MEMS 1060: Homework #8

Due on April 8, 2021 at 6:00pm

Professor Sammak Th 6:00PM

Made using LaTeX/Inkscape. Source files available upon request.

Shane Riley

Problem 1

Given the following system of two first-order ODEs, time span $0 < t < 1.2$, and initial conditions $x(0) = 1$ and $y(0) = 0.5$, evaluate the initial-value problem using 2nd order Runge Kutta:

$$\frac{dx}{dt} = xt - y = F_1$$
$$\frac{dy}{dt} = yt + x = F_2$$

Solution

To employ RK, we must first pick our constants. We will follow Heun's method by pinning $c_2 = 0.75$, fixing the other values as follows:

$$c_1 = 0.25$$
$$c_2 = 0.75$$
$$a = \frac{2}{3}$$
$$b = \frac{2}{3}$$

Since the calculation of K_{x1}, K_{y1}, K_{x2}, and K_{y2} require use of F_1 and F_2, it is natural to employ anonymous functions in to employ the method. With the functions described, the script simply calculates the K-values in order and uses them to find x and y for the following timestep. For this analysis, the step size is chosen arbitrarily as $h = 0.01$.

Using Heun's method and a linear timestep, the solutions are determined and plotted. The equations are also put through **ode45**, MATLAB's stock RK solver, for comparison.

Using Heun's method, the solution to the initial value problem is plotted.

Problem 2

Given the first order ODE, use predictor-corrector (Euler's explicit and Adams-Moulton) to solve the initial value problem. $0 < t < t$ and $x(0) = 2000$.

$$\frac{dx}{dt} = -0.8x^{1.5} + 20{,}000(1 - e^{-3t}) = F_1(t_i, x_i)$$

Solution

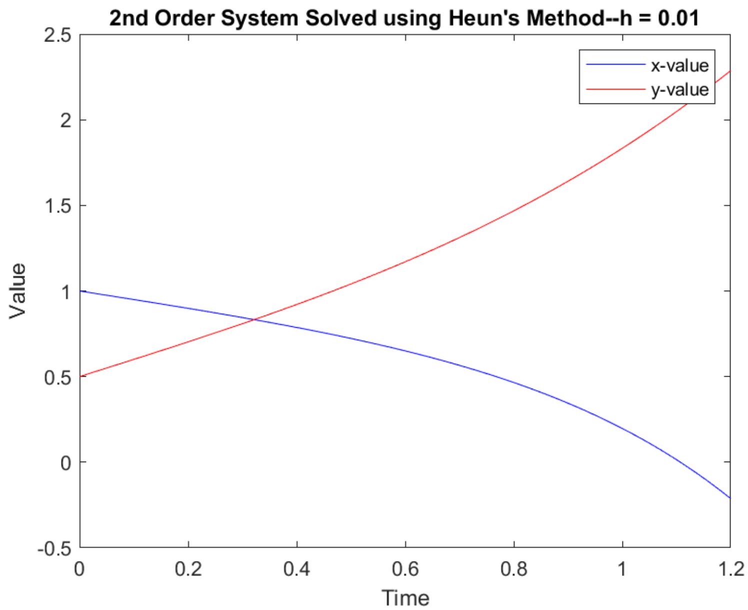
The first guess for x_{i+1} is calculated using Euler's Explicit method, which is trivial:

$$x^1_{i+1} = x_i + h * F_1(t_i, x_i)$$

The first two steps in the time interval cannot be iterated since there are not enough previous times yet. The iteration uses Adams-Moulton, which uses a weighted average of steps:

Problem 2 continued on next page. . .

Figure 1: Problem 1 Heun's.



$$x_{i+1}^k = x_i + \frac{h}{12}(5F_1(t_{i+1}, x_{i+1}^k) + 8F_1(t_i, x_i) - F_1(t_{i-1}, x_{i-1}))$$

Once the relative difference between steps is less than $\epsilon = 10^{-4}$, the value is stored and the next timestep begins. Using this method (and **ode45** for comparison), the solution is plotted using $h = 0.01$:

Using Explicit Euler's and Adams-Moulton as predictor-corrector, the solution to the initial value problem is plotted.

