AMATH 301
Homework 8
Due: Friday, March 8, 2019

Linear Pendulum

Consider the linear second-order differential equation for the motion of a pendulum,

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = \frac{-g}{l}\theta. \tag{1}$$

Here θ (theta) is the angle of deflection of the pendulum from the vertical. $g = 9.8$ is acceleration due to gravity, $l = 10$ is the length of the pendulum.

By defining ϕ (phi) as the derivative $\phi = \frac{\mathrm{d}\theta}{\mathrm{d}t}$, we can convert this single second-order ODE into a system of two first-order ODEs,

$$\begin{cases} \dfrac{\mathrm{d}\theta}{\mathrm{d}t} = \phi \\ \dfrac{\mathrm{d}\phi}{\mathrm{d}t} = \dfrac{-g}{l}\theta. \end{cases}$$

This is a standard 'trick' that is used very often. The reason this trick is so useful is that we don't need separate methods to deal with every order of ODE. Instead, we can just turn all ODEs of any order into systems of first-order ODEs.

This system of ODEs is linear and can in turn be written in matrix form,

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\theta \\ \phi\end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \dfrac{-g}{l} & 0 \end{bmatrix}\begin{bmatrix}\theta \\ \phi\end{bmatrix}.$$

We will investigate the following initial value problem, over the time interval $0 \le t \le T$ for $T = 50$,

$$\begin{cases} \dfrac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = \dfrac{-g}{l}\theta, \qquad 0 < t < T \\ \theta(0) = 1 \\ \theta'(0) = 0. \end{cases} \tag{2}$$

Problem 1 (Scorelator).

Learning Goal: Implement the Forward Euler time-stepping method for solving a system of ODEs.

Learning Goal: Explain the connection between the forward difference scheme for the first derivative and the Forward Euler time-stepping method.

Forward Euler is based on using a forward difference to approximate the derivative $\frac{\mathrm{d}\vec{Z}}{\mathrm{d}t}$, and gives rise to an explicit scheme (we have a direct formula for $\vec{Z}_{k+1}$ in terms of $\vec{Z}_k$),

$$\frac{\vec{Z}_{k+1} - \vec{Z}_k}{\Delta t} = f(\vec{Z}_k) \qquad \Rightarrow \qquad \vec{Z}_{k+1} = \vec{Z}_k + \Delta t f(\vec{Z}_k).$$

Implement the forward Euler method with $\Delta t = 0.01$ to solve the IVP (2). Here $\vec{Z}$ is a state variable with components θ and ϕ. Save the values of θ and ϕ at each time into column vectors `thsave_FE` and `phisave_FE`; both vectors should have 5001 elements. You should have `thsave_FE(1) = 1`, the initial condition for θ, and `phisave_FE(1) = 0`, the initial condition for the derivative. Save `thsave_FE` to **A1.dat** and `phisave_FE` to **A2.dat**.

Problem 2 (Scorelator).

Learning Goal: Explain the difference between explicit and implicit time-stepping methods.

Learning Goal: Explain the connection between the backward difference scheme for the first derivative and the Backward Euler time-stepping method.

Backward Euler is based on using a backward difference to approximate the derivative. We obtain an implicit scheme (we have to solve for $\vec{Z}_k$ instead of just plugging in numbers we already have to a direct formula),

$$\frac{\vec{Z}_k - \vec{Z}_{k-1}}{\Delta t} = f(\vec{Z}_k).$$

Solving for the new iterate $\vec{Z}_k$ requires doing algebraic manipulation by hand. If $f(\vec{Z}_k)$ is a linear system, then this means we must solve a linear system.

Implement the backward Euler method with $\Delta t = 0.01$ to solve the IVP (2). Save the values of θ and ϕ at each time into the column vectors `thsave_BE` and `phisave_BE`. Save these vectors to **B1.dat** and **B2.dat**.

Problem 3 (Scorelator).

Learning Goal: Explain the connection between the central difference scheme for the first derivative and the Leapfrog time-stepping method.

