CS 370 Fall 2008: Assignment 5

Instructor: Professor Keith Geddes

Lectures: MWF 3:30-4:20 MC 2017

Web Site: UW-ACE

Due: Mon Dec 1, 2008, 5:00 pm, in the Assignment Boxes, 3rd Floor MC

Task 1

With pencil and paper, compute two steps of the Forward Euler method using stepsize $h = 0.1$ for the initial value problem

$$\begin{aligned} x'(t) &= 4\,x(t) - 2\,y(t) - 4\,t - 2 \\ y'(t) &= 3\,x(t) + 5\,t \end{aligned}$$

with initial conditions: $x(0) = 4\,,\ \ y(0) = -5\,.$

Task 2

In this task, you will write your own ODE solving suite that performs the Modified Euler method, and use the suite to solve for the trajectory of a cannon ball. The system of differential equations that governs the motion of a ballistic projectile, like a cannon ball, is

$$\begin{aligned} x''(t) &= -K\ x'(t) \\ y''(t) &= -g - K\ y'(t) \end{aligned}$$

where g is $9.81\,\mathrm{m/s^2}$ and K is a coefficient of air resistance. Use the value $K = 0.2$ for this task.

1. Write a Matlab function called `MyOde` with the prototype

   ```
   function [t, y] = MyOde(f, tspan, y0, N, events) .
   ```

   See the help documentation for the supplied skeleton code for an explanation of the input and output variables.

2. Formulate the above system of ODEs into a first order system, and create a Matlab function called `ballistics` that fits the prototype of the dynamics function described in `MyOde` help.

3. Write a simple event function called `ballistics_events` that switches sign as the cannon ball hits the ground. This function should also fit the prototype described in `MyOde` help.

4. The supplied `Cannon.m` script uses your ODE suite to numerically solve the above IVP for the cannon ball trajectory. The initial state is set so that the cannon ball starts at the origin, and has an initial speed of $200\,\mathrm{m/s}$ at an angle of θ° from the ground. Run the script for a chosen θ-value. Note that the initial angle and landing distance is stated in the plot title.

5. Re-run the script for different θ-values, and determine the angle (to the nearest degree) that gives you the longest shot.

Task 3

The pursuit model presented in the Course Notes is not physically realistic – a missile does not typically have a constant speed. Newton's law of motion, $F = ma$, states that the acceleration of an object is proportional to the net force on the object. A missile has a rocket engine that generates a force. For this task, we will assume the magnitude of the force is constant. We will also add air resistance, which acts as a force in the direction opposite to the missile's motion. Moreover, we will model the system in 2D, rather than 3D. The system of ODEs for this model is

$$\begin{aligned} x''(t) &= Fu_x - Kx'(t) \\ y''(t) &= Fu_y - Ky'(t) \end{aligned}$$

where K (a positive constant) is the coefficient of air resistance, F is the constant force from the rocket engine, and $u = [u_x \; u_y]^{\mathrm{T}}$ is a unit vector. Once again, assume that the missile is always pointed at the target, so u is a unit vector pointing from the missile to the target.

Show that the above system of ODEs can be converted to the system of first-order ODEs

$$\begin{aligned} z_1'(t) &= z_3(t) \\ z_2'(t) &= z_4(t) \\ z_3'(t) &= Fu_x - Kz_3(t) \\ z_4'(t) &= Fu_y - Kz_4(t) \; . \end{aligned}$$

In this task, you will alter a set of Matlab files to implement the above pursuit problem.

Matlab Files

pursuit.m: This is a Matlab script that sets a number of simulation parameters, calls `odeset` to set options for the ODE solver, calls the ODE solver, and plots the resulting trajectories.

target.m: This Matlab function returns the target's position. There is one trajectory already implemented (selected by the 2nd input parameter `target_number`). There is space for a 2nd trajectory as well.

missile.m: This Matlab function will contain the system dynamics equations for the missile (pursuer). It is currently blank, returning $[0 \; 0]^{\mathrm{T}}$.

pursuit_events.m: This is a Matlab terminal event function. It is currently blank, simply returning the value 1.

Extending the Matlab Files

1. Edit the file `missile.m` by adding the code required to implement the dynamics for the pursuer problem. The calling sequence for the function should also include an argument for the missile force (`F`), the coefficient of air resistance (`K`), and which target to use (`target_number`).

2. Edit the file `target.m` by implementing a 2nd trajectory (corresponding to `target_number`=2) that follows the curve

$$\begin{aligned} x_{\mathrm{T}}(t) &= 1 - 2\,|\mathrm{mod}(t, 2) - 1| \\ y_{\mathrm{T}}(t) &= 1 + t \; . \end{aligned}$$

3. Edit the terminal event function file `pursuit_events.m` to encode the event that the missile acquires its target. Assume that the missile acquires its target if it is within a distance `dmin` of the target (defined in the script `pursuit.m`). Thus, `dmin` should be an argument passed into the event function. (*Hint*: You will also have to change the arguments passed into `missile.m` to get this to work with Matlab's ODE functions.)

4. Alter the file `pursuit.m` to work with the new versions of the above functions. This involves changing the call to the ODE solver (`ode45`), and enabling the terminal event capabilities in the ODE solver options.

Examining several trajectories

For each of the scenarios below, create two plots: one showing the trajectories of the target and the missile, and one showing the time course of the distance between the two.

(a) `target_number` $= 1$, $F = 10$, and $K = 1$.
(b) `target_number` $= 2$, $F = 5$, and $K = 1$.
(c) `target_number` $= 2$, $F = 6$, and $K = 0$.

Put a title on each plot indicating which scenario it is, and label the axes appropriately.

What to submit

Task 1

1. Your handwritten computations for the two steps of the Forward Euler method.

Task 2

Printouts of:

1. Your completed `MyOde.m` function.
2. Your dynamics function `ballistics.m` .
3. Your terminal events function `ballistics_events.m` .
4. A plot of the cannon ball trajectory for your chosen firing angle.
5. A plot of the cannon ball trajectory for the furthest firing distance.

Task 3

1. A written derivation of the first-order system of ODEs arising from the original system of ODEs.
2. Printouts of the revised scripts `pursuit.m`, `missile.m`, `target.m` and `pursuit_events.m` .
3. Three pairs of plots. Each pair consists of: one plot of the target and missile trajectories, and one plot of the history of the distance between them.