Université de Bourgogne Franche-Comté

Introduction to COMSOL: Microelectromechanical gyroscope

Authors:
Christophe RAMSEYER
Noah PERREAU

December 9, 2021

Contents

1 Introduction

1.1 Angular momentum and gyroscopes

If you recall your courses of classical mechanics, there is a nice symmetry between linear and rotational motion. In rotational motion, we are mostly interested in *angular momentum*, which is similar to linear momentum: it is one way to represent the "amount of motion." Angular momentum is defined in the following way:

$$\overrightarrow{L} = \overrightarrow{OM} \times \overrightarrow{p}, \tag{1}$$

where $\overrightarrow{p}$ is the linear momentum of point M and $\overrightarrow{OM}$ is the position of M relative to some origin O. It is straightforward to generalize this to a set of points, wether discrete or continuous (like a solid or a fluid).

By differentiating angular momentum, one obtains an equation similar to Newton's second law but for circular motion:

$$\begin{aligned} \frac{d\overrightarrow{L}}{dt} &= \frac{d\overrightarrow{OM}}{dt} \times \overrightarrow{p} + \overrightarrow{OM} \times \frac{d\overrightarrow{p}}{dt} \\ &= \overrightarrow{OM} \times \frac{d\overrightarrow{p}}{dt} = \overrightarrow{OM} \times \overrightarrow{F}, \end{aligned} \tag{2}$$

By analogy, we introduce the torque $\overrightarrow{\tau}$ to obtain the second law for rotational motion:

$$\frac{d\overrightarrow{L}}{dt} = \overrightarrow{\tau} \tag{3}$$

This differential equation shows something similar to Newton's second law: momentum, wether linear or angular, quantifies the *resistance* to a change in linear or rotational motion, respectively. In the case of perfectly rigid bodies, linear and rotational motion can be treated independently and summarized as six degrees of freedom attached to a reference frame, usually located at the bodie's center of mass. In the case of deformable bodies, each point in the system becomes a six dimensional system in its own right, potentially interacting in arbitrary ways with any other point. Solving for their collective motion is therefore significantly more complicated. Fortunately, software like COMSOL can do just that.

Angular momentum plays a crucial role in many areas of physics, and easily leads to complex and non-intuitive behavior. In quantum mechanics it arises both as a direct classical equivalent (for instance, the circular motion of an electron around a nuclei), and as a more abstract intrinsic property of particles called *spin*[1]. Counting and quantifying angular momentum is central to understanding the electronic structure of molecules and solids. Even in classical mechanics, angular momentum is fundamental to many systems of practical interest, among which the *gyroscope*.

[1] *Spin* being a notoriously bad name, since nothing indicates that particles do indeed spin about themselves. Particles just happen to have an intrinsic angular momentum unrelated to circular motion, to the best of our knowledge. This view has been well justified experimentally and theoretically.

In its most basic form, a gyroscope is just a rotating wheel. Thanks to the conservation of angular momentum (which is just another way of saying that you have to apply torque to change it), the rotating wheel will keep the same orientation as long as it keeps rotating. If the wheel is very heavy and rotates fastly enough, angular momentum will be high and you will indeed need to apply quite a large torque to change its orientation. Commonly, a gyroscope is made of such a rotating wheel attached to a frame with three axis free to rotate relative to each other. If the friction between the three axis of the frame is low enough, changing the orientation of the frame will cause the three axis to rotate in order for the rotating wheel to keep its orientation.

To summarize, gyroscopes can be seen as devices resisting a change in orientation as a direct consequence of the conservation of angular momentum. This property can be used in two opposite ways, both useful:

- To *stabilize* a system, by rigidly attaching a gyroscope to it, and having it spin at some constant and high enough speed. Since the attachment is (hopefully) rigid enough, the gyroscope will carry the system with it and therefore also resist against its rotation.
- To *measure* rotation, in which case the gyroscope is free to rotate, but its orientation is measured against a fixed system (for instance, the wall of a box containing the gyroscope, since it can be quite fragile). Seen from this fixed system, it is as if the gyroscope did not move along it when rotating; equivalently, the gyroscope provides a *reference axis*. Such mechanical gyroscopes have been used for years in planes, but more recently, miniaturized versions have been used in embedded devices. This is used for this neat, and sometimes also quite annoying feature of smartphones and tablets, to cause the interface to rotate when you rotate the device.

