% CalcResistance.m
% Patrick Utz, 3/2/18, 8.1

% Problem: The resistance R in ohms of a conductor is given by R = V/I,
% where V is the voltage potential in volts and I is the current in
% amperes. The power consumed by the resistor in watts is given by P = VI.
% Write a MATLAB program with the following files,
% 1. A function named readVI that prompts the user to input the potential
% and the current
% 2. A function named calcRP that takes in two input arguments as potential
% and current, and returns two output arguments as resistance and power.
% 3. A function named printRP that takes in two input arguments as
% resistance and power, and prints out the values of the resistance
% and power to the screen.
% 4. A script file that will call the above three functions.
% Run a test of your program using input values of V=5volts
% and I=0.5ampere. Attach the results.

% Variables: potential = the voltage potential, current = the current,
% resistance = the resistance, power = the power

```
clear
[potential, current] = readVI;
[resistance, power] = calcRP(potential, current);
printRP(resistance, power);
```

```
function [potential, current] = readVI
% readVI prompts the user to input the potential and the current
% Format of call: readVI()
% Returns the inputted potential and the current

potential = input('Hello! Please enter the potential: ');
current = input('Hello! Please enter the current: ');
end
```

```
function [resistance, power] = calcRP(potential, current)
% calcRP prompts the user to input the potential and the current
% Format of call: calcRP(potential, current)
% Returns the proper resistance and power

resistance = potential/current;
power = potential*current;
end
```

```
function printRP(resistance, power)
% printRP prompts the user to input the resistance and power
% Format of call: printRP(resistance, power)
% Prints out the values of the resistance and power to the screen

fprintf('The resistance is %f\n', resistance);
fprintf('The power is %f\n', power);
end
```

Results:

```
>> CalcResistance
Hello! Please enter the potential: 5
Hello! Please enter the current: .5
The resistance is 10.000000
The power is 2.500000
```

% CalcPosition.m
% Patrick Utz, 3/2/18, 8.2

% Problem: Write a program to calculate the position of a projectile
% at a given time t. For an initial velocity v0 and angle of departure
% theta, the position is given by x and y coordinates as follows (note:
% the gravity constant g is 9.81m/s2):

$$x = v_0 \cos(\theta_0)t \; ;$$

$$y = v_0 \sin(\theta_0)t - \frac{1}{2}gt^2$$

% Your program should consist of the following files:
% 1. A function to get the values for the initial velocity and angle
% of departure from users input.
% 2. A function that takes the above information as inputs, calculates
% and returns how long it takes for the projectile to fall on ground. The
% result of time (in seconds) should be accurate to the second decimal
% place (i.e., 0.01second).
% 3. A function that takes the outputs from the above functions as input,
% and plots the trajectory of the projectile when it is above ground.
% 4. A script file that will call the above three functions.
% Use initial velocity of 60mph and angle of departure of 45 degrees to
% test your program. Attach your results.

% Variables: initV = initial velocity, angle = angle of departure,
% time (in sec) = how long it takes for the projectile to fall on ground,

```
clear
[initV, angle] = readVA;
time = calcTimeFall(initV, angle);
plotTrajectory(initV, angle, time);
```

```
function [initV, angle] = readVA
% readVA prompts the user to input the initial velocity and angle of
% departure
% Format of call: readVA()
% Returns the inputted initial velocity and angle of
% departure

initV = input('Hello! Please enter the initial velocity: ');
angle = input('Hello! Please enter the angle of departure: ');
end
```

