Class on numerical mechanics:

From Lagrangian mechanics to simulation tools for computer graphics

Practicals

Florence Bertails-Descoubes [1], Mélina Skouras [2], Mickaël Ly [3]

2019, September 12 - ENS Lyon

[1] florence.descoubes@inria.fr
[2] melina.skouras@inria.fr
[3] mickael.ly@inria.fr

General instructions

- One report per practical
- Report to send at `mickael.ly@inria.fr`
- Deadline: the Monday following the practical at 9pm
- Can be done invididually or by pairs
- General comments to the whole class at the beginning of the next session
- Final grades after the end of the course

Instructions for the reports

- The report should be clear and concise (avoid a too messy/too verbose report)
- The evaluation will be on your insights, your argumentation, and your conclusions (think of a scientific paper)
- The code is not evaluated (neither as is, nor as excerpts in the report)
- Feel free to ask questions during the course, or by email (and not at the last moment !)
- First time for the practicals, so please give your feedbacks ! (Too easy/too hard/too long ?)

Framework

- You may have already installed the proposed code base (`https://github.com/Astcort/PyGLViewer`), but you can use your own

Framework

- You may have already installed the proposed code base (`https://github.com/Astcort/PyGLViewer`), but you can use your own
- In `Python3` for the prototyping of the numerical methods + `OpenGL` for the visualization

Framework

- You may have already installed the proposed code base (`https://github.com/Astcort/PyGLViewer`), but you can use your own
- In `Python3` for the prototyping of the numerical methods + `OpenGL` for the visualization
- There is the base code to work on rods and meshes, and to render them, but feel free to modify or to add your own stuff

Framework

The structure for a test case should enable you to easily add your own (see the files `main.py` and `scenes.py`).
For instance, to animate a mesh:

```
# Create the object
myMesh = Mesh2D(positions, colours)
```

Framework

The structure for a test case should enable you to easily add your own (see the files `main.py` and `scenes.py`).
For instance, to animate a mesh:

```
# Create the object
myMesh = Mesh2D(positions, colours)

# Create the dynamic system
myDn = MyDynamicSystem(myMesh) # See the class DummyDynamicSystem e.g.
```

Framework

The structure for a test case should enable you to easily add your own (see the files `main.py` and `scenes.py`).
For instance, to animate a mesh:

```
# Create the object
myMesh = Mesh2D(positions, colours)

# Create the dynamic system
myDn = MyDynamicSystem(myMesh) # See the class DummyDynamicSystem e.g.

# Add it to the viewer
# At each frame, the viewer will call its method 'step'
viewer.addDynamicSystem(myDn)
```

Framework

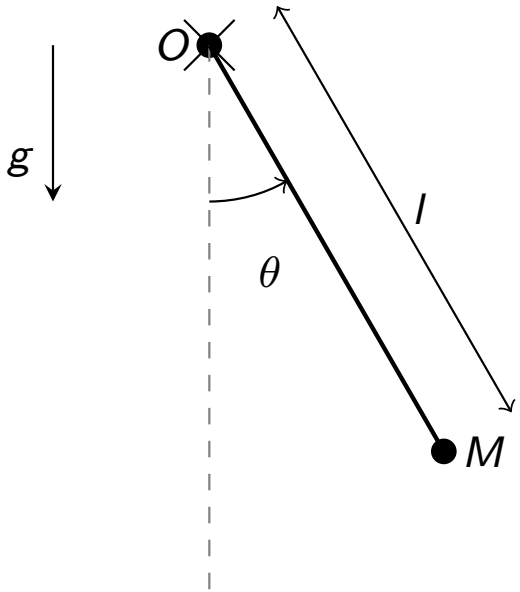
The structure for a test case should enable you to easily add your own (see the files `main.py` and `scenes.py`).
For instance, to animate a mesh:

```
# Create the object
myMesh = Mesh2D(positions, colours)

# Create the dynamic system
myDn = MyDynamicSystem(myMesh) # See the class DummyDynamicSystem e.g.

# Add it to the viewer
# At each frame, the viewer will call its method 'step'
viewer.addDynamicSystem(myDn)

# Add the rendered object
myMeshRenderable = Mesh2DRenderable(myMesh)
viewer.addRenderable(myMeshRenderable)
```

then run `python3 main.py`.