The Leapfrog method is based on using a central difference to approximate the derivative,

$$\frac{\vec{Z}_{k+1} - \vec{Z}_{k-1}}{2\Delta t} = f(\vec{Z}_k) \qquad \Rightarrow \qquad \vec{Z}_{k+1} = \vec{Z}_{k-1} + 2\Delta t f(\vec{Z}_k).$$

This method is explicit like Forward Euler, because the formula we end up with only depends on past solution values. A major difference between this method and the Euler methods is that it is a *multistep* method – the iteration actually uses two past solution values: $\vec{Z}_{k-1}$ and $\vec{Z}_k$. This means that each time we generate a new iterate $\vec{Z}_{k+1}$, we need to use the 'current' solution value $\vec{Z}_k$ as well as the one before it, $\vec{Z}_{k-1}$.

Since this method is a multistep method, we actually cannot start the iteration with just the initial values. The problem is that to compute $\vec{Z}_1$, the iteration calls for

$$\vec{Z}_1 = \vec{Z}_{-1} + 2\Delta t f(\vec{Z}_0),$$

which is a problem since $\vec{Z}_{-1}$ isn't defined yet. To get around this, we perform a single step of Forward Euler to calculate $\vec{Z}_1$, and then switch over to Leapfrog.

Implement the Leapfrog method with $\Delta t = 0.01$ to solve the IVP (2). Save the values of θ and ϕ at each time into the vectors `thsave_LF` and `phisave_LF`. Save these vectors to **C1.dat** and **C2.dat**.

Problem 4 (Scorelator).

Learning Goal: Use the MATLAB function `ode45` to solve a system of ODEs.

Use the MATLAB function `ode45` to solve the IVP (2). There are multiple ways to invoke `ode45`. Make sure that you obtain a solution with evenly spaced times.

`ode45` will output a solution matrix. Pull out the relevant information to assemble the vectors `thsave_ODE45` and `phisave_ODE45` and save them to **D1.dat** and **D2.dat**.

Problem 5 (Writeup).

Learning Goal: Graphically illustrate the growth in error over time of numerical solutions of ODEs.

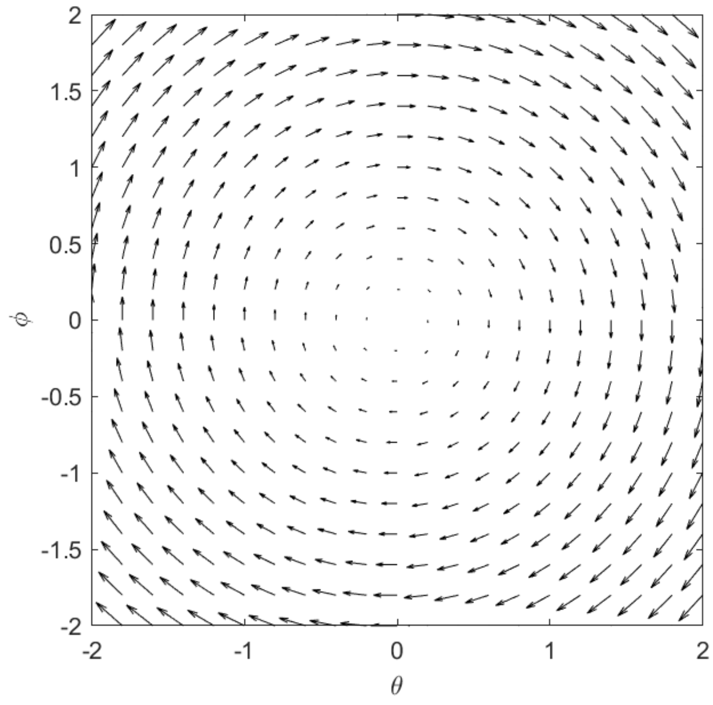
Plot the numerical solutions for $\theta(t)$ that you obtained in Problems 1–3 all on the same axes. Add a horizontal line at the level $\theta = 1$, which corresponds to the maximum amplitude that the pendulum should swing through. Label your axes and add a legend distinguishing the trajectories from the three solution methods. Save your figure as `linear_pendulum_solutions.png`

Problem 6 (Writeup).