1.2 Microelectromechanical systems (MEMS)

Microlithography is the large set of techniques used to print and etch patterns in matter at a microscopic and nanometric scale. These technologies have been used for more than sixty years to manufacture integrated circuits of exponentially growing complexity, but more recently have also been applied to more exotic microsystems. One of these modern applications are *microelectromechanical systems* (MEMS): microscopic systems involving an interplay of classical mechanics and electricity. At this scale, the balance between the strength of various forces is different than our scale, sometimes leading to unexpected but interesting and useful behavior.

A huge variety of such systems have been designed and continue to be imagined, but a very simple classification separates them into two groups:

- *actuators* are designed to act on their surroundings or another system, usually in response to an electrical command. This action is commonly mechanical, but could be anything designed to influence the state of the system we are interested in (for instance, an indicator LED or an antenna).
- *sensors* on the other hand measure properties from their surroundings, creating information. Essentially each physical aspect or phenomena has its set of sensors,

to be used in specific ranges of conditions. To mention a few, there are sensors to measure temperature, force, pressure, altitude, orientation, acceleration, or even the composition of some gas.

Both sensors and actuators tend to be used in the design of a single system, in addition to a processor which controls what commands to emit in response to a given set of measurements. Understanding such systems is the role of *automation* and *control theory*, which quantify the unwillingness of the Universe to do what we want, and how to force it to do so nonetheless.

1.3 Vibrating gyroscopes

Rotational motion is central to gyroscopes, but rotation is often complex to implement concretely, especially if you have to have electrical contact with the rotating body. A simpler alternative is used to make MEMS gyroscopes: *vibrating structure gyroscopes*[2] exploit the effects of the Coriolis force on the vibrational modes of a system.

The vibrational modes of a system are the generalization of the reasonant frequency of simple spring (following Hooke's law) to continuous, elastic solids. Essentially, you can consider that each point in the solid is connected to nearby point by a small spring, of which the mass and stiffness depend on the material (those properties can be linked to Young's modulus, for instance). Because of the diversity of interactions between atoms in a solid, correctly translating this microscopic view to macroscopic laws is far from trivial. However, many simplified (and less simplified) models exist, which are called *constitutive laws*: these are the physical relations you plug into your computation, and you just assume that the real material does behave in this way.

Vibrational modes are characterized by a frequency and an associated displacement. The displacement (usually denoted $\vec{u}$) is a vector field quantifying the maximum displacement at a point, for each point in the solid. For instance, the displacement field for a well-tuned guitar string would be a superposition of sine waves of the note's harmonics, perpendicular to the axis of the string[3].

Now, if the vibrating system also rotates, additional *fictitious* forces will act on it, among which the Coriolis force. As a reminder, the Coriolis force has the following form:

$$\overrightarrow{F_{cor}} = -2\rho\vec{\Omega} \times \frac{\partial \vec{u}}{\partial t}, \tag{4}$$

where $\vec{\Omega}$ is the angular velocity of the rotating system, ρ is the system's distribution of mass and $\vec{u}$ is its displacement field. For example, a mass oscillating at the end of a flexible arm in the xz plane, while the system rotates along the z axis, will tend to shift the oscillating mass out of the oscillation plane. A similar effect is exploited in the vibration gyroscope, as well as some species of insects to stabilize their flight.[4]

When exciting a specific mode, rotation will induce a Coriolis force in each point of the object, depending on its location and motion relative to the axis of rotation. This will

[2] https://en.wikipedia.org/wiki/Vibrating_structure_gyroscope

[3] At least, in the linear approximation; other frequencies can appear otherwise.

