Table of Contents

Homework 7

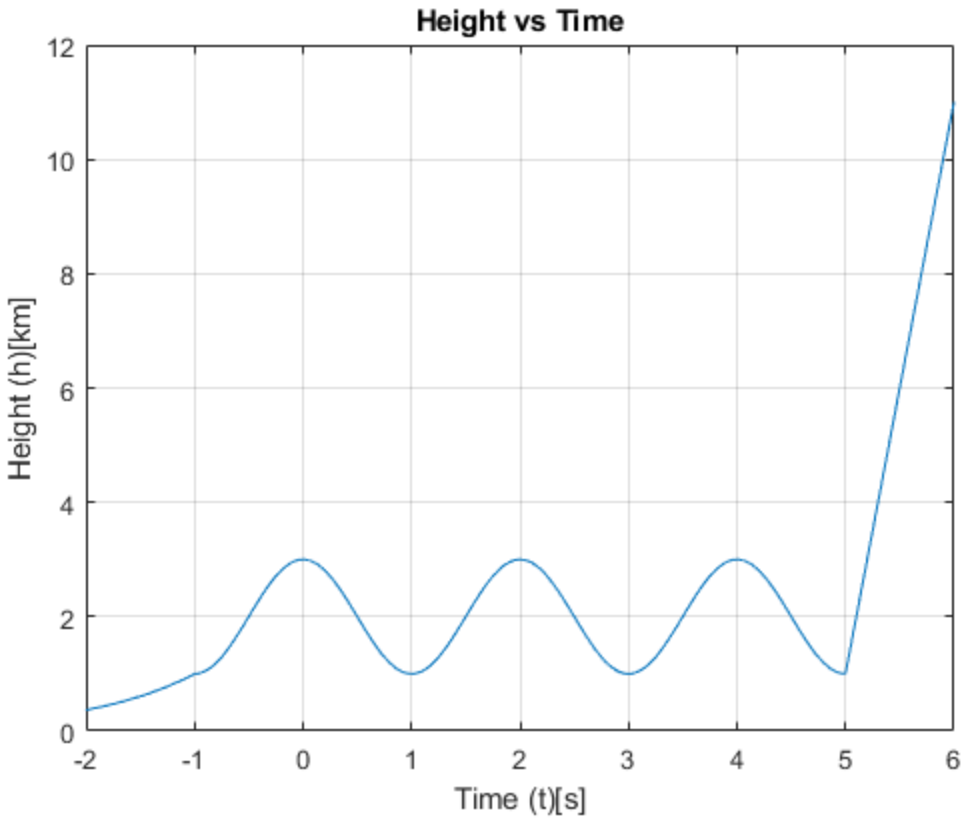
ENGR 133-003 Created by Sean DeBarr 3/08/2019

```
clear
clc
```

Problem 22

```
clear

disp("**********************" + newline + "Problem 22" + newline);

% declare x interval
t = [-2:0.01:6];

% preallocated y for 78.84% faster proccessing speed
h = zeros(1, length(t));

% calculate all values of y for given interval
for (i = 1:length(t))
    h(i) = prob16(t(i));
end

% plot the graph of the function
fprintf("Please refer to external plot\n\n");
plot(t,h);
grid on;
xlabel("Time (t)[s]");
ylabel("Height (h)[km]");
title("Height vs Time");
```

```
**********************
Problem 22

Please refer to external plot
```