Problem 3

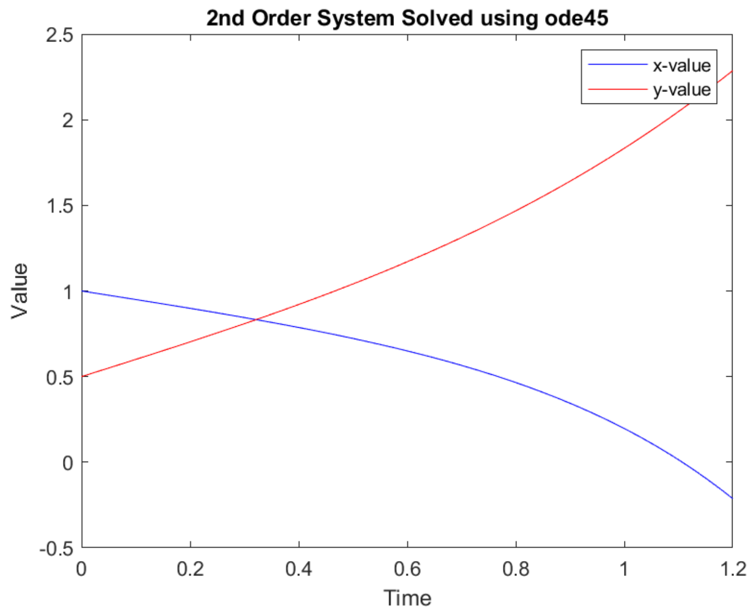
Given the equations of motion and time-variable mass of a rocket, find the position, velocity, and acceleration of the rocket in the interval $0 < t < 3[s]$.

$$\frac{w}{g}\frac{d^2y}{dt^2} = T - w - D$$
$$D(v) = 0.008gv^2$$
$$g = 32.2[ft/s^2]$$
$$w(t) = 3000 - 80t[lb]$$
$$T = 7000[lb]$$

Solution

The equation of motion for the rocket is second order. In order to solve it, we must manipulate it into a system of two first order ODEs. We do this by defining a state vector s, and finding the time derivative

Figure 2: Problem 1 Stock.



of state $\dot{s}$. We can know the entire state of the system using position and velocity, so those will be our components. x, v, and a represent position, velocity, and acceleration respectively.

$$s = \begin{bmatrix} x \\ v \end{bmatrix}$$

$$\dot{s} = \begin{bmatrix} v \\ a \end{bmatrix}$$

The key to finishing our setup is to represent the components of $\dot{s}$ in terms of s and time. This first component is trivial, since velocity is expressed explicitly in the state vector. By rearranging our equation of motion, we find an expression for acceleration:

$$a = (T - w(t) - D(v)) * \frac{g}{w(t)}$$

$$\dot{s} = \begin{bmatrix} s_2 \\ (T - w(t) - D(s_2)) * \frac{g}{w(t)} \end{bmatrix}$$

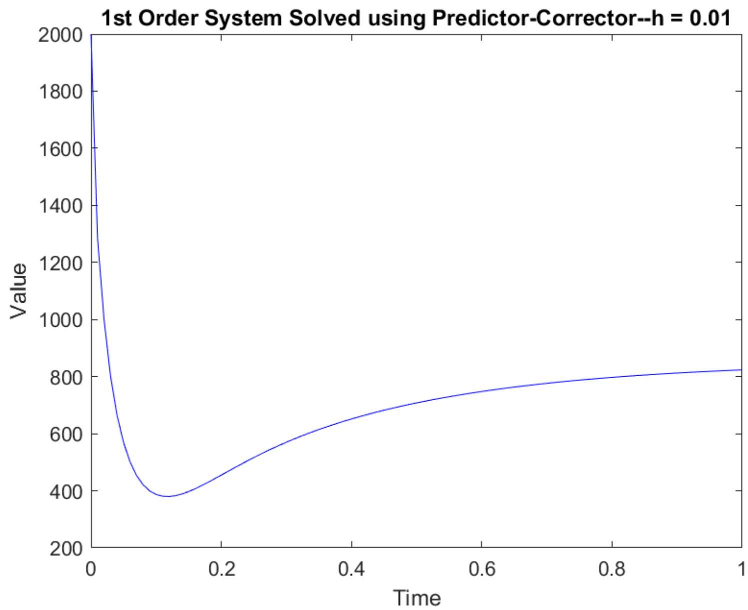
With our equations and initial conditions, we will simply employ Euler's Explicit with $h = 0.005[s]$ to solve. A more sophisticated method could also be used with the same setup. See the MATLAB script for implementation.

$$s_{i+1} = s_i + \dot{s_i}$$

With position and velocity found for every timestep, all of the timesteps are iterated again to find the acceleration using the expression already found. This is appended to the existing results and plotted.