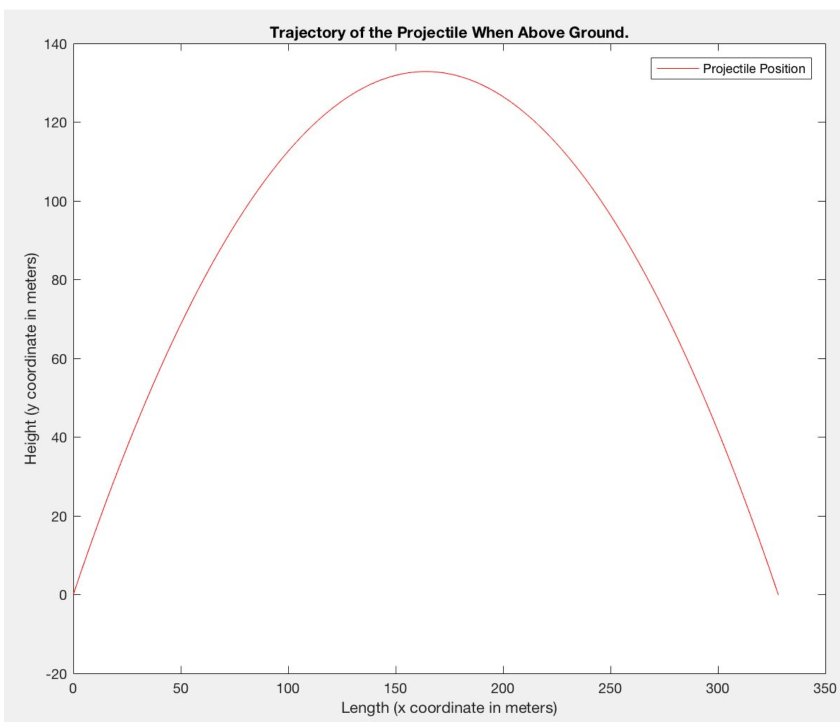
```
function time = calcTimeFall(initV, angle)
% calcTimeFall prompts the user to input the initial velocity and angle of
% departure
% Format of call: calcTimeFall(initial velocity, angle of departure)
% Returns time (in seconds) it takes for the projectile to fall on the
% ground

gravity = 9.81;
rawTime = (2*initV*sin(angle))/gravity;
time = round(rawTime,2);
end
```

```
function plotTrajectory(initV, angle, time)
% plotTrajectory prompts the user to input the initial velocity, angle
% of departure, and time
% Format of call: plotTrajectory(initial velocity, angle of departure, time)
% Plots a graph of the trajectory of the projectile when it is above ground

g = 9.81;
t = 0:.01:time;
x = initV*cos(angle).*t;
y = (initV*sin(angle).*t)-(.5*g.*(t.^2));

plot(x,y,'r');
xlabel('Length (x coordinate in meters)');
ylabel('Height (y coordinate in meters)');
title('Trajectory of the Projectile When Above Ground.');
legend('Projectile Position');
end
```



A M-th order median filter runs through the signal x element by element, replacing each element x(n) (n=1:length(x) with the median value of its M neighboring elements from x(n-$\frac{M-1}{2}$) to x(n+$\frac{M-1}{2}$). For simplicity, we only use odd numbers for M. The boundary pixels are going to be repeated for $\frac{M-1}{2}$ times to obtain enough elements. For example, a median filter with M=3 will be applied to the following simple signal stored in vector x:

x = [2 80 6 3]

As the results, the median filtered output signal y will be:
y[1] = Median[2 2 80] = 2
y[2] = Median[2 80 6] = Median[2 6 80] = 6
y[3] = Median[80 6 3] = Median[3 6 80] = 6

```
% medianFilter.m
% Patrick Utz, 3/2/18, 8.3
% Problem: Write a MATLAB function to implement the median filter.
% This function should take two input arguments, one is the vector to
% be filtered, and the other one is an odd integer M as the order of the
% filter. It should output the resulting median-filtered vector. Run and
% test your function with the above x and M=3, attach your results.

% Variables: medVec = temp vector to store the neighboring values to
% be analyzed, rawVec = unfiltered input vector, filteredVec = output
% filtered vector, M = order of the filter

function filteredVec = medianFilter(rawVec, M)
% medianFilter prompts the user to input the vector to be filtered, and
% the odd integer M as the order of the filter
% Format of call: medianFilter(vector to be filtered, odd integer M as the
% order of the filter)
% Returns the resulting median-filtered vector

medVec = [];
for k = 1:length(rawVec)
   if ( k - ((M-1)/2) ) < 1 || ( k + ((M-1)/2) ) > length(rawVec)
      filteredVec(k) = rawVec(k);
   else
      medVec = rawVec( (k - ((M-1)/2)):(k + ((M-1)/2)) );
      filteredVec(k) = median(medVec);
   end
end
end
>> x = [2 80 6 3]
x =
    2   80    6    3
>> medianFilter(x,3)
ans =
    2    6    6    3
```