[4] https://en.wikipedia.org/wiki/Halteres

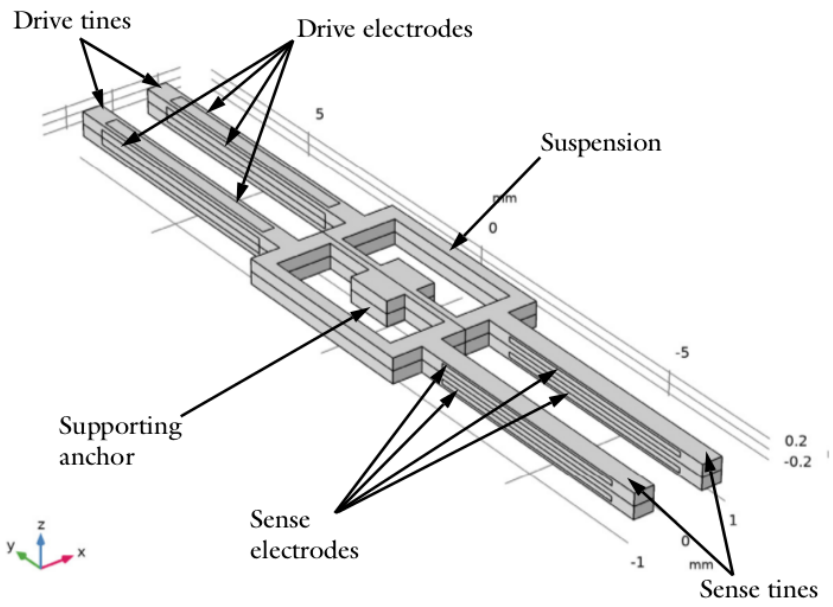

Figure 1: The MEMS gyroscope we will study.

alter the motion of the mode, and assuming that no additional energy is provided to the system, this will decrease the amplitude along the originally excited mode. In general, some modes will be attenuated while others will be amplified. [5]

1.4 Studied system

The vibrating gyroscope we will study is rather simple, as displayed in figure 1. As you can see, it looks much like a tuning fork, which makes sense since we are also interested in a specific vibration frequency.

All along this practical work, you will gradually refine your understanding of the system until you can finally explain how it works. Figure 1 should already give you some clues, and you will hopefully be able to elucidate what the different parts and terms mean.

> **Question 1:** Just looking at figure 1, and with the previous sections, take a guess as to the general idea of how this system is going to work. It is not important that you guess correctly, since you will verify later with COMSOL; just try to figure out something which makes sense to you with the information to your disposal right now.

2 Computing the eigenmodes

It should be obvious that vibrational modes are going to play a critical role in this system. Naturally, the first step is therefore to compute these eigenmodes. As in the first practical

[5] Don't feel overwhelmed if it feels overwhelming: these eigenthingies tend to make sense only after a while. For the record, it took us two full days of fiddling and discussing to understand the computation well enough.

work, you will open an already existing computation file instead of building it completely from cratch. Go to the **Application Library** and open the **Piezoelectric Rate Gyroscope** application. Since you now know COMSOL's principles and interface fairly well, you should skim the various tabs and look at the geometry for a while to familiarize yourself with the subject.

2.1 Geometry

Question 2: The geometry of this system is more complicated than before. By looking at the steps in the geometry tab, try to explain how it was built. In particular, you should explain how the *extrude* and *mirror* operations work (You can use COMSOL's help if needed).

2.2 Boundary conditions

Question 3: Explain the **Free 1** and **Fixed Constraint 1** boundary conditions. Do you think they are important (*i.e.* would they change the vibrational modes)?

Question 4: What is the gyroscope's angular velocity (direction and speed)?

The boundary conditions for the electrostatics module are similar to the previous practical work, except for the addition of boundaries concerning the drive and sense terminals.

Question 5: Explain the role of the drive terminals; what kind of voltage will be applied to them? (*Hint: does a simple mass-spring system oscillate spontaneously, or do you have to excite it using specific parameters?*)

Question 6: Explain the role of the sense terminals. Does it make sense now that we are using a piezoelectric material?

2.3 Mesh

Question 7: How are the maximum and minimum element sizes defined? How is it useful to parameterize the mesh in this way?

Question 8: Why do we need to specify a minimum element size?

2.4 Computation and resulting eigenmodes

Question 9: Briefly explain the parameters in the **Eigenmodes** study tab. In particular, pay attention to the parameters related to the eigenproblem: do we calculate all the eigenvalues or just a subset of them? In the second case, how would we select the frequencies of interest? How?

Question 10: Launch the computation and generate a report. How many degrees of freedom are there in this computation (with the default mesh)? Did you change your mind compared to the previous question: do we compute all the eigenvalues? Why?

The results tab is already populated with plots and data outputs since we loaded an already made application. Some of these outputs are not available yet, since we only computed the eigenmodes (such as the frequency response and the sensitivity).

Question 11: Find the eigenfrequencies table, and output it to a well-named text file. Did the solver find as many eigenvalues as requested?

Question 12: Look now at the graphical plot for the eigenmodes, called **Mode Shape**. Select the mode at frequency close to 8396.2 Hz, and interpret the 3D plot. It is useful to enable/disable the subplots to know which title goes with which image. Briefly describe the general behavior of the fork at this frequency (for instance, left tine goes up, right tine goes down, and so on).