Problem 24

```
clear

% Upon review of this problem and using tic toc, using a for loop is
% approximately 100x slower than not using one

disp("**********************" + newline + "Problem 24" + newline);

% functions for x(t) and y(t)
xT = @(t) (5 * t) - 10;
yT = @(t) (25 * t.^2) - (120 * t) + 144;

% declare t interval
t = [0:0.01:4];

% preallocate variables
d1 = zeros(1, length(t));
x1 = zeros(1, length(t));
y1 = zeros(1, length(t));

%
 ********************************************************************************
% Part a
disp("Part a:" + newline);
```

```
fprintf("With for loop\n");

for (i = 1:length(t))
    % calculate x and y with respect to time
    x1(i) = xT(t(i));
    y1(i) = yT(t(i));

    % calculate the distance from origin
    d1(i) = sqrt(x1(i).^2 + y1(i).^2);
end

% calculate minimum distance
minDist1 = min(d1);

% calculate the time closest to origin
tClos1 = t(d1 == minDist1);

% display results
fprintf("\nThe minimum distance to the origin is %g", minDist1);
disp(newline);
fprintf("The time at which the object is closest to the origin is %g\n
\n", tClos1);

%
 **************************************************************************
% Part b
disp("Part b:" + newline);
fprintf("Without for loop\n");

% calculate x and y with respect to time
x2 = xT(t);
y2  = yT(t);

% calculate the distance from origin
d2 = sqrt(x2.^2 + y2.^2);

% calculate minimum distance
minDist2 = min(d2);

% calculate the time closest to origin
tClos2 = t(d2 == minDist2);

% display results
fprintf("\nThe minimum distance to the origin is %g", minDist2);
disp(newline);
fprintf("The time at which the object is closest to the origin is %g\n
\n", tClos2);

**********************
Problem 24

Part a:

With for loop
```

The minimum distance to the origin is 1.35813

The time at which the object is closest to the origin is 2.23

Part b:

Without for loop

The minimum distance to the origin is 1.35813

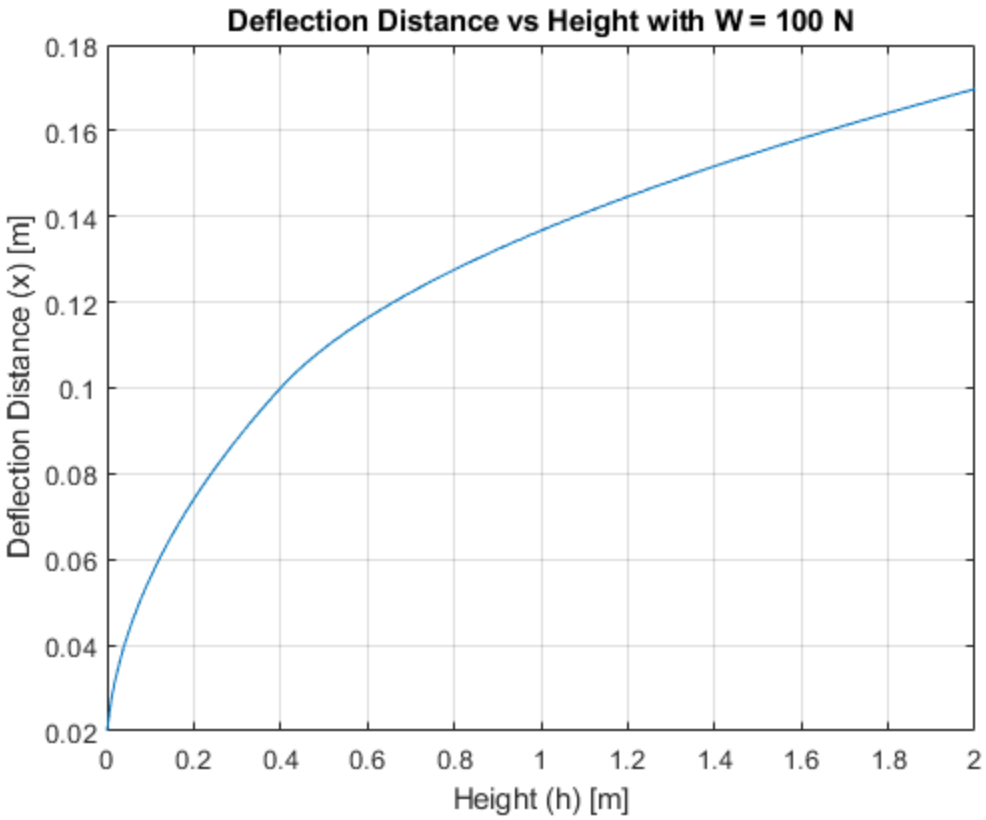
The time at which the object is closest to the origin is 2.23

