CS 370 Winter 2018: Assignment 2

Due February 12, noon.

Instructor: G. Labahn Office: DC3629 e-mail: *glabahn@uwaterloo.ca*
Lectures: MWF 8:30, 11:30 Office Hours: Tues 11:00-12:00

Instructor: Y. Li Office: DC3623 e-mail: *yuying@uwaterloo.ca*
Lectures: MWF 1:30 Office Hours: Thurs 2:00-3:00

Web Site: cs370 piazza

Your assignment should be handed in electronically on UW Learn. The submission should include one pdf file containing the assignment answers and the cover sheet and all the m-files required to run the code, with no folder structure (not zipped).

1. **(15 marks)** The suspension system in a car can be modeled by the system of differential equations for $u(t), v(t)$,

$$u''(t) + c_1(u'(t) - v'(t)) + k_1(u(t) - v(t)) + k_2 u(t) = \sin(t)$$
$$v''(t) + c_2(u(t) - v(t)) - c_1(u'(t) - v'(t)) = 0,$$

   where c_1, c_2, k_1, k_2 are constants. The initial conditions are:

$$u(0) = 1,\ v(0) = 2,\ u'(0) = 0,\ v'(0) = 0.$$

   (a) Write this system as a system of first order ordinary differential equations.

   (b) Matlab's ODE solvers (e.g. ode23) require the system of equations be written in the form

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

   The function which evaluates $\mathbf{f}$ is then passed to the ODE solver. Write a Matlab function called *dynamics* which evaluates the dynamics function $\mathbf{f}$ for the system in part (a). (The constants c_1, c_2, k_1, k_2 may be specified as parameters of the function.)

2. **(15 marks)** Consider the initial value problem

$$y'(t) = -3t^2 y(t)^2, \quad y(0) = \frac{1}{2}.$$

   For this particular IVP it is easily checked that

$$y(t) = \frac{1}{2 + t^3}$$

   is the exact analytical solution. We will use this exact solution to determine the errors in various numerical methods applied to solve the IVP.

   (a) Write a Matlab script to apply the `ForwardEuler` function from the course web site to solve the IVP to value $t_f = 5$ with $N = 50$ time steps. Plot the analytical solution (using solid lines) and the numerical solution (using the '.' symbol for each point) on the same graph, for $0 \le t \le 5$.

(b) Write a Matlab function, `y = ModifiedEuler(f,t0,tf,N,y0)`, which takes as input a system dynamics function f, the initial and final t values, `t0` and `tf`, the number of steps, `N`, and the initial value, `y0`, and implements the modified Euler method. It should return the approximate solution as a vector `y` of length N+1 such that

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

where $t_i = t_0 + ih, h = \frac{t_f - t_0}{N}$, and $y(t_{i-1})$ denotes the exact solution at $t = t_{i-1}$.

(c) Repeat part (a) using the `Modified Euler` function from part (b).

(d) 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, $\text{err}(h) \approx c \cdot h^p$ for some constant c, and so the effect of cutting the step size in half is

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

i.e. 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 $t = 5$ based on cutting the step size in half a number of times. Specifically, use $N = 2^i \times 10$ for $i = 0, 1, ..., 10$. 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?

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

3. **(10 marks)** Suppose we are solving the ordinary differential equation

$$y'(t) \quad = \quad f(t, y(t)) \ .$$

Determine the local truncation error of the method

$$y_{n+1} \quad = \quad y_{n-1} + 2hf(t_n, y_n)$$

where y_n is an approximation to $y(t_n)$, and $t_n = nh$ with h the stepsize.

4. **(10 marks)** Consider the second-order Runge-Kutta method given by

$$\begin{aligned} {y^*}_{n+1} &= y_n + 3hf(t_n, y_n)/4 \\ y_{n+1} &= y_n + \frac{h}{3}\left[f(t_n, y_n) + 2f\left(t_n + 3h/4, {y^*}_{n+1}\right)\right]. \end{aligned}$$

Analyse the stability of this method by applying the test equation

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

What is your conclusion about stability limitations on h?

5. **(25 marks)** Consider a target moving in a two dimensional plane with location (x_t, y_t). The target is moving with constant speed $S_t = 2.0$, in direction given by the unit vector $(\cos\theta_t, \sin\theta_t)$, where θ_t is the angle of the velocity vector with respect to the x axis.

The target is pursued by a pursuer, with location (x_p, y_p). The pursuer is moving with constant speed $S_p = 3.0$, in direction given by the unit vector $(\cos\theta_p, \sin\theta_p)$, where θ_p is the angle of the velocity vector with respect to the x axis.

The pursuer attempts to hit the target, by turning as fast as possible towards the target. The minimum turning radius of the pursuer is R_p, and the minimum turning radius of the target is $R_t = 0.25$.