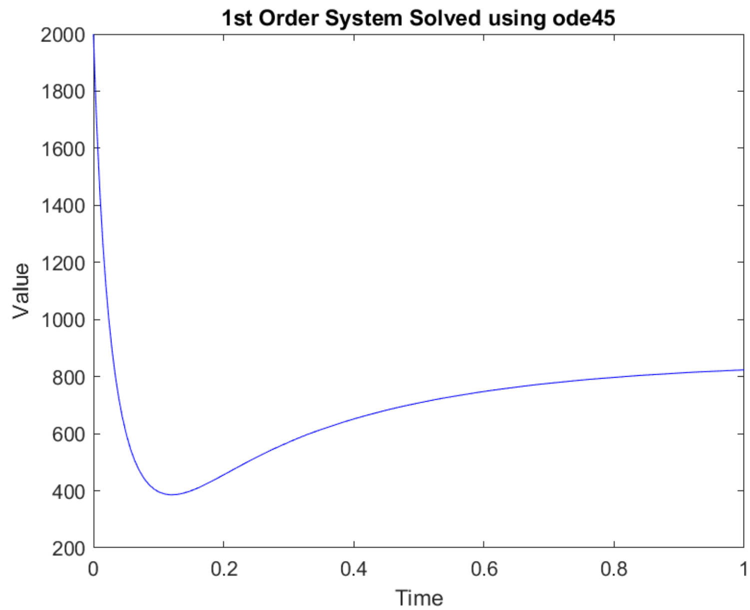
Using the equations of motion and Euler's Explicit method, the kinematics of the rocket are expressed.

Figure 3: Problem 2 P-C.



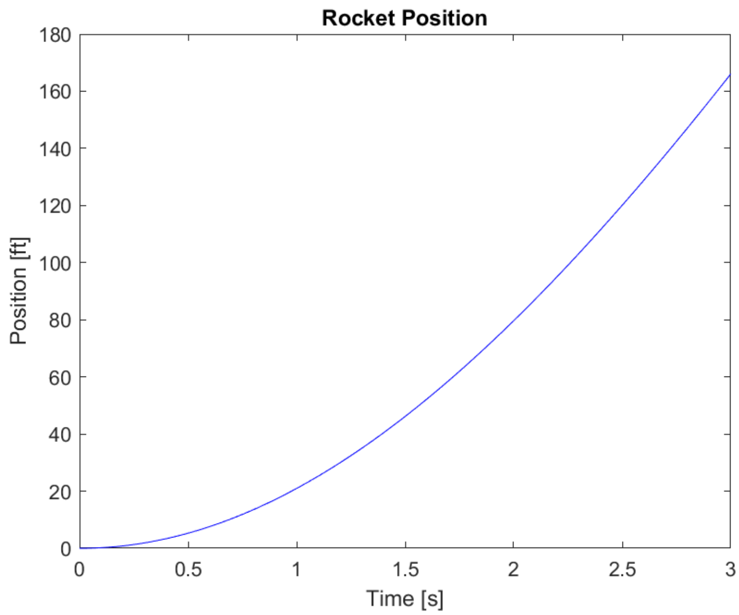

```matlab
%% MEMS 1060 Homework 8
% Author: Shane Riley
% Date: 4/8/2021
format long
clear
clc
close all
%% Problem 1
% Given dx/dt, dy/dt, tspan, x(0), and y(0), solve using 2nd order RK
disp(" ");
disp("Problem 1");

% Eq's and IC
dx = @(t,x,y) x*t - y;
dy = @(t,x,y) y*t + x;

% Specify step size and presize vectors
h = 0.01;
t = (0:h:1.2)';
x = zeros(length(t),1);
y = zeros(length(t),1);

% Initial value
x(1) = 1;
y(1) = 0.5;

% Specify constants
c2 = 0.75;  % Use Heun's method
```

Figure 4: Problem 2 Stock.



