Chapter 1
Preliminaries

1.1 The FEniCS Project

The FEniCS Project is a research and software project aimed at creating mathematical methods and software for automated computational mathematical modeling. This means creating easy, intuitive, efficient, and flexible software for solving partial differential equations (PDEs) using finite element methods. FEniCS was initially created in 2003 and is developed in collaboration between researchers from a number of universities and research institutes around the world. For more information about FEniCS and the latest updates of the FEniCS software and this tutorial, visit the FEniCS web page at `https://fenicsproject.org`.

FEniCS consists of a number of building blocks (software components) that together form the FEniCS software: DOLFIN [27], FFC [17], FIAT [16], UFL [1], mshr, and a few others. For an overview, see [26]. FEniCS users rarely need to think about this internal organization of FEniCS, but since even casual users may sometimes encounter the names of various FEniCS components, we briefly list the components and their main roles in FEniCS. DOLFIN is the computational high-performance C++ backend of FEniCS. DOLFIN implements data structures such as meshes, function spaces and functions, compute-intensive algorithms such as finite element assembly and mesh refinement, and interfaces to linear algebra solvers and data structures such as PETSc. DOLFIN also implements the FEniCS problem-solving environment in both C++ and Python. FFC is the code generation engine of FEniCS (the form compiler), responsible for generating efficient C++ code from high-level mathematical abstractions. FIAT is the finite element backend of FEniCS, responsible for generating finite element basis functions, UFL implements the abstract mathematical language by which users may express variational problems, and mshr provides FEniCS with mesh generation capabilities.

H.P Langtangen and A. Logg, *Solving PDEs in Python*,
Simula SpringerBriefs on Computing 3, DOI 10.1007/978-3-319-52462-7_1

1.2 What you will learn

The goal of this tutorial is to demonstrate how to apply the finite element to solve PDEs in FEniCS. Through a series of examples, we demonstrate how to:

- solve linear PDEs (such as the Poisson equation),
- solve time-dependent PDEs (such as the heat equation),
- solve nonlinear PDEs,
- solve systems of time-dependent nonlinear PDEs.

Important topics involve how to set boundary conditions of various types (Dirichlet, Neumann, Robin), how to create meshes, how to define variable coefficients, how to interact with linear and nonlinear solvers, and how to postprocess and visualize solutions.

We will also discuss how to best structure the Python code for a PDE solver, how to debug programs, and how to take advantage of testing frameworks.

1.3 Working with this tutorial

The mathematics of the illustrations is kept simple to better focus on FEniCS functionality and syntax. This means that we mostly use the Poisson equation and the time-dependent diffusion equation as model problems, often with input data adjusted such that we get a very simple solution that can be exactly reproduced by any standard finite element method over a uniform, structured mesh. This latter property greatly simplifies the verification of the implementations. Occasionally we insert a physically more relevant example to remind the reader that the step from solving a simple model problem to a challenging real-world problem is often quite short and easy with FEniCS.

Using FEniCS to solve PDEs may seem to require a thorough understanding of the abstract mathematical framework of the finite element method as well as expertise in Python programming. Nevertheless, it turns out that many users are able to pick up the fundamentals of finite elements *and* Python programming as they go along with this tutorial. Simply keep on reading and try out the examples. You will be amazed at how easy it is to solve PDEs with FEniCS!

1.4 Obtaining the software

Working with this tutorial obviously requires access to the FEniCS software. FEniCS is a complex software library, both in itself and due to its many dependencies to state-of-the-art open-source scientific software libraries. Manually building FEniCS and all its dependencies from source can thus be a daunting task. Even for an expert who knows exactly how to configure and build each component, a full build can literally take hours! In addition to the complexity of the software itself, there is an additional layer of complexity in how many different kinds of operating systems (Linux, Mac, Windows) may be running on a user's laptop or compute server, with different requirements for how to configure and build software.

For this reason, the FEniCS Project provides prebuilt packages to make the installation easy, fast, and foolproof.

FEniCS download and installation

In this tutorial, we highlight two main options for installing the FEniCS software: Docker containers and Ubuntu packages. While the Docker containers work on all operating systems, the Ubuntu packages only work on Ubuntu-based systems. Note that the built-in FEniCS plotting does currently not work from Docker, although rudimentary plotting is supported via the Docker Jupyter notebook option.

FEniCS may also be installed using other methods, including Conda packages and building from source. For more installation options and the latest information on the simplest and best options for installing FEniCS, check out the official FEniCS installation instructions. These can be found at `https://fenicsproject.org/download`.

FEniCS version: 2016.2

FEniCS versions are labeled 2016.1, 2016.2, 2017.1 and so on, where the major number indicates the year of release and the minor number is a counter starting at 1. The number of releases per year varies but typically one can expect 2–3 releases per year. This tutorial was prepared for and tested with FEniCS version 2016.2.

1.4.1 Installation using Docker containers

A modern solution to the challenge of software installation on diverse software platforms is to use so-called *containers*. The FEniCS Project provides custom-made containers that are controlled, consistent, and high-performance software environments for FEniCS programming. FEniCS containers work equally well[1] on all operating systems, including Linux, Mac, and Windows.

