Abelardo Riojas
ISC 4220C Spring 2020
Prof. Shanbhag

Lab 9 Report

1. The motion of a mass m attached to a spring of stiffness k, moving on a rough surface with friction coefficient μ may be described by the differential equation:

$$m\frac{d^2x}{dt^2} + \mu\frac{dx}{dt} + kx = 0; \quad \text{initial conditions: } x(0) = 1, \frac{dx(0)}{dt} = 0.$$

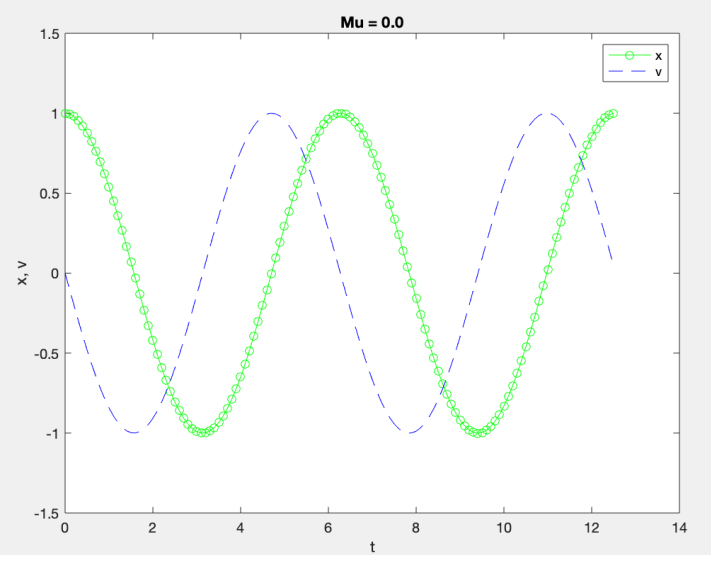
   a. Is the ODE linear or nonlinear? What is the order of the ODE?

   The ODE is linear because we have no $y^{(n)}$ terms multiplied by each other. The order of the ODE is second order because the highest derivative of x is 2.

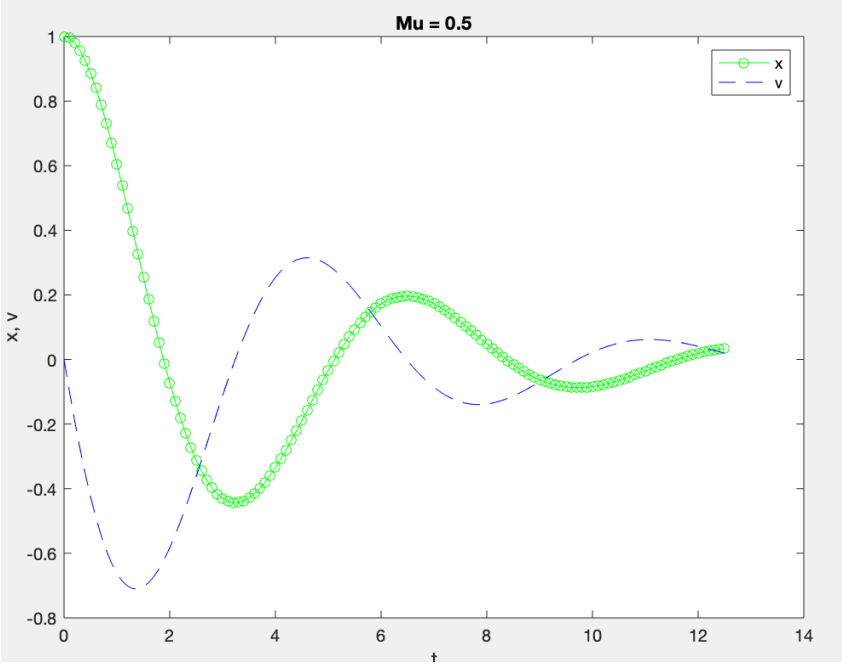
   b. Rewrite the ODE as a system of two first order ODEs. Also rewrite the initial conditions, as required.

