Solutions - Assignment 5 CS 370 Fall 2008

Task 1

The given IVP is (with $y_1(t) = x(t)$, $y_2(t) = y(t)$):

$$\underline{y}' = \underline{F}(t, \underline{y}) \quad \text{where} \quad \underline{F}(t, \underline{y}) = \begin{bmatrix} 4y_1(t) - 2y_2(t) - 4t - 2 \\ 3y_1(t) + 5t \end{bmatrix}.$$

and the initial conditions are:

$$\underline{y}(0) = \begin{bmatrix} 4 \\ -5 \end{bmatrix}.$$

The Forward Euler method is

$$\underline{y}^{(n+1)} = \underline{y}^{(n)} + h \cdot \underline{F}(t^{(n)}, \underline{y}^{(n)}).$$

$\boxed{h = 0.1}$; $\boxed{t^{(0)} = 0}$

$$\underline{y}^{(1)} = \underline{y}^{(0)} + h \cdot \underline{F}(t^{(0)}, \underline{y}^{(0)})$$

$$= \begin{bmatrix} 4 \\ -5 \end{bmatrix} + (0.1) \begin{bmatrix} 4\cdot(4) - 2\cdot(-5) - 4\cdot(0) - 2 \\ 3\cdot(4) + 5\cdot(0) \end{bmatrix}$$

$$= \begin{bmatrix} 4 \\ -5 \end{bmatrix} + (0.1) \begin{bmatrix} 24 \\ 12 \end{bmatrix} = \begin{bmatrix} 4 + 2.4 \\ -5 + 1.2 \end{bmatrix} = \begin{bmatrix} 6.4 \\ -3.8 \end{bmatrix}.$$

$\boxed{t^{(1)} = 0.1}$

$$\underline{y}^{(2)} = \underline{y}^{(1)} + h \cdot \underline{F}(t^{(1)}, \underline{y}^{(1)})$$

$$= \begin{bmatrix} 6.4 \\ -3.8 \end{bmatrix} + (0.1) \begin{bmatrix} 4\cdot(6.4) - 2\cdot(-3.8) - 4\cdot(0.1) - 2 \\ 3\cdot(6.4) + 5\cdot(0.1) \end{bmatrix}$$

$$= \begin{bmatrix} 6.4 \\ -3.8 \end{bmatrix} + (0.1) \begin{bmatrix} 30.8 \\ 19.7 \end{bmatrix} = \begin{bmatrix} 6.4 + 3.08 \\ -3.8 + 1.97 \end{bmatrix} = \begin{bmatrix} 9.48 \\ -1.83 \end{bmatrix}.$$

Task 2

2

```
%
% function [t, y] = MyOde(f, tspan, y0, N, events)
%
% Numerically solves the initial value problem
%
%    dy(t)/dt = f(t,y)
%        y(0) = y0
%
% using the Modified Euler time-stepping method.
%
% Input
%   f       handle to a Matlab dynamics function with calling sequence
%              dydt = f(t, y)
%   tspan   1x2 vector giving the start and end times, [start end]
%   y0      initial state of the system (as a column vector)
%   N       the number of time steps to take
%   events  handle to a Matlab event function with calling sequence
%              val = events(t, y)
%           The computation stops as soon as a negative value is
%           returned by the event function.
%
% Output
%   t       column vector holding time stamps
%   y       holds one state vector per row (corresponding
%           to the time stamps)
%
%   Note:
%       - t and y have the same number of rows.
%
%       - If the computation was stopped by the triggering of an event,
%         then the last row of t and y should correspond to the exact
%         time of the event.  That is, you should interpolate between
%         the last two points to extract the time and system state
%         corresponding to the event.
%
function [t, y] = MyOde(f, tspan, y0, N, events)

    h = ( tspan(2) - tspan(1) ) / N; % step size

    m = length(y0); % Number of state variables

    % Initialize output arrays
    t = zeros(N,1);
    y = zeros(N,m);

    y(1,:) = y0';
    t(1) = tspan(1);
    val = events(t(1), y(1,:));
    n = 1;

    while n <= N && val >= 0 % loop control (including event flag)

        t(n+1) = t(n) + h;
```

3

```
    f0 = f(t(n),y(n,:)')';
    y(n+1,:) = y(n,:) + h * f0; % take Euler step
    f1 = f(t(n+1),y(n+1,:)')'; % eval RHS at Euler step
    y(n+1,:) = y(n,:) + h/2 * ( f0 + f1 ); % modified Euler
    % Note: The dynamics function expects the state vector to
    % be a column vector. Its output is also a column vector.
    % Hence the transposes.

    n = n + 1;

    val = events(t(n), y(n,:)); % call to events function
end

if val < 0
    % Interpolate time and state of event
    val0 = events(t(n-1), y(n-1,:));
    alpha = val0 / (val0 - val);
    beta = - val / (val0 - val);
    t(n) = alpha*t(n) + beta*t(n-1);
    y(n,:) = alpha*y(n,:) + beta*y(n-1,:);
end

t = t(1:n);
y = y(1:n,:);
```