Problem 26

```
clear

disp("**********************" + newline + "Problem 26" + newline);

%{
function [x] = spring_compress(k1, k2, d, W, h)
% computes maximum compression due to falling weight
x = zeros(1, length(h));
for (i = 1:length(h))
    if d > ((W(i) + sqrt((W(i)^2)+(2*k1*W(i).*h(i))))/k1)
        x(i) = (W(i) + sqrt((W(i)^2)+(2*k1*W(i).*h(i))))/k1;
    else
        pos_root = roots([(k1+(2*k2)),-((4*k2*d)+(2*W(i))),
((2*k2*(d^2))-(2*W(i).*h(i)))]);
        x(i) = max(pos_root);
    end
end
end
%}

% declare known values
k1 = 10^4;
k2 = 1.5*10^4;
d = 0.1;

%
 **************************************************************************
% Part a
disp("Part a:" + newline);

% declare test cases
TCW = [100, 2000];
TCh = [0.5, 0.5];

% run test cases
TC1 = spring_compress(k1, k2, d, TCW(1), TCh(1));
TC2 = spring_compress(k1, k2, d, TCW(2), TCh(2));
```

```
% display results
fprintf("Test case #1 (W = %g N) (h = %g m), x = %g m.\n\n", TCW(1),
 TCh(1), TC1);
fprintf("Test case #2 (W = %g N) (h = %g m), x = %g m.\n\n", TCW(2),
 TCh(2), TC2);

%
 ************************************************************************
% Part b
disp("Part b:" + newline);

% declare interval h for height and W for weight
h = 0:0.01:2;
W = repmat(100, 1, length(h));

% calculate deflection distance
x = spring_compress(k1, k2, d, W, h);

% plot the function
fprintf("Please refer to external plot\n\n");
plot(h,x);
grid on;
xlabel("Height (h) [m]");
ylabel("Deflection Distance (x) [m]");
title("Deflection Distance vs Height with W = 100 N");
```

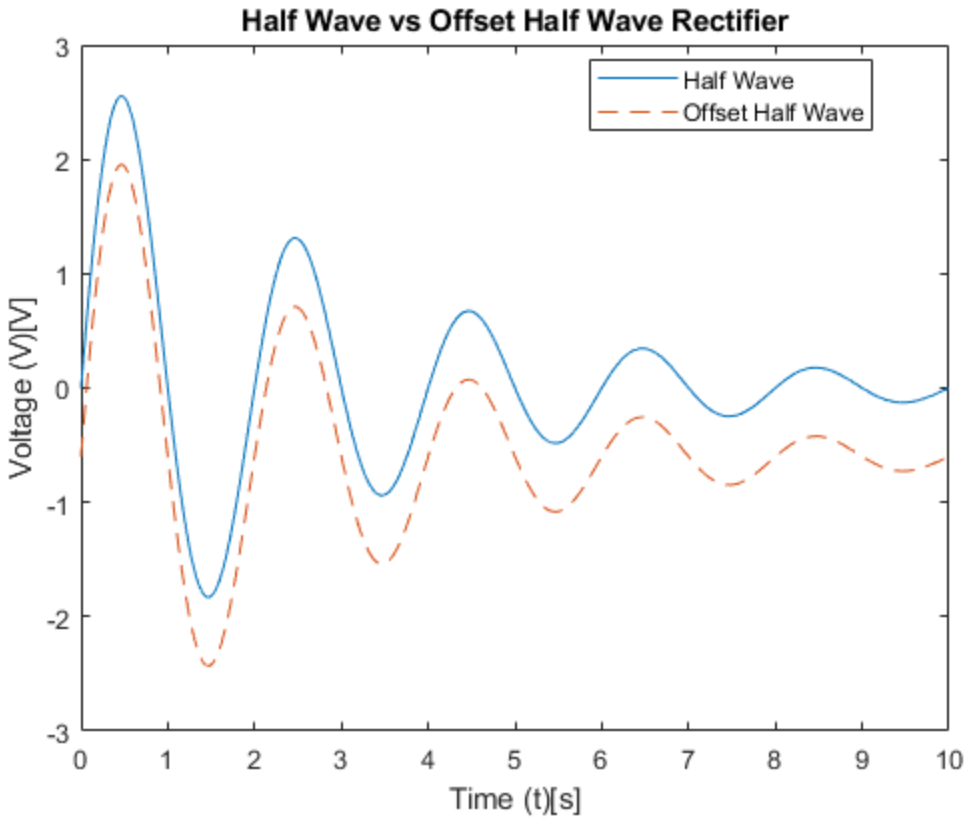
```
*********************
Problem 26

Part a:

Test case #1 (W = 100 N) (h = 0.5 m), x = 0.109221 m.

Test case #2 (W = 2000 N) (h = 0.5 m), x = 0.366091 m.

Part b:

Please refer to external plot
```