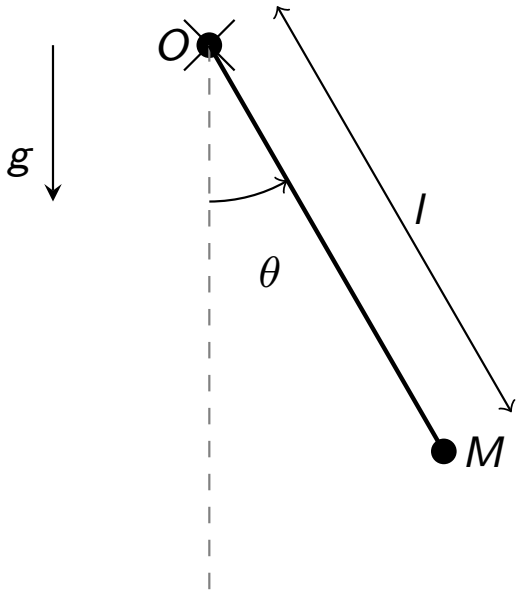
Practical 1: Euler implicit and explicit schemes

Exercise 1: Simple pendulum



During the class, you derived the equation for a simple pendulum

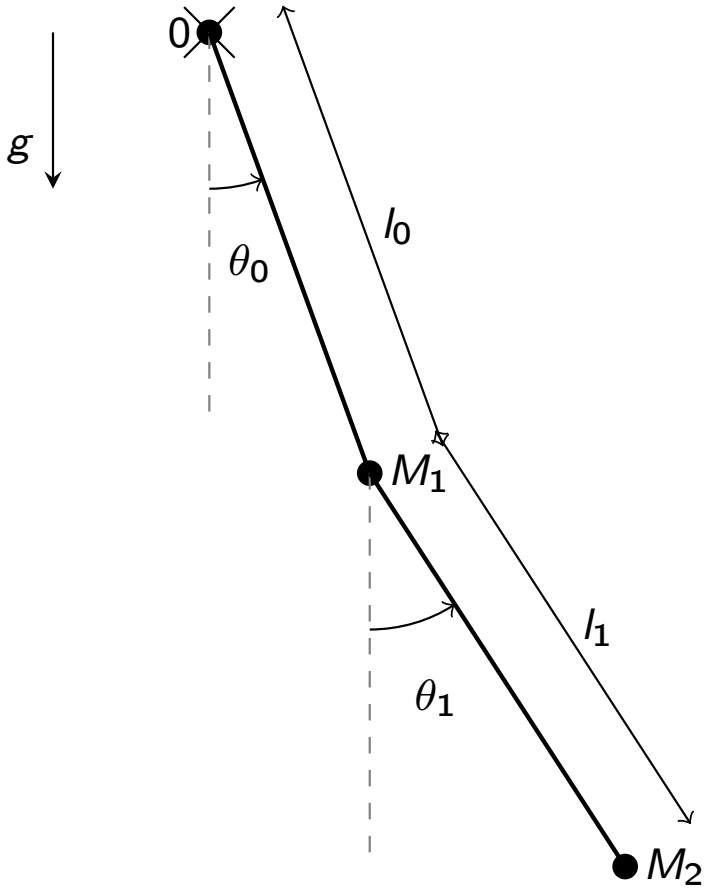
Exercise 1: Simple pendulum



During the class, you derived the equation for a simple pendulum

Now it is time to see it move ! To this end, you will compare the two basic numerical integration schemes (the *Euler explicit* and *Euler implicit* schemes), and also the Euler semi-implicit scheme.

Exercise 2: 2-arms pendulum



During the class, you also derived (or at least tried to derive) the motion equation for the double pendulum.

We now want to see its dynamical behaviour (with which numerical scheme ?).

Let's start !

(The detailed practical instructions are also on the Github
`https://github.com/Astcort/PyGLViewer`)