Part I

Tutorials

Chapter 1

Getting started

This chapter describes how to work with virtualized environments that contain pre-configured simulation packages. There are several such packages available. Each contains an implementation of the Medical Simulation Markup Language (MSML) as well as different simulation backends. In the following we describe how to run these environments based on Docker containers[1]. We also describe how to connect to server instances that run dockerized MSML environments.

1.1 Running MSML-based Docker containers

In this section, we describe how to run a pre-configured deployment of the Simulation Open Framework Architecture (SOFA) simulation package.

1.1.1 Installing the container

1. Install the Docker framework for your Linux distribution. Further details can be found on the Docker homepage. For Ubuntu Linux there is the possibility to use either pre-configured packages (see `www.ubuntuupdates.org/ppa/docker` or to install Docker manually (`http://docs.docker.com/installation/ubuntulinux/`).

2. Optional: You can add your user to the docker group in order to avoid the use ofsudo in front of the docker commands:

```
$ sudo –aG docker username
```

[1] `www.docker.com`

Please remember to log out in order to for the changes to take effect.

3. Pull the SOFA container from the Docker Hub:

   ```
   $ docker pull ssuwelack/msml_sofa
   ```

1.1.2 Running the container

There are two different options to run a graphical application inside the simulation container:

1. Start an SSH server inside the container and connect to it. This typically is a very stable set-up, but introduces an additional overhead.
2. Link the host X-server into the container. This approach can achieve bare metal performance, if the native graphics drivers are part of the container. However, it does not work on all host systems and has problems with Qt-based apps.

In the following we describe how to start the container in each mode. The start-up procedures described below are available in script form from Github:

```
$ git clone https://github.com/ssuwelack/msml-docker
  -runtime.git
```

Communicating to the container over SSH (recommended)

1. The following command starts the Docker container as a daemon, assigns it the name msml_sofa and binds it's SSH port to port 22000 of the localhost:

   ```
   $ docker run -d -p 127.0.0.1:22000:22 --name
       msml_sofa ssuwelack/msml_sofa /root/start_ssh
       .sh
   ```

2. Alternatively, the startup script can be used:

   ```
   $ ./start_ssh_msml_sofa.sh
   ```

3. We can use the pre-configured user msml (password: msml) to connect to the running container using SSH:

```
$ ssh -XC msml@localhost -p 22000
```

4. The sofa runtime is located at /opt/sofa/bin. All data is stored in the home directory of the msml user.

5. The container can be stopped by executing

   ```
   $ docker stop msml_sofa
   ```

   on the host machine. It can be re-started using the command

   ```
   $ docker start msml_sofa
   ```

   Stopped containers can be deleted through

   ```
   $ docker rm msml_sofa
   ```

Communicating to the container via X-server sockets

In order to run the container with minimally overhead, the X-server socket can be forwarded. A start-up script is available in the above mentioned repository that carries out the necessary steps:

```
$ ./run_msml_sofa.sh
```

1.2 Connecting to a server instance

In order to avoid the work of setting up individual environments, the containers can be run on a server instance. In order to connect to such a container, you need the following information:

- Server name
- Public port number of the container
- Jump host name (if server is protected by a firewall)
- If not changed by the container provider, login credentials are:

  User: msml

  Password: msml

1.2.1 Connecting from a Linux client

In order to connect from a linux clients, just run:

```
$ ssh –XC msml@servername –p portnumber
```

1.2.2 Connecting from a Windows client

The MobaXterm framework can be used to connect from a windows machine. In this section, we describe how to run SOFA on a server instance. The server i61sv002.ira.uka.de exposes 25 docker container that run SOFA from port 22001 to 22025. In order to access these containers from a Linux machine inside the the HIS network, simply run

```
$ ssh –XC msml@i61sv002.ira.uka.de –p 22010
```

In the following we describe how the containers can be accessed from any Windows machine.