Question 13: You are looking at a displacement field corresponding to a given eigenvector and frequency. What does this displacement field correspond to? What is the actual motion of the fork in the time-domain?

3 Frequency response

Eigenvalues identify precisely the *preferred* frequency at which a system vibrates, with the associated displacement. However, when forcing oscillation onto a system, nothing prevents us from applying any frequency. Of course, in general the most efficient transfer of energy happens when we are close to an eigenvalue, which is why it is called *reasonance*. For a complex system like this, we can use COMSOL's capabilities again to compute the frequency response around the frequency of interest, as shown in this section.

Question 14: Find the **Frequency Response** study tab. How are we going to compute the frequency response? What are the main parameters of the computation?

Question 15: We only computed eigenmodes, so we don't know displacements at frequencies other than eigenfrequencies. For this study, does COMSOL compute new eigenmodes, or perturbs those we computed before? (*Hint: you should look at COMSOL's documentation for* ***Asymptotic frequency sweep****)*

Question 16: Go to the results tab, and find the graph for the frequency response. Does it remind you of something (like for instance the frequency response of a simple mass-spring oscillator)? Where is the maximum located, and does it make sense?

As a bonus, you can also look at the displacement fields as a function of frequency in the **Frequency Response: Displacement** result tab. You can even create an animation looping over the frequency range!

4 Sensitivity

The final step is to find the relation between the angular velocity (around the chosen axis) and the sense voltage. The coupling between the vibrational modes and the Coriolis force effects this relation, but we still have to measure it in order to calibrate our gyroscope.

Question 17: Find the **Sensitivity** study tab. Explain what computation is going to be carried by looking at the parameters. Launch it, and generate report once it is done.

Question 18: Find the resulting curve. What do you get? Could we expect anything simpler or better?

Question 19: How will this curve be used in a pratical setting? That is, in the end, how to we get the angular velocity once the sense voltage has been acquired?

5 Influence of the tines' length and the mesh quality

5.1 Using a finer and a coarser mesh

Ideally, you should always study the *convergence* of your results depending at least on the main parameters of the problem, especially the coarsity of the mesh. The idea is to decrease the maximum element size and recompute the solution until some representative property becomes nearly constant. Commonly used properties are the total energy of the system or the estimated error, or in this case, the value of the eigenfrequencies.

Instead of a full convergence analysis, in this section you will just be asked to recompute the eigenmodes for a coarser and a finer mesh.

Question 20: In the meshing parameters, decrease the maximum element size to `tQz/6`. Recompute the eigenmodes; how much time did it take? How many degrees of freedom were there?

Question 21: Export the eigenfrequencies to a text file (careful, you must press the **Evaluate** button otherwise you will see the previously computed eigenfrequencies). Compare them to the previously obtained frequencies. Can you still identify each frequency from each computation? (i.e. are they close enough?)

Question 22: Similarly, increase it now to `tQz`, recompute and note the time it takes and the number of degrees of freedom. Export the eigenfrequencies in another well-named text file, and compare again with the eigenvalues using the default mesh. Comment on the tradeoff between computation time and accuracy.

5.2 Changing the length of the tines

Question 23: Using the default mesh, increase the length of the drive tines to 10 mm (by modifying the corresponding global parameter). Compute the solution, and interpret the results. Are the new frequencies consistent with this change in geometry? (*Hint: a violin is higher-pitched than a cello, which is in turn higher-pitched that a bass.*) Are the old modes recognizable, or completely different?

6 Conclusion

Question 24: Finally, summarize your understanding of this vibrational gyroscope. What is the main idea used in its operation? How does it compare to your original guess in the first question?

This third pratical work completes your introduction to COMSOL. It showed you some of the basic features of a finite element software package, especially its repetitive workflow. Of course, many more features exist and capabilities go far beyond what we have seen; but this is fertile ground for future practicals!

A The terminal and some useful commands

It is very common for computing environments to involve the use of a *terminal*, which is a textual but direct interface to your machine. The terminal expects you to type in commands by displaying a *prompt*, commonly represented by a dollar sign:

```
$
```

You should not type in the `$` symbol when you want to execute a command. Also, be aware that the prompt can be different on different systems, so do not expect to see a `$` all the time.

There is a large variety of commands available, depending on the system. Some commands are built-in, while some correspond to programs installed by the user (like COMSOL). The following sections describe a few of the most important commands, mostly built-in.

