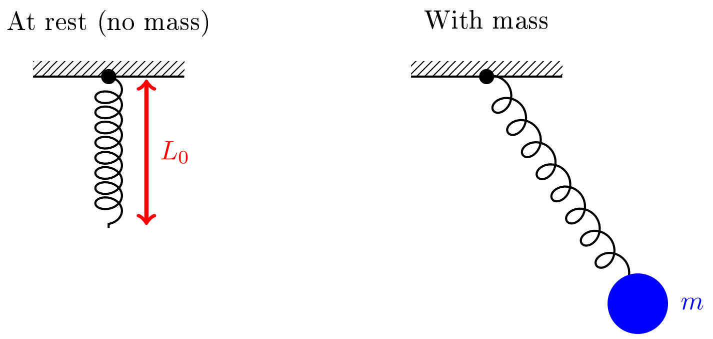
Project 2 in Numerical Analysis

Introduction

An elastic pendulum (also called spring pendulum) is a simple physical system where a mass m attached to a massless spring with rest length L_0 and spring constant k is allowed to oscillate freely, see below.

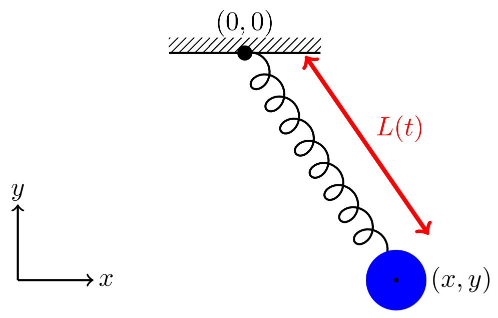


Many real-world systems can be modelled as adequate modifications of an idealized elastic pendulum. Examples include wind-induced vibrations of high towers, models of human and bipedal robot walking as well as large scale weather phenomena. The elastic pendulum is also fascinating in its own right: it is a very simple system with only two degrees of freedom, however it displays typical behaviour of complex non-linear systems, namely unpredictability, resonance and extreme sensitivity to initial conditions. In other words: chaos.

In order to derive the equations of motion we must agree on a coordinate system. We assume the spring is attached at $(0,0)$ while the mass is centered at $(x,y) = (x(t), y(t))$. The variable length of the pendulum is therefore

$$L(t) = \sqrt{(x(t))^2 + (y(t))^2}$$

which may be larger or smaller than L_0 depending on the state of the oscillation.



The force exerted by the spring on the mass can be computed according to Hooke's law:

$$\boldsymbol{F} = -k\Delta L \boldsymbol{n}$$

where $\Delta L = L(t) - L_0$ is the amount by which the spring is stretched, and $\boldsymbol{n}$ is the unit vector through $(0,0)$ and (x,y). In other words

$$\boldsymbol{n} = \frac{1}{L(t)}\begin{pmatrix} x \\ y \end{pmatrix}$$

so that the components of the spring force $\boldsymbol{F}$ along the x and y axes are

$$\boldsymbol{F}_x = -k\big(L(t) - L_0\big)\frac{x}{L(t)} \qquad \boldsymbol{F}_y = -k\big(L(t) - L_0\big)\frac{y}{L(t)}$$

We now apply Newton's law, including gravity in the y direction:

$$\boldsymbol{F} + \boldsymbol{F}_{\text{gravity}} = m\boldsymbol{a}$$

and obtain the second-order differential system

$$\left\{\begin{array}{rcl} \dfrac{d^2x}{dt^2} & = & -\dfrac{k}{m}\big(L(t) - L_0\big)\dfrac{x(t)}{L(t)} \\[2ex] \dfrac{d^2y}{dt^2} & = & -\dfrac{k}{m}\big(L(t) - L_0\big)\dfrac{y(t)}{L(t)} - g \end{array}\right.$$