Learning Goal: Construct a phase portrait of a system of two ODEs.

We will now construct a phase portrait of the system. A phase portrait is a lot like a slope field, but more general. At each point (θ, ϕ) in *phase space* (the space of the dependent variables), we associate the vector $(\frac{\mathrm{d}\theta}{\mathrm{d}t}, \frac{\mathrm{d}\phi}{\mathrm{d}t})$. This vector points in the direction that the system will evolve into, as specified by the system of ODEs.

Use `meshgrid` to generate a grid of points between $-2 \le \theta \le 2$ and $-2 \le \phi \le 2$ with 21 linearly-spaced points in both directions. Use the `quiver` function to draw a grid of arrows according to the above specifications: the arrows are positioned at the points (θ, ϕ) with lengths $(\frac{\mathrm{d}\theta}{\mathrm{d}t}, \frac{\mathrm{d}\phi}{\mathrm{d}t})$, where $\frac{\mathrm{d}\theta}{\mathrm{d}t}, \frac{\mathrm{d}\phi}{\mathrm{d}t}$ are given by the ODEs of the pendulum system. You should see something like the following:



Now add plots of $\theta(t)$ versus $\phi(t)$ for the three numerical solutions you obtained in Problems 1–3. These *solution trajectories* should (roughly) follow the arrows from the `quiver` plot. Make sure the colors of the solutions match those from your plot in Problem 5, for consistency. Label the axes and add a legend so that the trajectories can be easily distinguished. Save your figure as `linear_phase_portrait.png`

Nonlinear Pendulum

The linear pendulum equation (1) is based on the small angle approximation $\sin(\theta) \approx \theta$. Without this approximation, the equation is more accurate and becomes

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = \frac{-g}{l}\sin(\theta). \tag{3}$$

This second-order ODE is nonlinear and so is more difficult to solve. In particular, implicit methods like Backward Euler are harder to implement for nonlinear ODEs.

We will investigate the nonlinear, 'corrected' version of the IVP (2),

$$\begin{cases} \dfrac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = \dfrac{-g}{l}\sin\theta, & 0 < t < T = 50 \\ \theta(0) = 1 \\ \theta'(0) = 0 \end{cases} \tag{4}$$

Problem 7 (Scorelator).

Use the MATLAB function `ode45` to solve the IVP (4). There are multiple ways to invoke `ode45`. Make sure that you obtain a solution with evenly spaced times.

`ode45` will output a solution matrix. Pull out the relevant information to assemble the vectors `thsave_ODE45` and `phisave_ODE45` and save them to **E1.dat** and **E2.dat**.

Problem 8 (Writeup).

Learning Goal: Distinguish the behaviors of different solution trajectories in a phase portrait.

Repeat the procedure for drawing a phase portrait from Problem 6.

Use `meshgrid` to generate a grid of points. Use 25 linearly spaced points for θ between -2π and $+2\pi$. Use linearly spaced points with a gap of 0.5 between -3 and $+4$ for ϕ. Use the `quiver` function to draw a grid of arrows. The resulting picture should look similar to what you obtained in Problem 6 in the middle of the plot, but different around the edges (this is due to the nonlinearity of the system).

Use `ode45` to solve the system with the following variety of initial conditions:

- $\theta(0) = 0.1, \theta'(0) = 0$
- $\theta(0) = 1.0, \theta'(0) = 0$
- $\theta(0) = 3.0, \theta'(0) = 0$
- $\theta(0) = -2\pi, \theta'(0) = 2.1$
- $\theta(0) = -2\pi, \theta'(0) = 3$

Add these solution trajectories to the phase portrait as plots of $(\theta(t), \phi(t))$. These solution trajectories should flow with the arrows from `quiver`. You do not need to add a legend, because the trajectories shown in the phase portrait are self-explanatory (do you know why?).

Save your figure as `nonlinear_phase_portrait.png`

These trajectories come in two varieties: those that repeatedly encircle the origin, and those that do not. Can you interpret what behavior of the pendulum these two types of trajectories correspond to?