CS 370 Fall 2015: Assignment 2

Due October 22, 5:00 PM, in the Assignment Boxes, 4th Floor MC.

Instructor: G. Labahn Office: DC3629 e-mail: *glabahn@uwaterloo.ca*
Lectures: TuTh 8:30-9:50 PHY 235, 1:00-2:20 PHY 150 Office Hours: Tues 11:00-12:00

Web Site: www.student.cs.uwaterloo.ca/~cs370/

Version Oct 14, 2015

1. **(10 marks)**

   Consider the following second-order differential equation

   $$y''(t) + 2ty'(t) - 3y(t) + 6t \;=\; 0$$

   with initial conditions

   $$y(1) = 2 \text{ and } y'(1) = 1.$$

   (a) Write this initial value problem (IVP) as a system of first order ordinary differential equations, with appropriate initial conditions.

   (b) Using the results of part (a), compute an approximation to $y(1.4)$ using the (explicit) midpoint method for this IVP with a constant time step of $h = 0.2$. You may use a calculator to perform the individual arithmetic steps; show your answers to at least 3 significant digits.

   (c) Matlab's ODE solvers require the system of equations be written in the form

   $$\mathbf{z}'(t) = \mathbf{f}(t, \mathbf{z}).$$

   Write a Matlab function called *dynamics* which evaluates the system dynamics function $\mathbf{f}$ for the system you found in (a).

2. **(15 marks)**

   Consider the initial value problem:

   $$y'(x) = y(x) + 2\cos(2x)e^x + 4x + 8, \quad y(0) = 4.$$

   For this particular problem the exact analytical solution is known, namely

   $$y(x) = \sin(2x)\ e^x + 16e^x - 4x - 12.$$

   We will use the exact solution to determine the errors in two numerical methods applied to solve the initial value problem.

(a) Write a Matlab function, `y = Euler(x0,xf,N,y0)`, which takes as input the initial and final x values, `x0` and `xf`, the number of steps, `N`, and the initial value, `y0`, and implements the forward Euler method. It should return the approximate solution as a vector `y` of length N+1 such that

$$\texttt{y}_i \approx y(x_{i-1}) \quad \text{for} \quad i = 1, 2, ..., N+1.$$

Here $x_i = x_0 + ih, h = \frac{x_f - x_0}{N}$, and $y(x_{i-1})$ denotes the exact solution at $x = x_{i-1}$.

(b) Write a Matlab function, `y = ImprovedEuler(x0,xf,N,y0)`, which implements the *improved* Euler method, similar to part (a).

(c) Write a Matlab script to apply the `Euler` function to solve the initial value problem to value $x_f = 4$ with $N = 40$ time steps. Plot the analytical solution (using solid lines) and the numerical solution (using the '.' symbol for each point) on the same graph, for $0 \leq x \leq 4$.

(d) Repeat part (c) using the `ImprovedEuler` function.

(e) Calculate evidence illustrating the order of the forward Euler method as follows. Let $\text{err}(h)$ denote the error when computing with step size h. For a p-order method, $\text{err}(h) \approx Ch^p$ for some constant C, and so the effect of cutting the step size in half is

$$\frac{\text{err}(h/2)}{\text{err}(h)} \approx \frac{1}{2^p}$$

that is, the error is reduced by a factor of 2^p. Therefore, the order p of a given method can be illustrated by calculating the above ratios.

For the forward Euler method, calculate a vector of successive computed values for the solution at $x = 4$ based on cutting the step size in half a number of times. Specifically, use $N = 2^i \times 10$ for $i = 1, 2, ..., 10$. Calculate the errors and the successive ratios $\frac{\text{err}(h/2)}{\text{err}(h)}$. From your experimental evidence, what is the order of the forward Euler method?

(f) Repeat part (e) for the improved Euler method. From your experimental evidence, what is the order of method ImprovedEuler?

3. **(10 marks)**

The Adams-Bashford method is

$$y_{n+1} = y_n + \frac{h}{2}(3f(t_n, y_n) - f(t_{n-1}, y_{n-1})).$$

Determine the local truncation error of this method.

4. **(10 marks)**

Ralston's method is a second-order Runge-Kutta method given by

$$y_{n+1} = y_n + \frac{h}{3}\left(f(t_n, y_n) + 2f(t_n + \frac{3}{4}h, y_n + \frac{3}{4}hf(t_n, y_n))\right).$$

Carry out a stability analysis of this method using the test equation

$$y'(t) = -\lambda y(t), \quad \lambda > 0.$$

Justify your approach, and determine a precise condition for stability.

5. **(25 marks) Angry Birds Simulation**

For this question we will model the dynamics of the remarkably popular game "Angry Birds", where projectile birds are shot at green pigs using a slingshot. Download the MATLAB files `AngryTriangle.m` and `Shoot.m` from the course website. `AngryTriangle.m` is the main program and it invokes `Shoot.m`, which contains an incomplete implementation of the dynamics of the game. Your task is to complete the implementation.