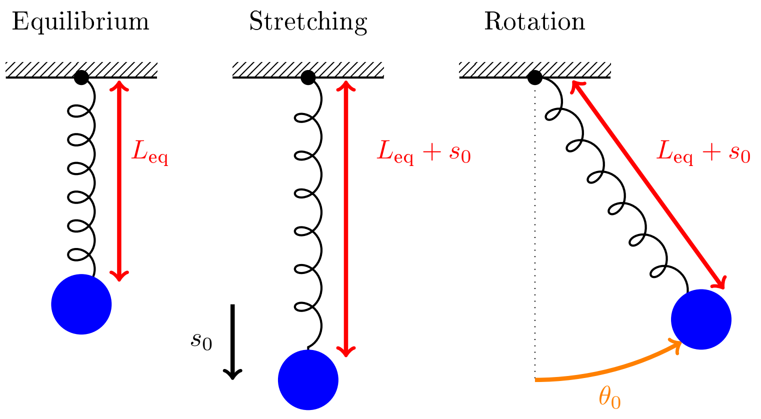
Solving this coupled non-linear system yields $x(t)$ and $y(t)$, that is, the trajectory of the mass. In order to solve the system we also need initial conditions at $t = 0$.

Initial conditions

We need four initial conditions in order to uniquely solve this system, namely values for $x(0)$, $x'(0)$, $y(0)$ and $y'(0)$. Unless it is explicitly stated we will only consider situations where we let go of the mass at $t = 0$ without pushing it, that is:

$$x'(0) = y'(0) = 0$$

A natural parametrization for the initial position $(x(0), y(0))$ is to know by how much the string is stretched at $t = 0$ from its equilibrium length L_{eq} and by which angle θ_0 it is diverted from the vertical. These two modes of action are displayed separately below.



The equilibrium length has to balance gravity and the force exerted by the spring hence

$$L_{\text{eq}} = L_0 + \frac{mg}{k}$$

where L_0 is the rest length without any mass. A bit of trigonometry then yields the initial position of the mass after stretching by s_0 and rotating by an angle θ_0:

$$\left\{\begin{array}{rcl} x(0) & = & (L_{\text{eq}} + s_0)\sin(\theta_0) \\ y(0) & = & -(L_{\text{eq}} + s_0)\cos(\theta_0) \end{array}\right.$$

Choosing s_0 and θ_0 completely determines the initial position.

Solving the system with two methods

1. Rewrite the differential system as a first-order differential system for four variables. Show your calculations and clearly indicate the order of the variables.

2. We use the parameters

$$m = 0.2\,\text{kg} \qquad L_0 = 1\,\text{m} \qquad k = 2.5\,\text{N/m} \qquad g = 9.81\,\text{m/s}^2$$

so that

$$L_{\text{eq}} = L_0 + \frac{mg}{k} = 2.7848\,\text{m}$$

Write a program solving the spring pendulum system using Euler's method. The program should at least have the following inputs:

- Initial stretching s_0 and initial rotation θ_0.
- Time interval length T (so that the equation is solved for $0 \le t \le T$).
- Total number of steps n.

The output of the program is a $(n+1) \times 4$ matrix w containing values for the four variables at each time step.

Test run. Run the program with no initial stretching $s_0 = 0$ and $\theta_0 = \frac{\pi}{12}$. Solve on the interval $[0, 10]$ with a total of $n = 300$ time steps. The final position of the pendulum at $T = 10$ should be

$$x = 0.4808 \qquad y = -1.9653$$

3. Write a program that uses the output w of **Eulersolver** in order to plot the trajectory of the pendulum in real time (**Tips regarding animation in Matlab can be found as an appendix to the project**). As a first step you can draw the string as a line segment to make things simpler. Run it using the same parameters as in question 2 together with initial conditions

$$s_0 = 0 \qquad \theta_0 = \frac{\pi}{12}$$

Explain why the displayed movement is clearly incorrect and what could explain it.

4. Replace the subroutine **eulerstep** by a new subroutine **RKstep** which uses the Runge-Kutta method instead. Run the program again and compare with the Euler run.

Explore different initial conditions as well as longer time intervals and show at least three different movement types.

