



## LUND Adaptive Finite Element Methods

UNIVERSITY Centre for Mathematical Sciences

---

### Deal.II Software

Topics: Installing deal.II and running examples on Linux and OSX platforms.

---

**Background:** The deal.II open source software is a C++ program library aimed for solving partial differential equations using finite element techniques. Since this library is quite large and extensive and installing can be tricky, you can use this introduction as a step by step manual for installing and running the examples. Also note that this information is also available through the deal.II homepage. There you can find detailed information on supported platforms, library dependancies etc. The next part of the introduction will focus on how to install the library on Linux and OSX platforms. If you have any problems completing these steps on your own computer we recommend that you try it on the Linux computers in the math building (MH:230), where its been verified.

**Installing Software:** Since this is an open source C++ software it is possible to build and run it on any platform, but we have chosen to focus on Linux and OSX platforms for now. The deal.II software comes with a configure script and makefiles to simplify setup and compilation of the code but the first thing you need to do is of course download the software (deal.II 6.2.1 Full Release at the moment). This can be found on the deal.II homepage. Create a new catalog, i.e. `./deal/` in your home directory and unzip the downloaded package there. Open a Terminal window and go to the unpacked deal.II catalog (i.e. `cd /deal/deal.II/` or similar). Run the configure script by typing:

```
>./configure
```

This script will check your software and hardware setup and prepare the installation. When this task is completed successfully type:

```
>make all
```

This command will now setup, compile and link the deal libraries and create easy to use makefiles for all of the included examples. The last ones will be of particular interest to us when we will run and edit the examples. On the OSX platform you have to add the path to the library to your PATH manually in your `.bash_profile` in your home directory. The code you

need to add should be automatically generated by the make all command and is shown at the final output (should look something like this: "export DYLD\_LIBRARY\_PATH=\$DYLD\_LIBRARY\_PATH:/Users/mathias/MYPATH/deal.II/lib").

**OBS!** These steps can take a long time to complete (5-15min on a new computer and 1-2h on a old computer) so be sure to prepare these steps in good time before you aim to solve the examples! You should now have a functioning deal.II library installed so let's try and run an example.

**Running Examples:** Lets start by running the first example in the deal.II tutorial. You can find the documentation and commented source code on the deal.II homepage. The first example creates meshes for two 2-D geometries. Open a terminal window and go to deal/deal.II/examples/step-1/. To compile and run the example just type:

```
>make
```

and the make script takes care of the compiler options and linking and running the program. The output from the examples can be found inside the example catalog. Feel free to browse around the examples and try to run them. On the deal.II homepage the examples can also be found grouped by topic.

**Assignments:** For the assignment you basically need to run and change some parameters of the step-14 example. I would recommend that you make a copy of the entire /step-14/ (i.e. /step-14BACKUP/) and work directly on the code in the /step-14 catalog. In this way you don't need to change anything in the make files and all you need to do is type make again to run your program and you have a backup if you accidentally destroy the code. This code (and all other examples) have a specific structure with the "main" program at the very end of the file. Find the "main" program in the end section of the code in the file step-14.cc. Then find the following definition:

```
const Point<dim> evaluation_point (0.75, 0.75);
```

By changing the values of the evaluation\_point(x,y) you can choose any point inside the domain. Also note that in the online documentation of the example there is a good introduction to dual based error estimation.

**Good luck and happy computing!**