1. Install the MobaXterm framework which you can download on its homepage (http://mobaxterm.mobatek.net/download-home-edition.html).
2. Open MobaXterm and start a new session (cf. figure 1.1).
3. Configure your SSH session (cf. figure 1.2):
   - server name
   - user / password (standard is msml/msml)
   - port number
   - jump host for port forwarding and login credentials for jump host

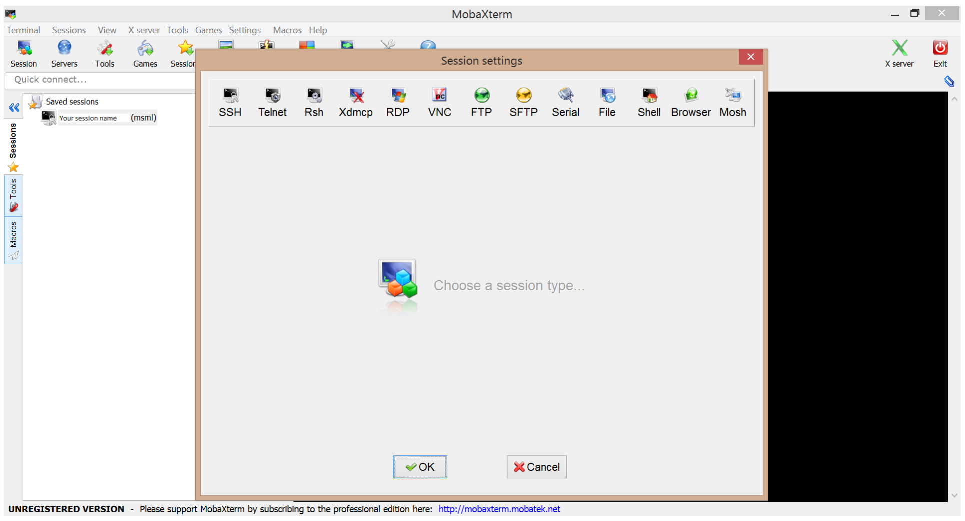
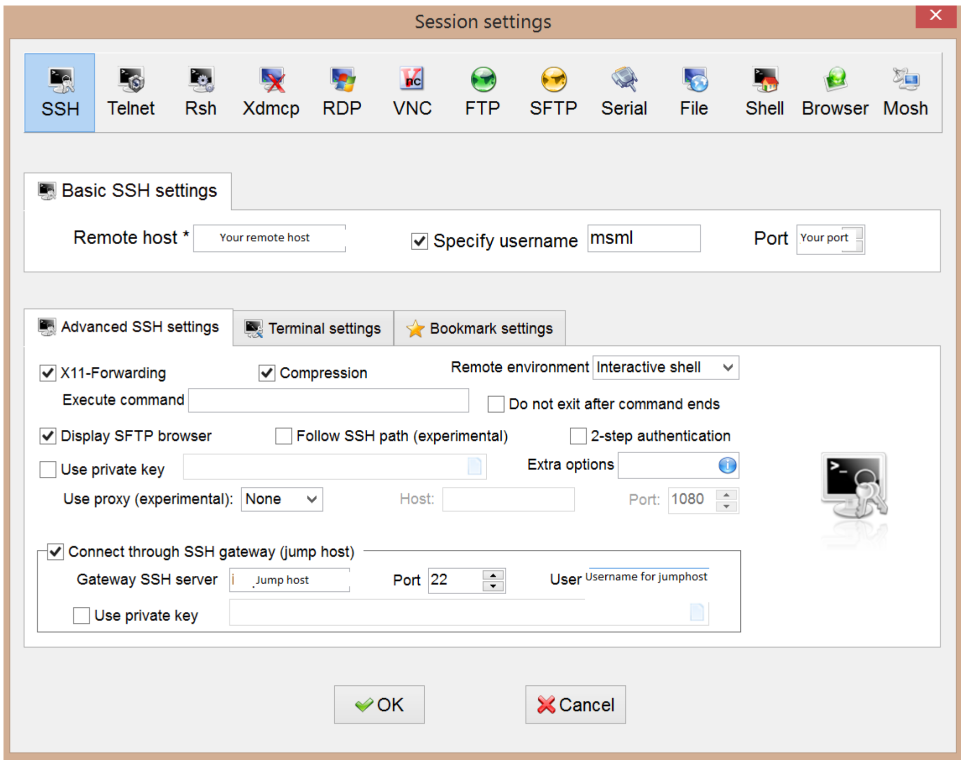

MobaXterm
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions
X server Exit
Quick connect...
Saved sessions
Your session name (msml)
Sessions Tools Macros
Session settings
SSH Telnet Rsh Xdmcp RDP VNC FTP SFTP Serial File Shell Browser Mosh
Choose a session type...
OK Cancel
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net

Figure 1.1: Screenshot of MobaXterm after starting a new session.