Energy and error analysis

The spring pendulum is a conservative system, which means that the total energy is conserved through the movement. So far we have not considered any friction forces. The total energy $E(t)$ has three components:

- Potential energy $U = mgy$
- Kinetic energy $K = \frac{1}{2}mv^2 = \frac{1}{2}m(x^2 + y^2)$
- Spring energy $W = \frac{1}{2}k(L_0 - L)^2$ where $L = \sqrt{x^2 + y^2}$.

5. We first consider the same low-energy initial conditions as before:

$$s_0 = 0 \qquad \theta_0 = \frac{\pi}{12}$$

Plot the total energy as a function of time over the time interval $0 \le T \le 40$, as well as all three components separately. Also show a plot of the energy error

$$\text{Error} = \left|E(t) - E_{\text{initial}}\right|$$

Interpret the results.

6. Repeat for higher-energy initial conditions. You may need to play around with n to improve the accuracy of the Runge-Kutta method. Interpret your findings.

Note. Our model is far more perfectly accurate and you might get strange results for some extreme initial conditions e.g. $|\theta_0| \approx \frac{\pi}{2}$. This is caused by the spring being so squished that $(x, y) \approx (0, 0)$ which is not realistic.

7. Since energy should be theoretically conserved, this gives us a way to estimate the accuracy of our numerical method. We consider first low-energy initial conditions

$$s_0 = 0 \qquad \theta_0 = \frac{\pi}{12}$$

on the interval $[0, 40]$ i.e. $T = 40$. Starting with $n = 10 \cdot T$ and doubling n at least eight times, compute the final energy error

$$\text{Error} = \left|E(40) - E_{\text{initial}}\right|$$

and plot it as a function of n (or $h = T/n$). Your graph should convincingly show that the Runge-Kutta method is at least a fourth-degree method. Explain why.

8. Repeat for a large enough range of initial conditions, for instance a random sample of values of s_0 and θ_0 on reasonable intervals. Is the Runge-Kutta method of order 4 in all possible situations? Explain why.

Order-Chaos-Order dynamics of the spring pendulum

A fascinating aspect of the spring pendulum is that it is an orderly system at low energy values, becomes chaotic as the energy increases and orderly again at very large energies. In this context, chaos is defined as extreme sensitivity to initial conditions. If two pendulums start at almost exactly the same position, their movements will end up completely disconnected after a short amount of time. This is known as the butterfly effect. The physical parameters of the system are unchanged, namely:

$$m = 0.2\,\mathrm{kg} \qquad L_0 = 1\,\mathrm{m} \qquad k = 2.5\,\mathrm{N/m} \qquad g = 9.81\,\mathrm{m/s^2}$$

9. Consider Pendulum One with initial conditions

$$s_0 = 0.5 \qquad \theta_0 = \frac{\pi}{3}$$

and Pendulum Two with slightly perturbed initial conditions

$$s_0 = 0.5 + \varepsilon \qquad \theta_0 = \frac{\pi}{3} + \varepsilon$$

where ε is a small position difference, e.g. $\varepsilon = 10^{-3}\,\mathrm{m}$. Run both pendulums on the time interval $[0, 60]$. Do the two oscillations stay roughly together?

10. Steadily increase $y'(0)$ while keeping all other parameters equal. We imagine the spring is being hit upwards with higher and higher force. For which value of $y'(0)$ does chaotic behaviour appear? Does it disappear again for higher $y'(0)$?

11. Study the effect of the following factors on the chaotic dynamics (that is, repeat the analysis of question 10) and try to identify the root cause of chaos.

- Accuracy of the numerical method (value of n at constant T).
- Discrepancy at the start (value of ε).
- Initial conditions s_0, θ_0. Is the spring pendulum always chaotic if $y'(0)$ is large enough i.e. at large enough energy?

Hint for all questions. Chaotic behaviour can directly be observed via the pendulum's movement, but this can prove time-consuming. One option is to plot the positions of Pendulum 1 and 2 as functions of time and see whether or when they diverge. Another option is to compute a time t_c such that when $t > t_c$ there is a visible gap between the two pendulums (e.g. 1 cm). This can be especially useful in Question 11 where you can then plot t_c as a function of ε for instance.