To use FEniCS containers, you must first install the Docker platform. Docker installation is simple and instructions are available on the Docker web page[2]. Once you have installed Docker, just copy the following line into a terminal window:

Terminal

```
Terminal> curl -s https://get.fenicsproject.org | bash
```

The command above will install the program `fenicsproject` on your system. This program lets you easily create FEniCS sessions (containers) on your system:

Terminal

```
Terminal> fenicsproject run
```

This command has several useful options, such as easily switching between the latest release of FEniCS, the latest development version and many more. To learn more, type `fenicsproject help`. FEniCS can also be used directly with Docker, but this typically requires typing a relatively complex Docker command, for example:

Terminal

```
docker run --rm -ti -v `pwd`:/home/fenics/shared -w
/home/fenics/shared quay.io/fenicsproject/stable:current '/bin/bash -l
-c "export TERM=xterm; bash -i"'
```

Sharing files with FEniCS containers

When you run a FEniCS session using `fenicsproject run`, it will automatically share your current working directory (the directory from

[1] Running Docker containers on Mac and Windows involves a small performance overhead compared to running Docker containers on Linux. However, this performance penalty is typically small and is often compensated for by using the highly tuned and optimized version of FEniCS that comes with the official FEniCS containers, compared to building FEniCS and its dependencies from source on Mac or Windows.

[2] `https://www.docker.com`

which you run the `fenicsproject` command) with the FEniCS session. When the FEniCS session starts, it will automatically enter into a directory named `shared` which will be identical with your current working directory on your host system. This means that you can easily edit files and write data inside the FEniCS session, and the files will be directly accessible on your host system. It is recommended that you edit your programs using your favorite editor (such as Emacs or Vim) on your host system and use the FEniCS session only to run your program(s).

1.4.2 Installation using Ubuntu packages

For users of Ubuntu GNU/Linux, FEniCS can also be installed easily via the standard Ubuntu package manager `apt-get`. Just copy the following lines into a terminal window:

Terminal

```
Terminal> sudo add-apt-repository ppa:fenics-packages/fenics
Terminal> sudo apt-get update
Terminal> sudo apt-get install fenics
Terminal> sudo apt-get dist-upgrade
```

This will add the FEniCS package archive (PPA) to your Ubuntu computer's list of software sources and then install FEniCS. It will will also automatically install packages for dependencies of FEniCS.

Watch out for old packages!

In addition to being available from the FEniCS PPA, the FEniCS software is also part of the official Ubuntu repositories. However, depending on which release of Ubuntu you are running, and when this release was created in relation to the latest FEniCS release, the official Ubuntu repositories might contain an outdated version of FEniCS. For this reason, it is better to install from the FEniCS PPA.

1.4.3 Testing your installation

Once you have installed FEniCS, you should make a quick test to see that your installation works properly. To do this, type the following command in a FEniCS-enabled[3] terminal:

Terminal

```
Terminal> python -c 'import fenics'
```

If all goes well, you should be able to run this command without any error message (or any other output).

1.5 Obtaining the tutorial examples

In this tutorial, you will learn finite element and FEniCS programming through a number of example programs that demonstrate both how to solve particular PDEs using the finite element method, how to program solvers in FEniCS, and how to create well-designed Python code that can later be extended to solve more complex problems. All example programs are available from the web page of this book at `https://fenicsproject.org/tutorial`. The programs as well as the source code for this text can also be accessed directly from the Git repository[4] for this book.

1.6 Background knowledge

1.6.1 Programming in Python

While you can likely pick up basic Python programming by working through the examples in this tutorial, you may want to study additional material on the Python language. A natural starting point for beginners is the classic *Python Tutorial* [11], or a tutorial geared towards scientific computing [22]. In the latter, you will also find pointers to other tutorials for scientific computing in Python. Among ordinary books we recommend the general introduction *Dive into Python* [28] as well as texts that focus on scientific computing with Python [15, 18–21].

[3] For users of FEniCS containers, this means first running the command `fenicsproject run`.

[4] `https://github.com/hplgit/fenics-tutorial/`

Python versions

Python comes in two versions, 2 and 3, and these are not compatible. FEniCS works with both versions of Python. All the programs in this tutorial are also developed such that they can be run under both Python 2 and 3. Python programs that need to print must then start with

```
from __future__ import print_function
```

to enable the `print` function from Python 3 in Python 2. All use of `print` in the programs in this tutorial consists of function calls, like `print('a:', a)`. Almost all other constructions are of a form that looks the same in Python 2 and 3.

1.6.2 The finite element method

Many good books have been written on the finite element method. The books typically fall in either of two categories: the abstract mathematical version of the method or the engineering "structural analysis" formulation. FEniCS builds heavily on concepts from the abstract mathematical exposition. The first author has a book[5] [24] in development that explains all details of the finite element method in an intuitive way, using the abstract mathematical formulations that FEniCS employs.

The finite element text by Larson and Bengzon [25] is our recommended introduction to the finite element method, with a mathematical notation that goes well with FEniCS. An easy-to-read book, which also provides a good general background for using FEniCS, is Gockenbach [12]. The book by Donea and Huerta [8] has a similar style, but aims at readers with an interest in fluid flow problems. Hughes [14] is also recommended, especially for readers interested in solid mechanics and heat transfer applications.

Readers with a background in the engineering "structural analysis" version of the finite element method may find Bickford [3] an attractive bridge over to the abstract mathematical formulation that FEniCS builds upon. Those who have a weak background in differential equations in general should consult a more fundamental book, and Eriksson *et al* [9] is a very good choice. On the other hand, FEniCS users with a strong background in mathematics will appreciate the texts by Brenner and Scott [5], Braess [4], Ern and Guermond [10], Quarteroni and Valli [29], or Ciarlet [7].

[5] `http://hplgit.github.io/fem-book/doc/web/index.html`