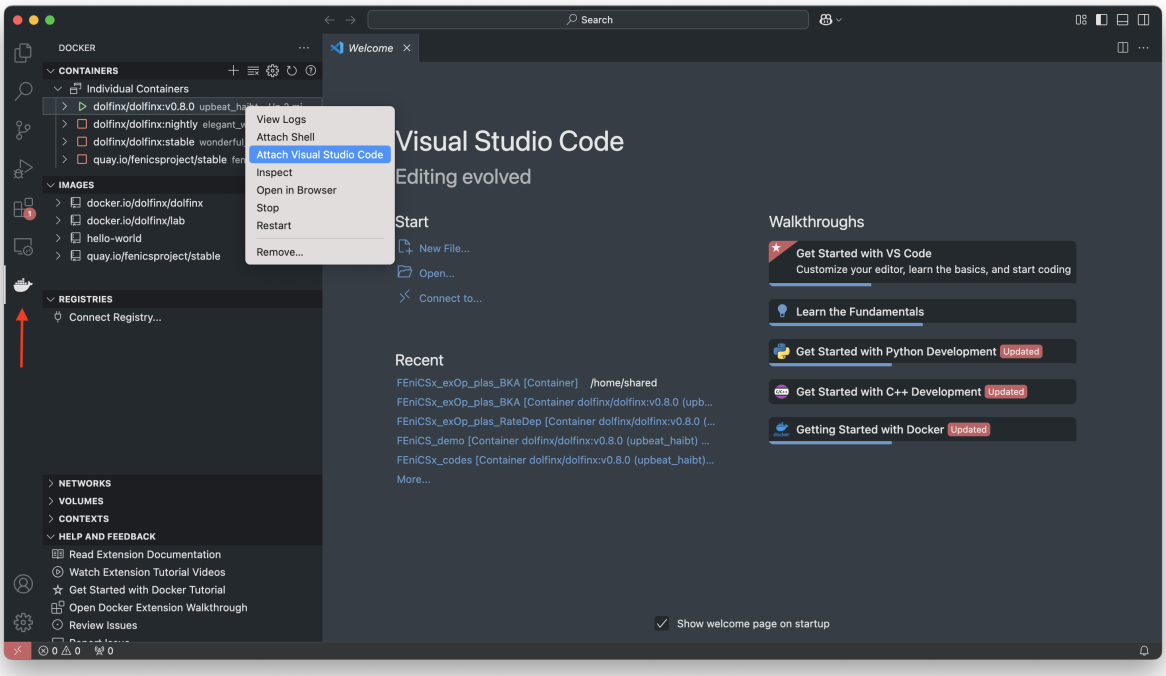

Figure 5: Right clicking on any of the containers in the Docker extension will allow us to attach a VSCode instance.

3. Ensure we are connected by looking at the bottom left corner of the screen.

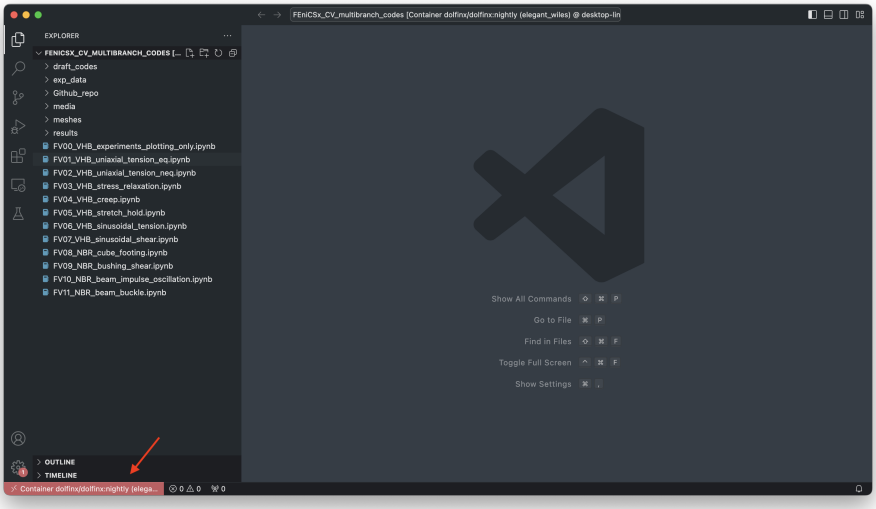

Figure 6: A VSCode instance which is "attached" to our v0.8.0 FEniCSx container, as indicated by the red arrow.