A.1 `pwd`: *program working directory* and filesystem paths

The *current directory* (or *program working directory*, PWD) is your current location in your file system. By default, when loggin into a system or opening a terminal, you will usually end up in your *home* directory. Each user has its home directory, which can only be accessed by its owner.

Using this command is simple, since it does not take any argument:

```
$ pwd
```

It will return you the path of the current directory, such as:

```
-> /home/nperrea
```

Here again, the arrow symbol `->` just indicates this is something output by the terminal. It is just a convention, since in practice most terminals don't print anything at the beginning of an output line.

The path which was returned by `pwd` is actually an *absolute path*, which describes the path of a file or directory from the root of the filesystem. There are also *relative paths*, which reference a file or directory from the current directory. Absolute paths always start with a `/` symbol, while relative paths never do. Therefore, if the `/home/nperrea` folder contains a file named `hello_world.c`, it can be referenced by the following absolute path:

```
/home/nperrea/hello_world.c
```

Or, if the current directory (output by the `pwd` command) is `/home/nperrea`, simply by:

```
hello_world.c
```

A.2 `cd`: *change directory*

The `cd` command allows you to change the current directory, by providing a path as argument:

```
$ cd new_directory_path
```

where `new_directory_path` can be absolute or relative. The following example changes the current directory from `/home/nperrea` to the `comsol_praticals` subdirectory:

```
$ pwd
-> /home/nperrea
$ cd comsol_practicals
$ pwd
-> /home/nperreau/comsol_practicals
```

Notice that `cd` doesn't return any output.

In addition to absolute and relative paths, there are two special paths which are extremely useful. A single dot (`'.'`) refers to the current directory while two dots (`'..'`) refers to the parent directory. Referring to the current directory is sometimes useful, for example when executing programs located in the current directory:

```
$ ./some_program
```

Referring to the parent directory allows you to revert the effects of a previous `cd` command, which otherwise would only allow you going down the filesystem hierarchy. Thus, applied to the previous example, we would have:

```
$ pwd
-> /home/nperrea/comsol_practicals
$ cd ..
$ pwd
-> /home/nperreau
```

A.3 `ls`: listing the contents of a directory

Of course, you do not need to remember the whole file hierarchy of your computer in order to use `cd` and other commands. Without any argument, the `ls` command lists the files and directories contained in the current folder.

It is however more useful when using some options, also called *switches*. By convention, an option to a command can be specified by starting it with a hyphen (`'-'`) or two (`'--'`). Being a convention, there is some variation among systems, but usually a single hyphen introduces so-called *short-options*, which are specified with a single letter, and two hyphens introduce long options which usually describe the purpose of the option.

For instance, `ls` has a short `'l'` option displaying the output as a list instead of a table:

```
$ ls -l
```

It also has long options. The `help` long is commonly implemented in many commands, and is used to print a documentation of what the command can do, among which all the options it can take:

```
$ ls --help
```

It is a matter of taste, but the most useful set of options for `ls` seem to be:

- '`l`', to print a list instead of a table ;
- '`h`', to print files sizes in easily readable units instead of the size in bytes ;
- '`a`', to also print the hidden folders and files.

Short options can be combined together in a single switch; that is, the following two commands display the same output:

```
$ ls -l -h -a
$ ls -lha
```

Since the '`lha`' switches are so commonly used, many systems have an alias command named `ll` for `ls -lha`.

A.4 `cp`: copy files and directories

Another very handy command is `cp`, allowing you to copy files from one location to another. It is used in the following way:

```
$ cp source_path destination_path
```

where you should substitute `source_path` and `destination_path` with the desired locations, either as absolute or relative paths. Notice a command convention amond terminal commands, that source operands precede destination operands.

You can also copy several files at once, in which case the destination is necessarily a directory:

```
$ cp source_A source_B source_C destination_path
```

When you want to copy folders, you have to use the `-r` flag (as in *recursive*):

```
$ cp -r source_directory destination_directory
```

This will copy all the files and subfiles contained in `source_directory` to `destination_directory`.