The trajectories of the pursuer and target can be described by the following system of ODEs, where t is the time.

$$\begin{aligned} x_p'(t) &= S_p \cos\theta_p(t), \quad y_p'(t) = S_p \sin\theta_p(t), \quad \theta_p'(t) = S_p \frac{\theta_d(t) - \theta_p(t)}{(|\theta_d(t) - \theta_p(t)| + D)} \\ x_t'(t) &= S_t \cos\theta_t(t), \quad y_t'(t) = S_t \sin\theta_t(t), \quad \theta_t'(t) = \frac{S_t}{10 R_t} \ . \end{aligned}$$

where

$$\begin{aligned} (x_p, y_p) &= \texttt{x, y coordinates of pursuer} \\ (x_t, y_t) &= \texttt{x, y coordinates of target} \\ \theta_t &= \texttt{target moving in direction } \cos\theta_t \hat{x} + \sin\theta_t \hat{y} \\ \theta_p &= \texttt{pursuer moving in direction } \cos\theta_p \hat{x} + \sin\theta_p \hat{y} \\ S_p &= \texttt{speed of pursuer} \\ S_t &= \texttt{speed of target} \\ R_t &= \texttt{minimum turning radius, target} \\ \theta_d &= \texttt{direction from pursuer to target (see Figure 1)} \\ D &= \texttt{turning damping factor, pursuer} \end{aligned} \tag{1}$$

The damping factor D in equation (1) prevents the pursuer direction from changing too rapidly when $\theta_d \simeq \theta_p$.

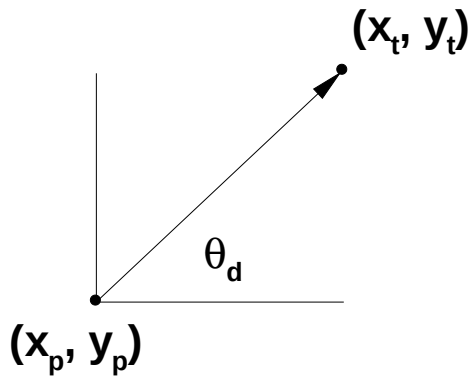

Figure 1: Direction from pursuer to target.

Parameter	Value
Initial Target Position	$(x_t, y_t) = (5, 5)$
Initial Pursuer Position	$(x_p, y_p) = (0, 0)$
Initial Target Direction	$\theta_t = 0.0$
Initial Pursuer Direction	$\theta_p = -\pi$
Integration Interval	$[0, 20]$
R_t	0.25
S_p	3.0
S_t	2.0
Damping Factor	$D = 10^{-3}$
Hitting Distance	$d_{stop} = 10^{-2}$
Error Tolerance	$tol = 10^{-6}$

Table 1: Data for pursuit problem.

Objective

The objective of this assignment is to develop Matlab code which solves the pursuit-target problem, and determines the time (if it occurs) when the pursuer hits the target. The pursuer is considered to have hit the target if the distance between the target and the pursuer is less than d_{stop}, i.e.

$$\sqrt{(x_t - x_p)^2 + (y_t - y_p)^2} \leq d_{stop} \ . \tag{2}$$

The data for this problem is given in Table 1.

(a) (10 marks) Write Matlab code to solve the ODE system (1). Use an event function (see course notes) to stop the simulation if the hitting criterion (2) is satisfied. In order to determine the direction θ_d to the target (equation (1)), we have to be careful, since θ_p, θ_p can be less than or larger than 2π (the pursuer/target could wind around a central point several times). As well, we want to make sure the pursuer turns towards the target in the shortest way possible. Use the supplied function *direction.m* (available on the course web site) to compute θ_d.

Use the stiff solver $ode15s$. Use the following options to be passed to the ODE solver (see the course notes).

```
options = odeset('AbsTol', tol,'RelTol',tol,'MaxOrder',5,'Stats','on',...
            'Events',@event, 'Refine',4);
```

Submit a hard copy of your code, a plot of the trajectories of the pursuer and target, and the hitting time. The trajectory of the pursurer is given by the the parametric curve $(x_p(t), y_p(t))$, and the trajectory of the target is given by the parametric curve $(x_t(t), y_t(t))$ for $t \in [0, T]$ where T is the stopping time or end of the integration interval (whichever is first).

(b) (5 marks) Now, carry out some tests to see the effect of the error control *tol* on the solution accuracy. Fill in Table 2. Explain what you see.

tol	Hitting Time	Number of Function Evaluations
10^{-3}		
10^{-4}		
10^{-5}		
10^{-6}		
10^{-7}		
10^{-8}		
10^{-9}		

Table 2: Test of error control

(c) (5 marks) Now, try solving the problem again, except use *ode*45 as the solver. Use $tol = 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}$. Compare with the solution using *ode*15*s*. Explain what you see.

(d) (5 marks) Attempt to devise an evasion strategy for the target. The target is constrained so that $S_t = 1.0$, and the $\omega_t \leq 1$. Some ideas: when the distance between the target and the pursuer is less than a small multiple of R_t, try turning the target in a direction away from the pursuer. Increase the maximum integration interval to $[0, 40]$.

Submit a pseudo-code description of your evasion strategy, a hard copy of your matlab code, and a plot of the target and pursuer trajectories.

The five most interesting strategies (as determined by the TAs), will earn 3 bonus marks. In the case of disputes, the TAs decisions are final.