Homework 8 (optional): Ordinary Differential Equations

CS 205A: Mathematical Methods for Robotics, Vision, and Graphics (Fall 2013),
Stanford University

Due Monday, December 9, at midnight (no extensions)

This homework can be used to replace your lowest homework grade. It is completely optional, and you cannot use late days. If you choose not to complete this assignment, you might look it over, as you will be responsible for the material it covers on the final exam.

Problem 1 (50 points). *In this problem you will implement several methods for simulating a set of particles connected by springs as they move over time. The file* `hw8.m` *contains a testing script.*
Matlab tip: `ctrl+enter` *will run the block of code separated by* `%%` *containing the cursor.*
Note: Since CS 205A is not a physics class, in this problem we will assume all particles have mass 1 and all springs have constant 1.

(a) *Suppose we are given a graph $G = (V, E)$ as in problem 1 of homework 6. We will use this graph to represent a system of particles on the plane attached by springs. In particular, each vertex $v \in V$ will represent a particle in a physical system, and each edge $(u, v) \in E$ will represent a spring attached to particles u and v. If two particles at positions $\vec{x}$ and $\vec{y}$ are connected by a spring, then the spring exerts force $\vec{y} - \vec{x}$ on the particle at $\vec{x}$ and force $\vec{x} - \vec{y}$ on the particle at $\vec{y}$.*

Suppose we write the positions of all the particles in the system in a matrix $X \in \mathbb{R}^{n\times 2}$. Show that you can construct a matrix $F \in \mathbb{R}^{n\times n}$ such that the k-th row of FX is the total force exerted by the spring system on particle k. Implement `forceMatrix.m` *to compute F.*

(b) *By Newton's second law, the ODE $X'' = FX$ determines the time evolution of the system, assuming all the particles have mass one. Recall from lecture that a second-order ODE can be reduced to a first-order ODE $Y' = AY$ for $Y \in \mathbb{R}^{2n\times 2}$. Derive this reduction and write code to find A given F in* `firstOrderMatrix.m`.

(c) *Implement the following three schemes for approximating solutions to $Y' = AY$ and discuss what happens to each method when you take large time steps.*

   (i) *Forward Euler:* `forwardEuler.m`

   (ii) *Backward Euler:* `backwardEuler.m`

   (iii) *Trapezoid:* `trapezoidalODE.m`

(d) *Implement Leapfrog integration for $X'' = FX$ in* `leapfrog.m`.
Hint: You need to formulate a strategy for moving the velocity an initial half step in time.

Problem 2 (50 points). *The swing angle θ of a pendulum under gravity satisfies the following ODE:*

$$\theta'' = -\sin\theta,$$

where $|\theta(0)| < \pi$ and $\theta'(0) = 0$.

(a) *Suppose $\theta(t)$ solves the ODE. Show that the following value (representing the energy of the system) is constant as a function of t:*

$$E(t) \equiv \frac{1}{2}(\theta')^2 - \cos\theta$$

(b) *As you saw in the previous problem, many ODE integrators show bad global behavior as time progresses over larger periods. For instance, forward Euler can add energy to a system by overshooting, while backward Euler tends to damp out motion and remove energy. In many computer graphics applications, quality long-term behavior can be very important, since these large scale issues cause visual artifacts. The class of* symplectic *integrators is designed to avoid this issue.*

Denote $\omega = \theta'$. The symplectic Euler *scheme makes a series of estimates $\theta_0, \theta_1, \theta_2, \theta_3, \ldots$ and $\omega_0, \omega_1, \omega_2, \omega_3, \ldots$ at time $t = 0, h, 2h, 3h, \ldots$ using the following iteration:*

$$\begin{aligned} \theta_{k+1} &= \theta_k + h\omega_k \\ \omega_{k+1} &= \omega_k - h\sin\theta_{k+1} \end{aligned}$$

Define

$$E_k \equiv \frac{1}{2}\omega_k^2 - \cos\theta_k.$$

Show that $E_{k+1} = E_k + O(h^2)$.

(c) *Suppose we make the small-angle approximation $\sin\theta \approx \theta$ and decide to solve the linear ODE $\theta'' = -\theta$ instead. Now, symplectic Euler takes the following form:*

$$\begin{aligned} \theta_{k+1} &= \theta_k + h\omega_k \\ \omega_{k+1} &= \omega_k - h\theta_{k+1} \end{aligned}$$

Write a 2×2 matrix A such that

$$\begin{pmatrix} \theta_{k+1} \\ \omega_{k+1} \end{pmatrix} = A \begin{pmatrix} \theta_k \\ \omega_k \end{pmatrix}.$$

(d) *If we define $E_k \equiv \omega_k^2 + h\omega_k\theta_k + \theta_k^2$, show that $E_{k+1} = E_k$ in the iteration from (c). In other words, E_k is constant from time step to time step.*
Note: If you must, you can use Wolfram Alpha to simplify algebraic expressions, but it is not too difficult to do this one by hand!