   If v = dx/dt, then
   d/dt [x;v]' = [v; (-kx - mu*v)/m]'
   and x(0) = 1 and v(0) = 0.

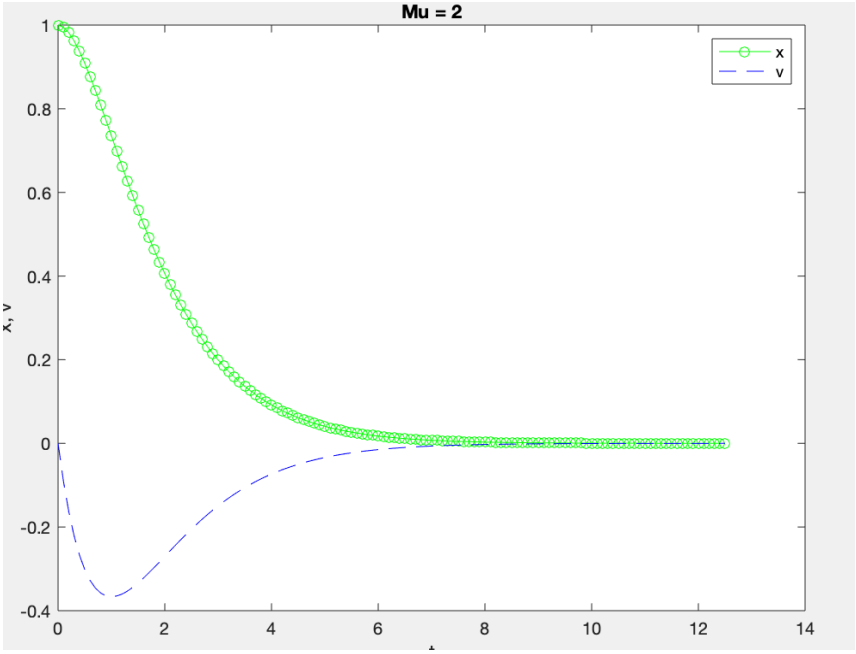
   c. Use midpoint method with step-size h = 0.1, and solve for x(t) between t = 0 and t = 4π with, m = k = 1, and
      i. μ = 0.0



ii. $\mu = .5$



iii. $\mu = 2$




```
% Slice up time, and preallocate arrays
t0 = 0.; Tmax = 4*pi; dt = 0.1;
npts = floor(Tmax/dt + 1);
t = zeros(npts, 1);
x = zeros(npts, 1);
v = zeros(npts, 1);
% Initial conditions
```

```
t(1) = 0; x(1) = 1; v(1) = 0.;
% Time-Stepping
for i = 2:npts
yold = [x(i-1); v(i-1)];
ynew = midpoint(@func, t(i-1), yold, dt); % function handle
here!
t(i) = t(i-1) + dt;
x(i) = ynew(1);
v(i) = ynew(2);
end

plot(t, x, 'go-'); hold on;
plot(t, v, 'b--');
legend('x','v')
xlabel('t')
ylabel('x, v')
title('Mu = 2')
hold off;

function yp = func(t,y)
k = 1;
m = 1;
mu = 2;
yp = zeros(size(y));
x = y(1);
v = y(2);
yp(1) = v;
yp(2) = (-k*x - (mu*v))/m;
end

function ynew = midpoint(f, told, yold, h)
ymid = yold + h/2 * f(told, yold);
ynew = yold + h * f(told + h/2, ymid);
end
```

d. Use the test equation method to show that the region of stability for the midpoint method is given by:

$$\frac{y_n}{y_{n-1}} = \left|1 + h\lambda + \frac{(h\lambda)^2}{2}\right| \leq 1.$$

Midpoint rule:

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

Using the test equation y' = λy

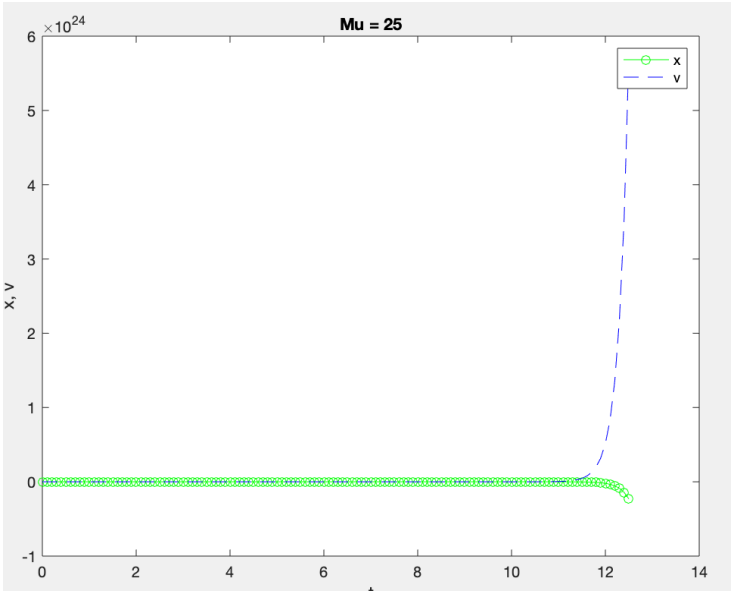
$y_n = y_{n-1} + h\lambda(y_n + \frac{h}{2}\lambda y_{n-1})$ which simplifies to…

$y_n = y_{n-1} + h\lambda y_{n-1} + \frac{(\lambda h)^2}{2} y_{n-1}$ therefore,

$$\frac{y_n}{y_{n-1}} = 1 + h\lambda + \frac{(h\lambda)^2}{2}$$

LHS must not be bigger than 1 so abs(RHS) <= 1.

e. Compute x(t) as in part (iii) with μ = 25. Interpret your observations using the stability criterion above.



The solution is not stable since the Mu term acts like the test equation and lies outside the region of stability.
abs(1 + .1*25 + ((.1*25)^2/2)) = 6.62500 which is greater than 1.

Consider the following ODE:

$$\frac{dy}{dt} - ty - t = 0,$$

with $y(t = 0) = 0$.

2.

a. Explain why (or why not) the following labels apply to this problem: first ordered, homogeneous, linear, implicit, initial value problem.

i. First order because the highest order of $y^{(n)}$ is 1.
ii. Homogeneous because RHS = 0.
iii. Linear because no $y^{(n)}y^{(m)}$ terms exist.

iv. Not implicit because y' can be written in terms of only y and t.

v. IVP because the first value is given to you, namely, y(0) = 0.

b. This problem can we rewritten as y' = f(t, y) = t(1 + y). Carry out a single step of the backward Euler method with step-size h = 0.1 to find y(t = 0.1).

Backward Euler: $y_n = y_{n-1} + hf_n$ where $f_n = (y_n - y_{n-1})/h$

Hence

$y_1 = y_0 + hf(t_1, y_1)$ therefore

$y_1 = 0 + .1(.1(1 + y_1))$ which simplifies to

$y_1 = .01/.99$

c. The exact solution to the problem is y(t) = exp(t^2/2) - 1. Verify that this satisfies the ODE and the initial condition.

$y' = t * e^{t^2/2}$ therefore

$te^{t^2/2} - te^{t^2/2} + t - t = 0$

Initial condition: $e^0 - 1 = 0$

d. Find the local truncation error for y(t = 0.1) computed by backward Euler above.

True value: exp((.1^2)/2) - 1 = 0.00501252086

Backward Euler value: 0.0101010101

Error = abs(.00501252086 - .0101010101) = 0.00508848924

e. True or False: Step-size required for the accuracy of a numerical method for solving ODEs is always smaller than that required for stability.

