Class on numerical mechanics:

From Lagrangian mechanics to simulation tools for computer graphics

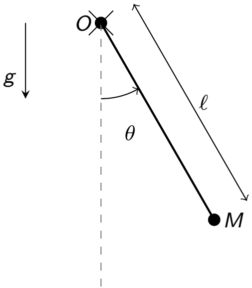
Practicals

Florence Bertails-Descoubes [1], Thibaut Métivet [2], Jean Jouve [3]

2023, October 3 - Ensimag

[1] florence.descoubes@inria.fr
[2] thibaut.metivet@inria.fr
[3] jean.jouve@inria.fr

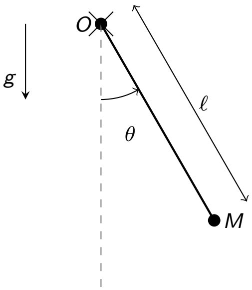
Back to the simple pendulum



Computing the dynamics

$$\ddot{\theta} + \frac{g}{\ell} \sin \theta = 0 \qquad \text{with } \theta(0) = \theta_0 \text{ and } \dot{\theta}(0) = \lambda_0$$

Back to the simple pendulum

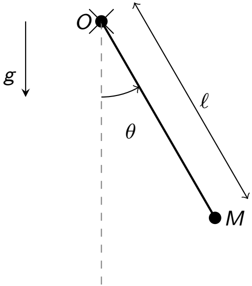


Computing the dynamics

$$\ddot{\theta} + \frac{g}{\ell}\sin\theta = 0 \qquad \text{with } \theta(0) = \theta_0 \text{ and } \dot{\theta}(0) = \lambda_0$$

- Nonlinear equation, no explicit solution

Back to the simple pendulum



Computing the dynamics

$$\ddot{\theta} + \frac{g}{\ell}\sin\theta = 0 \qquad \text{with } \theta(0) = \theta_0 \text{ and } \dot{\theta}(0) = \lambda_0$$

- Nonlinear equation, no explicit solution
- $\rightarrow$ Recourse to numerical integration

Framework

- You may have already installed the proposed code base (`https://gitlab.inria.fr/elan-public-code/pyglviewer`), but you can use your own

Framework

- You may have already installed the proposed code base (`https://gitlab.inria.fr/elan-public-code/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://gitlab.inria.fr/elan-public-code/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

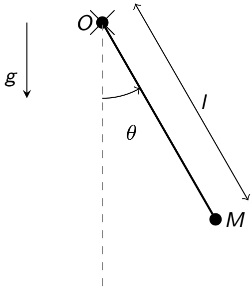
The structure for a test case should enable you to easily add your own (see the files `main.py` and `scenes.py`).
For instance, to animate a mesh:

```
# 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 - part 1:
Finite differences (explicit Euler scheme)

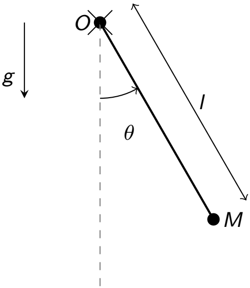
Exercise 1: Simple pendulum



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

Now it is time to see it move !

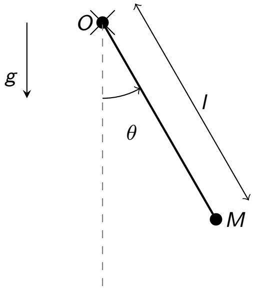
Exercise 1: Simple pendulum



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

Now it is time to see it move !
To simulate the pendulum, you will *discretize in time* the equation of motion using the simple explicit Euler scheme.

Exercise 1: Simple pendulum



Goals of this practical:

- Get familiar with the code;
- Have a quickly working simulator, to serve as a basis for your future project;
- Study and analyse various integration schemes (if enough time).

More: analysis

More

Explicit vs. Implicit Euler

Exercise 1

Consider the linearized pendulum problem (valid for small angle θ),

$$\ddot{\theta} + \frac{g}{\ell}\theta = 0 \qquad \text{with } \theta(0) = \theta_0 \text{ and } \dot{\theta}(0) = \lambda_0,$$

and express the condition on the time step h for `Explicit Euler` to be stable.

Explicit vs. Implicit Euler

Exercise 2

Same question for `Implicit Euler`.