When running `AngryTriangle.m`, a red triangle (representing an angry bird) is initially sitting on a slingshot at $(0.2, 0.2)$, and a green circle (representing a pig) is located at $(1.0, 0.05)$. A mouse click on the triangle can drag the triangle backward, pulling the sling back (to the left). When the button is released, the bird will shoot forward following a trajectory. Also, there is a slight twist: *the pig panics and runs back and forth.* The trajectories of the bird and the pig can be computed using MATLAB's ODE suite based on the mathematical model which is described below.

Mathematical model.

The fixed Cartesian coordinate system has a horizontal x-axis and a vertical y-axis. The triangle's initial velocity has magnitude v_0 and makes an angle with respect to the x-axis of θ_o radians. The only forces acting on the triangle after it is released are gravity and aerodynamic drag. The aerodynamic drag will be proportional to the square of the magnitude of the velocity. A wind force, given by a function $w(t)$, also impacts the flights. The equations describing the triangle's motion are:

$$\frac{dx}{dt} = v\cos(\theta), \quad \frac{dy}{dt} = v\sin(\theta), \quad \frac{d\theta}{dt} = -\frac{g}{v}\cos(\theta), \quad \frac{dv}{dt} = -\frac{D}{m} - g\sin\theta \tag{1}$$

where D is the drag force, given by

$$D = \frac{c \cdot \rho \cdot s}{2}\left(\left(\frac{dx}{dt} - w(t)\right)^2 + \left(\frac{dy}{dt}\right)^2\right).$$

The constant parameters for this problem are the magnitude of gravitational acceleration $g = 9.81$, mass $m = 0.2$, air density $\rho = 1.29$, drag coefficient $c = 0.72$, and the bird's cross-sectional area $s = 0.005$.

Let $p(t)$ represent the location of the pig on the x-axis. The pig will move according to the following initial value problem:

$$\frac{d^2p}{dt^2} = -5\pi(p(t) - 1), \quad p(0) = p_0, \quad \frac{dp}{dt}(0) = 1$$

The initial values for the combined initial value problem are determined by the initial release of the triangle. These values are already computed in the code and provided to `Shoot.m`.

Note: the motion of the bird and the pig are modeled simultaneously by a single system of differential equations. Therefore the pig is only in motion while the bird is in motion. They do not move independently.

The goal of the game is to adjust the shooting position of the triangular red bird so that it lands on the circular green pig.

Instructions. The MATLAB function `Shoot.m` takes as inputs the initial position vector `init_pos`, initial angle `init_theta`, magnitude of the initial velocity `init_vel` of the triangle, and the initial position (x-coordinate) of the pig `init_p`, and solves the first order system of ODEs using the MATLAB function `ode45`.

Please complete the missing parts of `Shoot.m` as follows:

- By writing the differential equations of motion in vector form $\mathbf{z}'(t) = \mathbf{f}(t, \mathbf{z})$, complete the ODE dynamics function `motion_ode`.
- The ODE solver `ode45` should take as inputs the function `motion_ode`, the time span $[0 : 0.05 : 5]$, the initial values, and the ODE options given by `options`. The output `t` is a vector containing the time steps. The output `z` is a matrix whose first, second, third, fourth, and fifth columns are the numerical approximations of $x(t), y(t), \theta(t)$, $v(t)$, and $p(t)$, respectively, at the corresponding time steps given by `t`. (A sixth column/variable will also be needed after converting the ODE system to the desired first order form.)
- In the event function `events`, the first event is used to detect when $y(t)$ becomes negative ($y < 0$). The second event detects when $x(t)$ becomes greater than 2 ($x > 2$, which occurs when the triangle has gone off the right side of the figure). Define a third event which detects when the triangle makes contact with the circle according to the following criterion:

  $$\sqrt{(x(t) - x_p)^2 + (y(t) - y_p)^2} < 0.05$$

  where (x_p, y_p) is the current position of the pig. (Recall that the vertical coordinate of the centre of the pig is $y_p = 0.05$). You might consider using the `norm` command.
- Set the wind parameter to a steady value $w(t) = -8$ for all t. Run `AngryTriangle`. Move the initial position of the red triangle to approximately $(0.14, 0.1)$ and let it go. It should land at $y = 0$ just short of the pig's position. Now, move the second red triangle to approximately $(0.14, 0.16)$, and let it go (do not call `AngryTriangle` a second time.) It should also land at $y = 0$ but closer to the slingshot than before. Print the figure with the plots of the two trajectories. Save the figure with the plots of the two trajectories as `misses.pdf`.

- Run `AngryTriangle` again. By trial and error, adjust the initial position of the red triangle so that it will land on the green circular pig. (You should see the screen print the text "Hit!") Save the figure with a plot of the successful trajectory as `hit.pdf`.

Submit `Shoot.m` *and the saved figures to the DropBox on LEARN.*