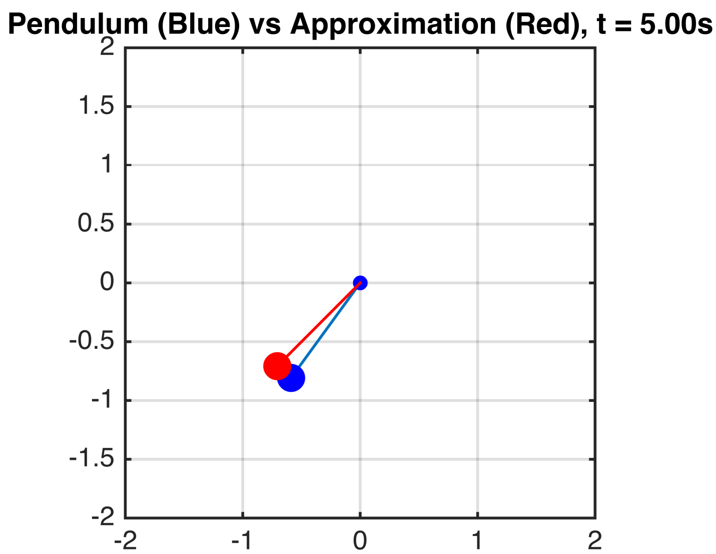
MECH 221 Computer Lab 4

A Simple Pendulum: Numerical Solutions and the Small Angle Approximation




```
>> M = pendulum(1,[0,5],[pi/4,0],50);
>> figure, axes('Position',[0 0 1 1]), movie(M)
```

Introduction

A simple pendulum is an idealized mathematical model of a pendulum consisting of a mass swinging on a massless rod. The equation of motion is

$$\ddot{\theta} + \frac{g}{L}\sin\theta = 0$$

where g is acceleration due to gravity, L is the length of the pendulum and θ is the angle between the rod and the vertical. Therefore, if the top of the rod is fixed at the origin $(0,0)$ in the coordinate system, the position of the swinging mass has coordinates

$$x = L\sin\theta$$
$$y = -L\cos\theta$$

If the angle θ is small, then it is common to use the small angle approximation $\sin\theta \approx \theta$ and rewrite the equation of motion of the pendulum as

$$\ddot{\theta} + \frac{g}{L}\theta = 0$$

where the general solution is

$$\theta(t) = \frac{\dot{\theta}(0)}{\omega_n} \sin(\omega_n t) + \theta(0)\cos(\omega_n t) \ , \ \omega_n = \sqrt{\frac{g}{L}}$$

The goal of this lab is to simultaneously animate the motion of a simple pendulum using the numerical solution produced by `ode45` of the true equation of motion of the pendulum, and also the solution given by the small angle approximation. In particular, when you run the animation with large angles (perhaps with $\theta(0) = 3\pi/4$ and $\dot{\theta}(0) = 0$), the animation demonstrates the large error in the solution given by the small angle approximation.

Instructions

Write a function called `pendulum` which **simultaneously** animates the motion of a simple pendulum in two ways:

```
function M = pendulum(L,tspan,y0,fps)
% Animate the motion of a simple pendulum in two ways:
% 1. Use ode45 to solve the the nonlinear equation:  y'' + (g/L)sin(y) = 0
% 2. Use the small angle approximation:  y'' + (g/L)y = 0
```

- ☐ `pendulum` has 4 input parameters:
  - ☐ `L` is the length of the pendulum rod (in metres)
  - ☐ `tspan` is a vector of length 2:

    `tspan(1)` is the start time of the animation (in seconds)

    `tspan(2)` is the end time of the animation (in seconds)
  - ☐ `y0` is a vector of length 2:

    `y0(1)` is the initial angle θ at time `tspan(1)` (in radians)