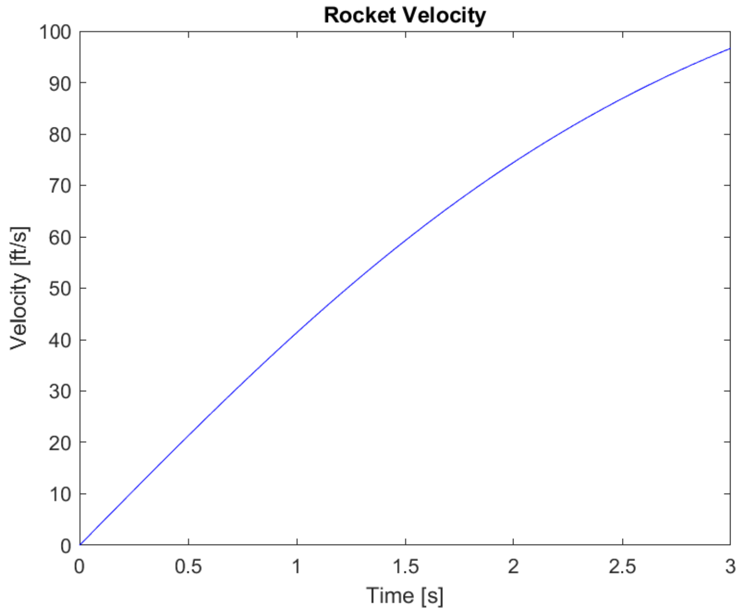

```
c1 = 1 - c2;
a = 0.5/c2;
b = 0.5/c2;

% Specify K values as functions
Kx1 = @(t,x,y) dx(t,x,y);
Ky1 = @(t,x,y) dy(t,x,y);
Kx2 = @(t,x,y,Kx1,Ky1) dx(t + a*h, x + Kx1(t,x,y)*b*h, y + Ky1(t,x,y)*b*h);
Ky2 = @(t,x,y,Kx1,Ky1) dy(t + a*h, x + Kx1(t,x,y)*b*h, y + Ky1(t,x,y)*b*h);
xnew = @(t,x,y) x + ( c1*Kx1(t,x,y) + c2*Kx2(t,x,y,Kx1,Ky1) )*h;
ynew = @(t,x,y) y + ( c1*Ky1(t,x,y) + c2*Ky2(t,x,y,Kx1,Ky1) )*h;

% Iterate
for i=2:length(t)
    x(i) = xnew(t(i-1),x(i-1),y(i-1));
    y(i) = ynew(t(i-1),x(i-1),y(i-1));
end

% Plot
figure(1);
plot(t,x,'b-');
hold on
plot(t,y,'r-');
title("2nd Order System Solved using Heun's Method--h = 0.01");
ylabel("Value");
xlabel("Time");
legend("x-value", "y-value");
print("images/figure1",'-dpng');
```

Figure 5: Problem 3–Rocket Position.



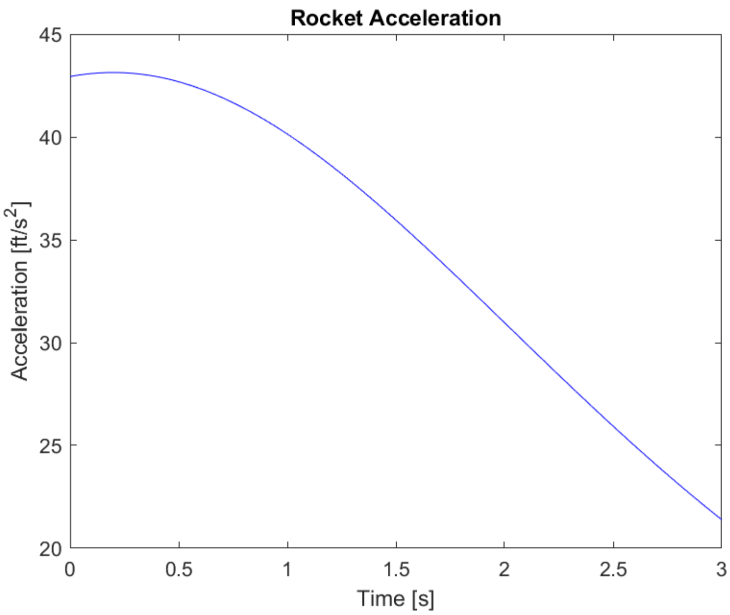

```

% Compare to stock RK solver
[t,v] = ode45((@(t,v) [dx(t,v(1),v(2));dy(t,v(1),v(2))]), [0 1.2], [1; 0.5]);
figure(2);
plot(t,v(:,1),'b-');
hold on
plot(t,v(:,2),'r-');
title("2nd Order System Solved using ode45");
ylabel("Value");
xlabel("Time");
legend("x-value", "y-value");
print("images/figure2",'-dpng');

%% Problem 2
% Given dx/dt, tspan, x(0), use predictor-corrector (Euler's explicit with
% implicit third-order Adams-Moulton) to solve.
disp(" ");
disp("Problem 2");

% Equation
dx = @(t,x) -0.8 * x^(1.5) + 20000 * (1 - exp(-3*t));

% Step size and time vector
h = 0.01;
t = 0:h:1;
x = zeros(length(t),1);
x(1) = 2000;
eps = 10^(-4);
```