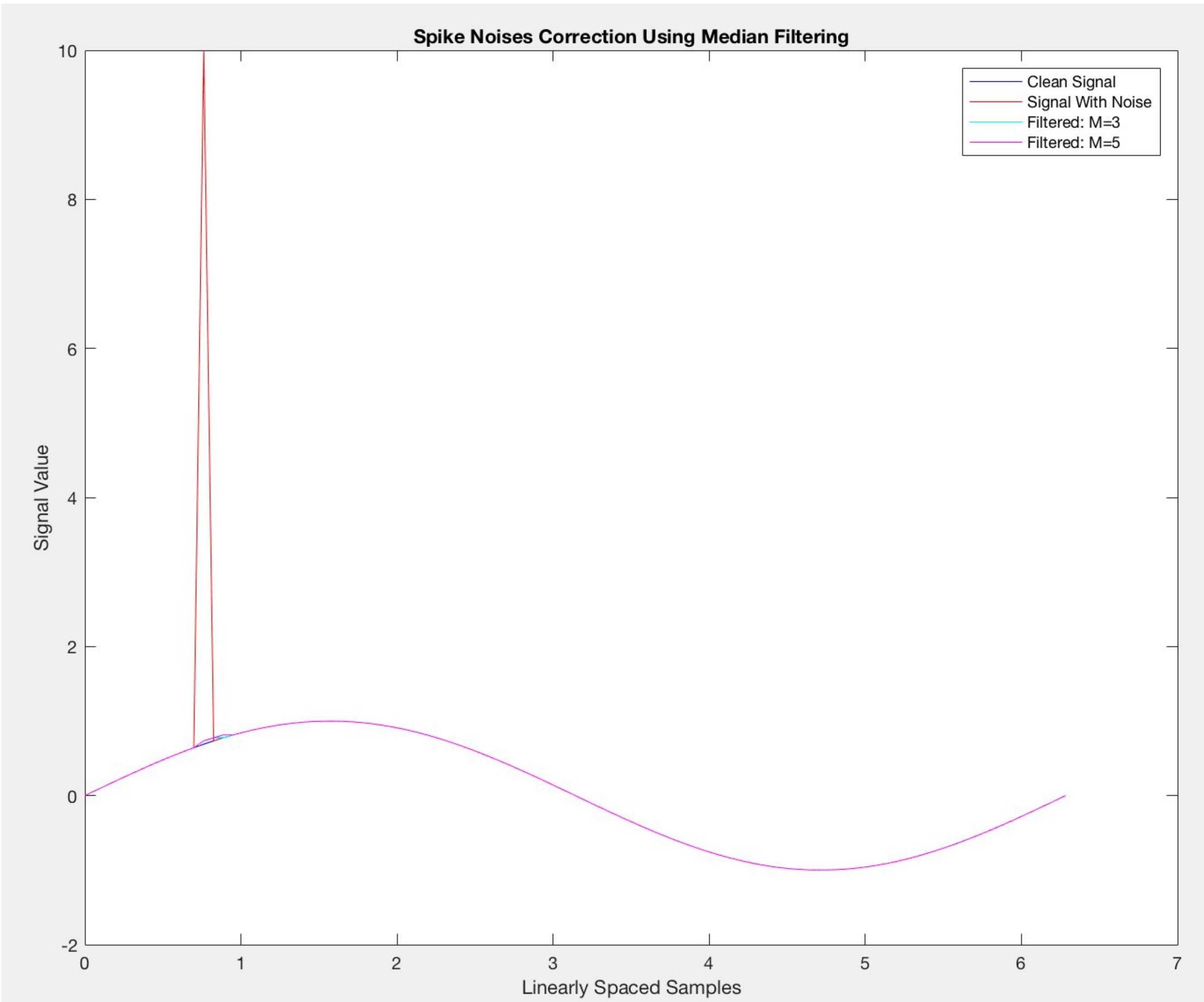
```
% SpikeNoisesSimulation.m
% Patrick Utz, 3/2/18, 8.4

% Problem: Other than random noises, there is another type of noises
% that is common in real-world digital signals, called spike noises, i.e.,
% isolated data samples whose values are very different from the normal
% data samples around. For example, dark pixels in bright area or bright
% pixels in dark area in a digital image, which could be a result of dead
% pixel of imaging device. In this problem, we will write a program to
% simulate the spike noise reduction by median filtering.
% 1. Generate a one-period sinusoidal signal s(n) with 100 linearly spaced
%    samples from 0 to 2pi.
% 2. Add a spike noise with value 10 at a random position in s(t) to
%    generate a noisy signal x(t)
% 3. Call the median filtering function you wrote in Problem 8.3 to perform
%    median filtering to x(t) with M=3, and M=5, respectively.
% 4. Plot the clean signal s(t), the noisy signal x(t), and the two filtered
%    signals in the same plot.
% Run your program and attach your results.

% Variables: n = sample range, s = clean sinusoidal signal, x = sinusoidal
% signal with random noise, M3 = filtered signal with M = 3, M5 = filtered
% signal with M = 5

clear
n = linspace(0,2*pi,100);
s = sin(n);
x = s;
x(randi([1,100])) = 10;
M3 = medianFilter(x,3);
M5 = medianFilter(x,5);

plot(n,s,'b');
hold on
plot(n,x,'r');
plot(n,M3,'c');
plot(n,M5,'m');
xlabel('Linearly Spaced Samples');
ylabel('Signal Value');
title('Spike Noises Correction Using Median Filtering');
legend('Clean Signal','Signal With Noise','Filtered: M=3','Filtered: M=5');
```


Spike Noises Correction Using Median Filtering
Clean Signal
Signal With Noise
Filtered: M=3
Filtered: M=5
Signal Value
Linearly Spaced Samples
10
8
6
4
2
0
-2
0
1
2
3
4
5
6
7