    `y0(2)` is the initial angular velocity $\dot{\theta}$ at time `tspan(1)` (in radians per second)
  - ☐ `fps` is the number of frames to plot per second
- ☐ Use `ode45` to find a numerical solution of the equation of motion of the pendulum

  $$\ddot{\theta} + \frac{g}{L}\sin\theta = 0$$

  and animate the result.
- ☐ Animate the exact solution of the equation given by the small angle approximation

  $$\ddot{\theta} + \frac{g}{L}\theta = 0$$
- ☐ `pendulum` has 1 output parameter:
  - ☐ `M` is an array of movie frames captured by `getframe` which can be played back using the function `movie`
- ☐ Include a title which displays the time value (which is changing over the animation from `tspan(1)` to `tspan(2)`)

When you are satisfied with your function, submit your M-file (called `pendulum.m`) to Connect.

Hints

Here is a list of steps to guide you through the lab:

1. Write the second order equation of motion of the pendulum as a system of first order equations. In particular, let $u_1 = \theta$ and $u_2 = \dot{\theta}$ and write

$$\begin{aligned} \dot{u}_1 &= u_2 \\ \dot{u}_2 &= (-g/L)\sin u_1 \end{aligned}$$

2. Write a function `odefun` within your function `pendulum` which defines the right side of the system of equations. Check out the documentation on nested functions:

   `https://www.mathworks.com/help/matlab/matlab_prog/nested-functions.html`

3. When you call `ode45`:

```
[T,Y] = ode45(@odefun,t,y0)
```

   the input `t` is an array of time values and `y0` is a vector of initial conditions. The output `T` will be the same as `t` and the array `Y` will consist of the solution values at the corresponding values of `T`. Check out the documentation for `ode45`:

   `https://www.mathworks.com/help/matlab/ref/ode45.html`

4. Look at the MATLAB documentation on animation techniques:

   `https://www.mathworks.com/help/matlab/creating_plots/animation-techniques-1.html`

   For example, the following function creates an array `M` of frames which can be played back with the command `movie`:

```
function M = circle(num_frames)
t = linspace(0,2*pi,num_frames);
x = cos(t); y = sin(t);
for i = 1:length(t)
    plot(x,y,'b--');
    xlim([-2,2]); ylim([-2,2]);
    hold on; axis off equal;
    plot(cos(t(i)),sin(t(i)),'ro','MarkerFaceColor','r');
    hold off;
    M(i) = getframe(gcf);
end
end
```

5. To make our animation smooth, the array `t` should consist of `fps*(tspan(2)-tspan(1))` evenly spaced values.

6. To animate the motion of the pendulum, write a `for` loop which loops over the time values `T` and plots the position of the mass given the `ode45` solution. The x and y positions of the mass are written in the introduction above as functions of the angle θ. Note, the first column `Y(:,1)` gives the angle θ of the pendulum over time. To plot the mass as a large blue dot, you could modify the following command which plots a big blue dot at the origin:

```
plot(0,0,'bo','MarkerSize',10,'MarkerFaceColor','b');
```

7. To animate the rod, simply plot a line which connects the base of the rod to the mass. For example, the following command will plot a line connecting $(0, 0)$ and $(1, -1)$:

```
plot([0,1],[0,-1]);
```

8. In the same `for` loop, plot the position of the mass defined by the general solution of the equation given by the small angle approximation as described in the introduction above. Also plot a line connecting the origin to the mass to represent the rod of the pendulum. Use a different color to distinguish it from the solution given by `ode45`. For example, the image above shows the numerical solution in blue and the small angle approximation solution in red.

9. Use `axis equal`, `xlim` and `ylim` to display the animation properly.

10. Finally, the following commands create a new figure, create an axes object which fills the figure window (from bottom left $(0, 0)$ to top right $(1, 1)$) and displays the animation (stored in the variable `M`, the output of the function `pendulum`):

```
>> M = pendulum(1,[0,5],[pi/4,0],50);
>> figure, axes('Position',[0 0 1 1]), movie(M)
```