Figure 6: Problem 3–Rocket Velocity.




```
% Functions
x_predict = @(t,x) x + h*dx(t,x);
x_correct = @(t,x) x(2) + ((h/12)*(5*dx(t(1),x(1)) + 8*dx(t(2),x(2)) - dx(t(3)
    ,x(3))));
checkexp = @(x1,x2) abs((x2-x1)/x2);

% Get 1st value only using prediction (no x_{i-1} for correction yet)
x(2) = x_predict(t(1),x(1));

for i=3:length(t)

    % Predict
    xn = x_predict(t(i-1),x(i-1));

    xs = [xn, x(i-1), x(i-2)];
    ts = [t(i), t(i-1), t(i-2)];

    % Run tries
    while true
        % Run correction
        xnn = x_correct(ts, xs);

        % Check convergence
        if (checkexp(xnn, xs(1)) < eps), break; end

        % Else, rebuild
        xs(1) = xnn;
```

Figure 7: Problem 3–Rocket Acceleration.




```
    end

    % Store value
    x(i) = xnn;
end

% Plot
figure(3);
plot(t,x,'b-');
title("1st Order System Solved using Predictor-Corrector--h = 0.01");
ylabel("Value");
xlabel("Time");
print("images/figure3",'-dpng');

% Compare to stock RK solver
[t,x] = ode45((@(t,x) dx(t,x) ), [0 1], 2000);
figure(4);
plot(t,x,'b-');
title("1st Order System Solved using ode45");
ylabel("Value");
xlabel("Time");
print("images/figure4",'-dpng');

%% Problem 3
% Use the EOM for a rocket to determine acceleration, velocity, and
% position
disp(" ");
disp("Problem 3");
```

```
% Express terms
w = @(t) 3000 - 80*t;  % [lb]
g = 32.2;  % [ft/s^2]
T = 7000;  % [lb]
D = @(v) 0.008*g*(v)^2;  % [lb]

% get xdot and vdot into a vector s, see report
vel = @(t,s) s(2);
accel = @(t,s) (g/w(t))*(T - w(t) - D(s(2)));

diffeq = @(t,s) [...
    vel(t,s)
    accel(t,s)
]';

% Time span and initial condition
tspan = [0 3];
s_init = [0 0];

% Step size and t vector
h = 0.005;
t = (tspan(1):h:tspan(end))';

% Solve using Euler's Explicit
s_new = @(t,s) s + diffeq(t,s).*h;
s = zeros(length(t), length(s_init));
s(1,:) = s_init;
for i=2:length(t)
    % Get state row
    s(i,:) = s_new(t,s(i-1,:));
end

% Add acceleration
accels = zeros(length(t),1);
for i=1:length(t)
    accels(i) = accel(t(i), s(i,:));
end
s = [s accels];

% Plots
figure(5)
plot(t,s(:,1), 'b-');
xlabel("Time [s]");
ylabel("Position [ft]");
title("Rocket Position");
print("images/figure5",'-dpng');

figure(6)
```

```
plot(t,s(:,2), 'b-');
xlabel("Time [s]");
ylabel("Velocity [ft/s]");
title("Rocket Velocity");
print("images/figure6",'-dpng');

figure(7)
plot(t,s(:,3), 'b-');
xlabel("Time [s]");
ylabel("Acceleration [ft/s^2]");
title("Rocket Acceleration");
print("images/figure7",'-dpng');
```