4. You can now close the original instance of VSCode which is not connected to the Docker container.

5. In the instance of VSCode connected to the Docker container, search for the **Python** and **Jupyter** extensions and install them. Installing these extensions will also automatically install their dependencies, which contain the tools necessary to run Jupyter notebooks and produce interactive in-line plots.[2]

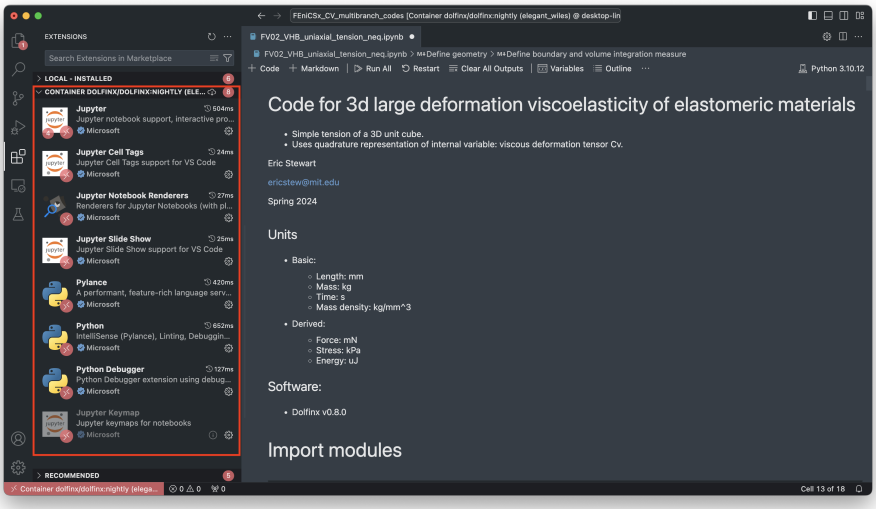

Figure 7: Installing the **Python** and **Jupyter** extensions will result in all necessary dependencies — the 8 individual extensions shown — being installed in the container.

Congratulations, you are now set up to run and write FEniCSx codes with VSCode!

Move on to the next section to run an example code, and verify your installation.

[2]Some extensions may require you to reload the VScode window and re-attach to the Docker container to install properly.

3.2 Running an example FEniCSx code

Next, run an example code in order to verify your installation:

1. Download a FEniCSx code such as, for example:

   - `https://github.com/SolidMechanicsCoupledTheories/FEniCSx_codes/blob/main/02_finite_viscoelasticity/FV08_NBR_cube_footing.ipynb`

   which simulates viscoelastic "footing" compression of an elastomeric cube.

2. Move the Jupyter notebook file (`*.ipynb`) to your `FEniCSWrk/example` folder, created in Section 2.1.

   - Since the container can only access files on your host machine which are in the shared folder, FEniCSx codes must be in the `FEniCSWrk` folder in order to be run.
   - Ensure that you have an empty folder `FEniCSWrk/example/results`, to be used for output of the simulation results files.

3. In VSCode, select `File -> Open Folder` and navigate to or type in the shared folder filepath `/home/shared`. This is the name of your `FEniCSWrk` folder inside the Docker container.

   - Note that you may have to move up through the file directory using the (`..`) entry before you are able to find the `/home/` folder.

4. Once in the `/home/shared` directory, open your Jupyter notebook file using the navigation pane on the left side of the VSCode window.

5. Press the `Run all` button at the top of the page to run all the cells in the Jupyter notebook, or you can run each cell individually using the command `shift + return`.

   - The first time running a notebook file, VSCode will prompt you to select a Python environment. Simply select the recommended Python environment.

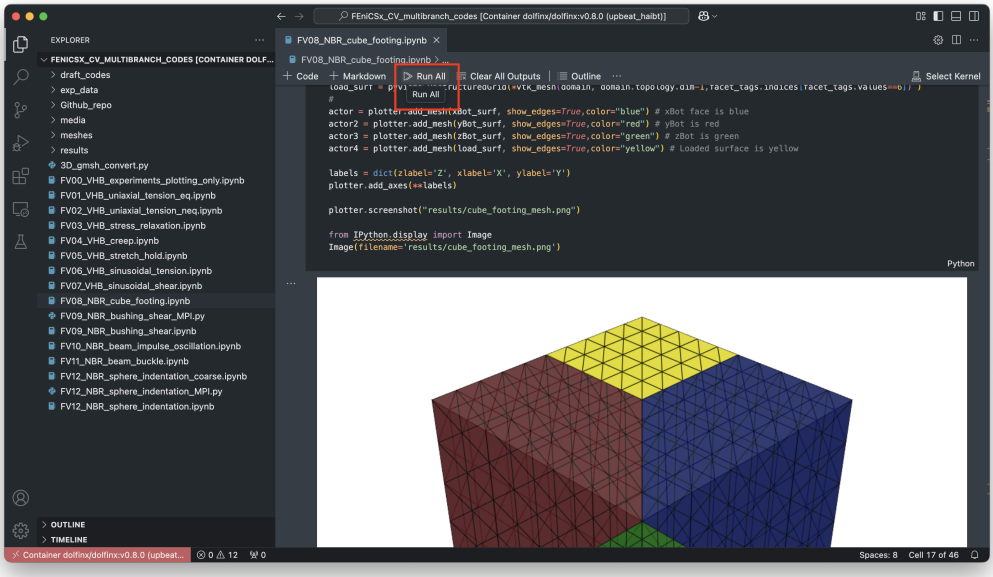

Figure 8: Running an example FEniCSx Jupyter notebook in VSCode, using the `Run all` option.