A.5 `mv`: move files and directories

To move files and directories, you can use the `mv` command, which works lie `cp`, except the source files will not remain afterwards. It is easy to inadvertently erase files without knowing using `mv`, so it is always a good idea to use it with the `v` (verbose) and `n` (don't overwrite) switches:

```
$ mv -vn source_path destination_path
```

A.6 `mkdir`: creating directories

You should of course organise your files in a clever hierarchy. In addition, commands like `cp` will not create directories if they do not exist. To create a directory, you can use the `mkdir` command:

```
$ mkdir new_directory
```

A.7 `man`: print documentation about a command

Many commands have the `help` option (or similarly, a '`h`' short option) to print some documentation when in doubt. More extensive documentation can be found in the venerable *man pages*, short for *manual pages*, through the `man`[6] command. To display documentation about the `ls` command, simply type:

```
$ man ls
```

This will launch a text interface in which you can navigate using the arrow keys, and which you exit by typing '`q`'. It is always a good idea to check the man pages when you are introduced to a new command. As a suggestion, it would be wise for this practical work to see what it can teach you about `ssh` and `scp`:

```
$ man ssh
$ man scp
```

A.8 Command history and autocompletion

Two nice features implemented in most terminals is the *command history* and *autocompletion*. The first allows you to recall previously-typed commands using the up and down arrow keys. It is especially useful when typing long commands involving long paths or addresses, such as `ssh` and `scp` commands.

Autocompletion is useful for lazy people (that is, essentially every single programmer), since it can complete command names and even paths while typing commands. It is used through the tab key. If you press it once, it can either complete the current part of the command you are typing if it can choose unambiguously, or display nothing. In the

[6] No sexism intended.

second case, you can hit tab twice in a row, and this will list all the possibilities it found. This allows you to complete the command a bit until it can choose among the possibilities unambiguously, at which point you can press tab again to let the terminal complete it for you.

It is much more intuitive than it sounds.

B Working with remote computers: SSH and X forwarding

The COMSOL program will be launched on a remote computing server, which is more powerful than the local computers. The terminal introduced in the previous section executes commands locally, that is, on the computer on which they are typed in. When working remotely, you need a way to execute commands *as if* you were typing them on that remote computer.

B.1 The *Secure Shell* (SSH)

This is exactly the purpose of the *Secure Shell* (SSH): it provides you with a terminal executing on a remote machine, in addition to encrypting the channel. Apart from this, the terminal is similar to any terminal. However, you should always remember that SSH executes on the remote computer, and therefore provides you with the environment of the remote system. For instance, a command might be available on you local computer but not of the remote computer, and vice versa.

To use SSH, you need to provide the name of an existing user on the remote machine as well the IP address of the machine. If we use the fake user `my_user` and the IP address of the Compuphys server, the command is simply:

```
$ ssh my_user@172.20.134.7
```

You will then be asked to type your password, and if successful, the remote shell will be launched. You can now use this terminal as if it was a regular terminal.

B.2 `scp`: secure copy

At times you will be asked to let COMSOL generate some files for you. Of course, because it executes remotely, it will store them remotely as well, and you need to copy them back to your local machine. This can be done with the `scp` command, which is a mix between `cp` and `ssh`, and has to be executed on a terminal on you *local* machine:

```
$ scp user@172.20.134.7:/remote/path /local/path
```

which will prompt you for your password before starting the transfer. In this command, `/remote/path` is the path to a file or directory on the *remote* server, from which we want to download, and `/local/path` is the path on the *local* machine where we want to store the file or directory.

When downloading a single file, this command can be used directly, but you may want to download a full directory with all its subdirectories, or a set of files all at once. In the first case, you can use the `-r` switch (`r` for *recursive*):

```
$ scp -r user@172.20.134.7:/some/directory /local/directory
```

while in the second case, you can specify a list of files enclosed with *escaped* curly braces. This means you type `\{` instead of just `{`, because otherwise the terminal will interpret the curly braces differently. So, to transfer several files at once, the command will typically look like:

```
$ scp user@172.20.134.7:\{file1,file2,file3\} /local/directory
```

You can also use the `-r` switch in this command, so that you can also download several directory hierarchies at once. A useful command is `readlink`, which allows you to display the absolute path of a remote file on the server:

```
$ readlink -f ./some_file
```

You can copy-paste the resulting path and substitute it for `remote_path`; this way, you won't have to care about relative paths and working directories.

B.3 Graphical forwarding

SSH only allows you to access a remote terminal, i.e. a text-based interface. On computers using the *X11* graphics protocol such as most UNIX-based systems, it is possible to use *X11 forwarding*: graphical commands from the program are redirected to our local graphics system instead of that of the server. This type of SSH channel is established with the following command (to be typed in a terminal):

```
$ ssh -X user@172.20.134.7
```

where `user` is your user name for the Compuphys server, and `172.20.134.7` is the server's IP address. Then, you can simply launch COMSOL:

```
$ comsol
```

and this should open a window on your local computer.