4

```
function dzdt = ballistics(t, z)
    % z(1) = x(t)
    % z(2) = y(t)
    % z(3) = x'(t)
    % z(4) = y'(t)
    g = 9.81;
    K = 0.2;
    dzdt = [ z(3) ; z(4) ; -K*z(3) ; -g - K*z(4) ];
```

(5)

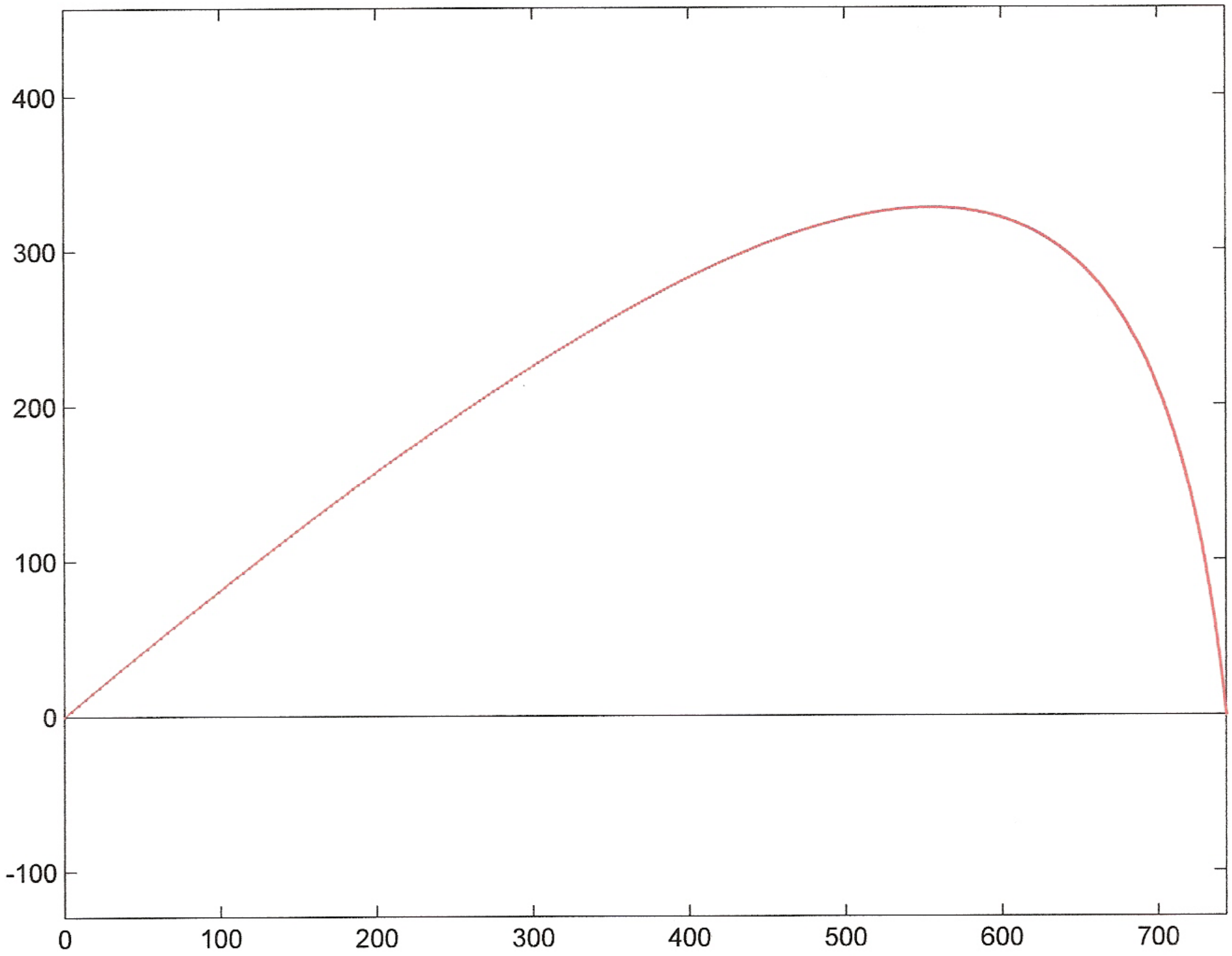
```
% ballistics_events.m
%
function value = ballistics_events(t, z)

    % the cannon ball hits the ground
    % (the ground is located at z(2) = y = 0).
    value = z(2);
```

6

```
% Cannon.m

theta = 60;     % Angle of initial velocity
S = 200;        % Initial speed
tspan = [0 30]; % Set start and end times for computation

% Set up initial state
yStart = [0;0;                     % initial position (0,0)
          S*cos(theta/180*pi);     % x velocity
          S*sin(theta/180*pi)];    % y velocity


[t,y] = MyOde(@ballistics, tspan, yStart, 1000, @ballistics_events);


plot([0 y(end,1)], [0 0], 'k-', y(:,1), y(:,2), 'r.-', 'MarkerSize', 3);
axis equal;
title(['\theta = ' num2str(theta) '^\circ: Cannon ball landed at ' num2str(y(end,1))↙
'm'], 'FontSize', 18);
```

A sample trajectory.

$\theta = 40^{\circ}$: Cannon ball landed at 743.2105m



Furthest firing distance