Remote host * Your remote host
[✓] Specify username msml
Port Your port

[✓] X11-Forwarding
[✓] Compression
Remote environment Interactive shell
Execute command
[ ] Do not exit after command ends
[✓] Display SFTP browser
[ ] Follow SSH path (experimental)
[ ] 2-step authentication
[ ] Use private key
Extra options
Use proxy (experimental): None
Host:
Port: 1080
[✓] Connect through SSH gateway (jump host)
Gateway SSH server Jump host
Port 22
User Username for jumphost
[ ] Use private key

Figure 1.2: Screenshot of MobaXterm after starting a new session.

Chapter 2

Tutorial 1: Phenomenological Modeling

2.1 Pendulum with implicit Euler time integration

Start the SOFA scene file *Spring_EulerImplicit.scn* that describes a simple mass-spring system which is discretized using an implicit Euler scheme.

If you are using the pre-built Docker container this can be achieved by executing

```
$ cd /opt/sofa/bin
```

```
$ ./runSofa /home/msml/tutorials/Tutorial1/
    Spring_EulerImplicit.scn
```

In order to see the spring-mass model as shown Fig. 2.1, please click *All* in the tab *View*. To animate a scene click on *animate*. For moving a particle press *shift* and click on the specified particle.

Parametric studies can be used to get a better understanding of numerical methods. In order to analyze the implicit Euler technique, change the following input parameters

- time step size dt in s [0.001, 0.01, 0.1, 1]
- stiffness K_s and damping coefficient K_d of the spring (spring='Id1, Id2, K_s, K_d, L')

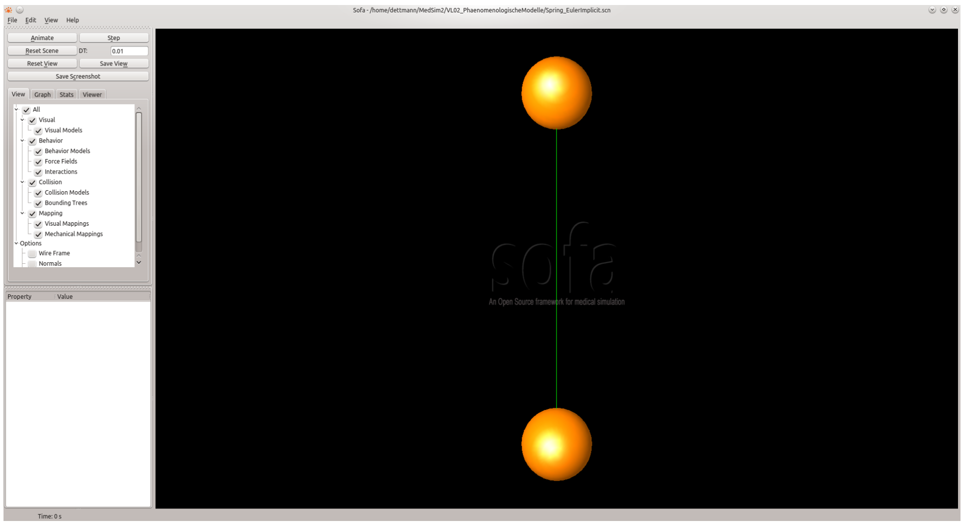


Figure 2.1: SOFA framework running a simple pendulum simulation.

The parameters can be changed by opening the SOFA scene file (.scn) in a suitable text editor, e.g.

```
$ geany /home/msml/tutorials/Tutorial1/
    Spring_EulerImplicit.scn &
```

While changing the parameters observe the behavior of the simulation. In particular, try to answer the following questions

- How does the parameter effect the stability of the simulation?
- Does the frequency or amplitude change?
- How does the energy of the system behave over time?

2.2 Pendulum with explicit Euler time integration

The pendulum can also be discretized with an explicit Euler time integration scheme (*Spring_EulerImplicit.scn*). Repeat your analysis on this set-up. In particular, compare the behavior of the explicit and the implicit scheme.