False. Sometimes small step sizes of h can still not compute an accurate answer even with a method that allows for large h values for stability.

3. Consider the model of flame propagation, described by Larry Shampine and Cleve Moler: "If you light a match, the ball of flame grows rapidly until it reaches a critical size. Then it remains at that size because the amount of oxygen being consumed by the combustion in the interior of the ball balances the amount available through the surface".

One can write down an approximate model for the radius r of the ball as:

$$\frac{dr}{dt} = r^2 - r^3$$

a. Set $\delta = 0.1$. Use the Matlab program ode45 to solve the ODE with a relative error of $10-4$. Hint: use opts = odeset('RelTol',1.0e-4). Find the amount of time it takes to solve the problem

```
tic
opts = odeset('RelTol',1.0e-4);
del = .1;
tspan = [0 2/del];
r0 = del;
[t,r] = ode45(@(t,r) r^2 - r^3, tspan, r0,opts);
toc
```

```
Elapsed time is 0.074395 seconds.
```

b. Try to repeat the above with $\delta = 0.0001$ or $\delta = 0.00001$. Describe and explain what you observe. You may have to zoom in after $t = 1/\delta$ to figure out. Find the amount of time it takes to solve the problem.

```
tic
opts = odeset('RelTol',1.0e-4);
del = .0001;
tspan = [0 2/del];
r0 = del;
[t,r] = ode45(@(t,r) r^2 - r^3, tspan, r0,opts);
toc
```

```
Elapsed time is 0.147169 seconds.
```

The problem took twice as long because the interval was increased by a factor of 100.

c. Repeat with the Matlab program ode23. Find the amount of time it takes to solve the problem.

```
tic
opts = odeset('RelTol',1.0e-4);
del = .1;
tspan = [0 2/del];
r0 = del;
[t,r] = ode23(@(t,r) r^2 - r^3, tspan, r0,opts);
plot(t,r)
toc
```

```
Elapsed time is 0.092349 seconds.
```

Consider the following BVP over the domain $0 \leq t \leq 1$:

$$y'' = y - t, \qquad y(t=0) = y(t=1) = 0.$$

The exact solution to the problem is given by:

4.

$$y(t) = t + \left(\frac{e}{e^2 - 1}\right)(e^{-t} - e^{t})$$

We want to use the "shooting method" to solve the BVP.

a. Verify the exact solution.

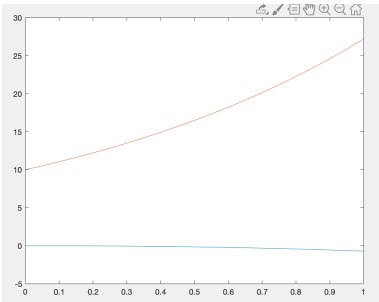
Let k = exp(1)/(exp(2)-1)
y(t) = t + kexp(-t) - kexp(t).
y'(t) = -kexp(-t) - kexp(t)
y''(t) = kexp(-t) - kexp(t)
Therefore…
k(exp(-t) - exp(t)) - y + t = 0. v
If y = t + k(exp(-t) - exp(t)) then
k(exp(-t) - exp(t)) - t -k(exp(-t) - exp(t)) + t = 0

b. Rewrite the second order ODE as a system of two first order IVPs.
If v = dy/dt, then
d/dt [y;v]' = [v; y-t]'

c. Use Matlab's intrinsic IVP solver ode45 to solve the IVPs with a relative tolerance of 10−4.

```
opts = odeset('RelTol',1.0e-4);
tspan = [0 1];
y0 = 0;
z0 = 10; %guess
[t,y] = ode45(@(t,y,z) y-t, tspan, y0,opts);
plot(t,y)
hold on
[t,z] = ode45(@(t,z) z, tspan, z0,opts);
plot(t,z)
```

d. Plot the exact and computed solutions over the domain $0 \leq t \leq 1$.


30
25
20
15
10
5
0
-5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1