Problem 28

```
clear

disp("**********************" + newline + "Problem 28" + newline);

% declare time array
t = 0:0.01:10;

% calculate supply voltage
Vs = 3 .* exp(-t ./ 3) .* sin(pi .* t);

%
 ************************************************************************
% Part a
disp("Part a:" + newline);

% preallocate variable
vL1 = zeros(1, length(Vs));

% calculate voltage
for (i = 1:length(Vs))
if Vs(Vs > 0)
    vL1(i) = Vs(i);
else
```

```
    vL1(i) = 0;
end
end

% plot function
plot(t, vL1);
hold on

%
 ************************************************************************
% Part b
disp("Part b:" + newline);

% preallocate variable
vL2 = zeros(1, length(Vs));

% calculate voltage
for (i = 1:length(Vs))
if Vs(Vs >  0.6)
    vL2(i) = Vs(i) - 0.6;
else
    vL2(i) = 0;
end
end

% plot the function and label it
plot(t, vL2, '--');
xlabel("Time (t)[s]");
ylabel("Voltage (V)[V]");
title("Half Wave vs Offset Half Wave Rectifier");
legend({"Half Wave","Offset Half Wave"},"Location","best");
disp("Please refer to external plot");
disp(newline);

**********
Problem 28

Part a:

Part b:

Please refer to external plot
```



Problem 42

```
clear

disp("**********************" + newline + "Problem 42" + newline);

% declare coefficient of friction array
cF = [0.2, 0.35, 0.4, 0.7];

% get user input of weight
% loads a default value if encounters and error
try
    W = input('Enter a weight value: ');
catch
    load matlab.mat
    fprintf("\nNo user.. Loading defaults...\n");
    fprintf('Weight = %g\n', W);
end

% display options
fprintf("\nOptions\n");
fprintf("1. Metal on Metal\n");
fprintf("2. Wood on Wood\n");
fprintf("3. Metal on Wood\n");
fprintf("4. Rubber on Concrete\n");
```

```
% get user input of material
% loads a default value if encounters and error
try
    material = input('Please enter option number: ');
catch
    load matlab.mat
    fprintf("\nNo user.. Loading defaults...\n");
    fprintf('Option = %g\n', material);
end

% computes force depending on material
switch material
    case 1
        F = cF(1) * W;
    case 2
        F = cF(2) * W;
    case 3
        F = cF(3) * W;
    case 4
        F = cF(4) * W;
    otherwise
        fprintf("...Invalid choice given...");
end

% display results
fprintf("\nThe amount of force required to move %gkg\n", W);
fprintf("on that surface is %gN.\n", F);
```

```
**********
Problem 42


No user.. Loading defaults...
Weight = 100

Options
1. Metal on Metal
2. Wood on Wood
3. Metal on Wood
4. Rubber on Concrete

No user.. Loading defaults...
Option = 1

The amount of force required to move 100kg
on that surface is 20N.
```

Published with MATLAB® R2017b