6. Wait for the code to finish running, and scroll through the notebook to view the results.

7. Results of the simulation are post-processed and visualized in two primary ways:

   (a) Plots of curves are constructed and shown in-line using `matplotlib`. The example FEniCSx code above produces the following plots using the `matplotlib` package:

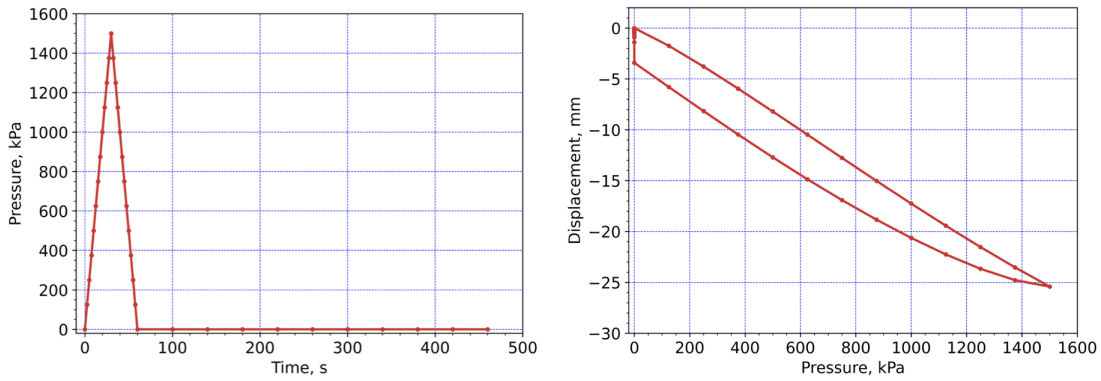

Figure 9: Example `matplotlib` plots of simulation results.

   (b) Field data for the simulation — i.e. contours of displacement, stress, and so on — are written to a file which can be read and visualized by ParaView.

   - Download ParaView here: `https://www.paraview.org/download/`.
   - The file `results/3D_cube_footing.bp` can be opened in Paraview to create visualizations of the simulation results, such as the contour plot shown below.[3]
     - The first time a `*.bp` results file is opened, ParaView will ask which protocol to use to interpret the file. Select the option `ADIOS2VTXReader`, and then select `Set as Default`.

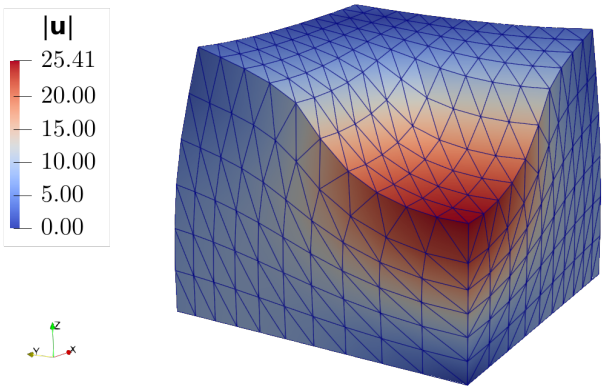

Figure 10: Example ParaView visualization of simulation results.

[3]In this visualization, the "Warp by vector" filter has been used to show the deformed body.

4 Quick usage guide for the FEniCSx container

After the container has first been created and set up as described in this document, you can use the Docker container to run FEniCSx codes in the future by following these steps.

1. Start up Docker Desktop. In the Docker dashboard, Click the "play" button next to your `dolfinx/dolfinx:v0.8.0` container to start it.
2. Open VSCode. Attach an instance of VSCode to the running FEniCSx Docker container using the Docker extension (cf. Section 3.1).
3. Move any FEniCSx codes you would like to run into the `FEniCSWrk` directory on your host machine.
4. In the VSCode window connected to your FEniCSx container, navigate to your `FEniCSWrk` directory `/home/shared` using `File -> Open Folder`, and open up and run the FEniCSx code of your choice.
    - The first time you run a code, a pop-up menu may ask you to choose which Python kernel to use. Select the (recommended) option for the python kernel installed in the Docker container, usually in the location `/usr/bin/python3`.

Note that:

- The first time you attach VSCode to this container, you will have to install the **Python** and **Jupyter** extensions (which automatically also installs their dependencies, a total of 8 packages).
- After these packages have been installed once, future instances of VSCode that attach to this container will already have the extensions installed.

A About Docker

Docker is a platform that uses containerization technology to simplify the deployment and management of software applications. In the context of software development, and especially for complex software applications like FEniCSx, Docker offers several advantages:

- Containerization:
  - Containers are isolated environments in which applications can run. They are lightweight because they share the host system's kernel — but not the OS itself — and are not burdened with the overhead of a true virtual machine.
  - For FEniCSx, this means that each instance of the application runs in a consistent environment, irrespective of the underlying hardware or software configurations of the host system.
- Consistency and reproducibility:
  - Docker ensures that FEniCSx runs identically on every machine, avoiding the common"it works on my machine" problem in software development.
- Ease of distribution and version control:
  - Docker images, which are essentially blueprints for containers, can be version-controlled and shared through Docker Hub or other registries. This ensures that all users have access to the same version of FEniCSx and its dependencies.

Because of these advantages, we use Docker to run FEniCSx.