Independent work

Implement one or two experiments of your choice on this system. Here are a few suggestions but you are welcome to use your own ideas.

- Study the 3D string pendulum. The equations of motions become

$$\begin{cases} \dfrac{d^2x}{dt^2} &= -\dfrac{k}{m}\big(L(t)-L_0\big)\dfrac{x(t)}{L(t)} \\[2ex] \dfrac{d^2y}{dt^2} &= -\dfrac{k}{m}\big(L(t)-L_0\big)\dfrac{y(t)}{L(t)} \\[2ex] \dfrac{d^2z}{dt^2} &= -\dfrac{k}{m}\big(L(t)-L_0\big)\dfrac{z(t)}{L(t)} - g \end{cases}$$

  with $L = \sqrt{x^2+y^2+z^2}$. Initial conditions are now determined by a stretching s_0 and two angles θ_0 and φ_0 (in spherical coordinates). Plot solutions for different initial values, both in 3D and from the above or from a side. Possibly investigate chaos as well.

- Add air resistance to the equations of motion. For a spherical ball of radius R the drag force is

$$\boldsymbol{F}_{\text{drag}} = -\frac{\pi R^2}{2}\rho C_D|\boldsymbol{v}|\boldsymbol{v}$$

  where ρ is the density of the ambient medium, $C_D = 0.47$ is the drag coefficient for a ball, and $\boldsymbol{v}$ is the velocity vector. Air has density $\rho = 1.225\,\text{kg/m}^3$.
  Show plots for various initial conditions and dimensions of the ball. Plot the energy as a function of time and calculate dissipation due to friction.

- Study the effect of changing the parameters L, m and k, especially regarding the chaotic dynamics.

Grading for this part depends both on difficulty and the quality of the solution.

Hints

Question 2. The structure of the program could for instance be:

```
function w=Eulersolver(s0,theta0,T,n) %might need more inputs later
%calculate initial position x0, y0 and step size h
w(1,:)=[x0,0,y0,0] %initial conditions
for i=1:n
     w(i+1,:)=w(i,:)+eulerstep(w(i,:),h);
end
end

function z=eulerstep(w,h) %will eventually be replaced by RKstep(w,h)
z=h*system(w);
end

function z=system(w)
%the differential system
%z(1)=...
%z(2)=...
%z(3)=...
%z(4)=...
end
```

Questions 3-4 - Animation tips. The **animatedline** command defines an animated object in Matlab. For instance

```
ball = animatedline('color','b','Marker','.','markersize',20);
```

defines the ball as a large filled blue circle. Using the output w of **Eulersolver** one can extract vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ containing all necessary coordinates. The loop

```
for k = 1:n+1
    clearpoints(ball)
    addpoints(ball,x(k),y(k));
    pause(0.01) %optional, changes speed
    drawnow %set to comment if only final picture is needed
end
```

first erases the previous position using **clearpoints** then plots the new position using **addpoints**. Similarly you can define a second animated object for the spring. It is also very much recommended to define a third trace object which shows the whole trajectory for the ball, in which case the **clearpoints** should not be used.

In order to draw a proper wiggly spring, a possible solution is as follows. First off define the string as an animatedline:

```
spring = animatedline('Color','k','LineWidth',1);
```

and use the following code snippet:

```
L=sqrt(x.^2+y.^2); %calculates length of spring at all time steps

for k = 1:n+1
    clearpoints(spring)
    omega=12*pi/L(k); %12 controls the number of coils
    xx=linspace(0,L(k),200);
    yy=0.1*sin(omega*xx); %plot(xx,yy) draws a spring pointing to the right. The
       string is stretched or compressed according to the length L(k)

    theta=atan2(y(k),x(k)); %rotating angle is arctan(y/x)
    xp = xx*cos(theta) - yy*sin(theta); %multiply (xx,yy) by a rotation matrix
    yp = xx*sin(theta) + yy*cos(theta); %xx as a vector connects (0,0) to the
       position of the string. yy is the coily plot in the direction of xx.

    addpoints(spring,xp,yp)
    pause(0.01)
    drawnow
end
```