10/11/2021

Automotive Electronics Integration

AuE 835 Assignment-2

SIDDHARTH THORAT
CU-ICAR

Problem 1

Fundamental Operations:

1. Compare the properties of lead-acid battery and Lithium-ion battery in terms of power density, surge current, cost and safety. Give two application examples in an automobile which use lead-acid battery and Lithium-ion battery respectively and analyze the reasons.

PROPERITES	**LITHIUM-ION BATTERY**	**LEAD-ACID BATTERY**
POWER DENSITY	The power density of lithium-ion battery ranges from **245 to 430 W/Kg**.	The power density of lead-acid battery ranges from **35-40 wh/kg**.
SURGE CURRENT	The surge current of lithium-ion battery is **3200 mAh**.	The surge current of lead-acid battery **650 mAh**.
COST	The cost of lithium-ion battery is **more**.	The cost of lead-acid battery is **less**.
SAFETY	The safety of lithium-ion battery is **safer but protection circuit is mandatory**.	The safety of lead-acid battery is less **safe but thermally stable.**

Applications:

<u>Based on Lithium-ion Battery:</u>

a. Starting lighting ignition:

- Starting lighting ignition (SLI) is the car battery placed in every car for the last 100 years. Commonly they are called a ‘12 V battery’, but its normal voltage is approx. 14 V. It should be noted that a conventional lead-acid battery as well as the lithium-ion one for use in cold temperatures where the lithium-ion pack may not be able to provide enough power to crank the engine.

b. Mild Hybrids:

- In a mild hybrid the electrical energy is used to enhancement the energy from the combustion engine. The mild hybrids are similar to self-charging cars but has a smaller battery. Commonly seen in Audi, BMW, Daimler, Porsche and Volkswagen and resulted in the LV 148 standard.

Based on Lead acid battery:

a. Ignition Power:

- The engine spark from the coil or coils is directed towards the spark plugs to ignite the fuel in the vehicle. Generally, a 12 V battery is used in an automobile.

b. UPS Systems:

- The UPS system generally provides a battery back-up for vehicle tracking systems and other electronics in a vehicle.

2. Compare the CAN bus and LIN bus in terms of bandwidth, wiring, topology and cost. Give two application examples in an automobile which use CAN bus and Lin bus respectively and analyze the reasons.

PROPERITES	**CANBUS**	**LINBUS**
BANDWIDTH	The bandwidth of CANBUS ranges from **62.5 to 500 k baud Or 1 Mbit/s.**	The LINBUS bandwidth ranges from **2.4 to 19.6 baud or 20 kbit/s.**
WIRING	The CANBUS has **Twisted pair of 5 V wiring**.	The LINBUS has **Single Wire of 12 V (battery) of wiring**.
COST	The cost of CANBUS is **more**.	The cost of LINBUS is **half of the cost of CANBUS**.
TOPOLOGY	The topology is **BUS** where it is determined by maximum allowed Bus length or maximum undetermined huge lines connected to main Bus line and number of nodes.	The topology is **BUS** which is determined by serial network protocol used for communication between the components in the vehicle**.**

Applications:

Based on CANBUS:

a. Dash Systems:

- The dash systems consist of vital controls and readings which informs the way we function the car. If the dash systems are not responding to the controls and meters then we might go through serious injuries.

b. ECU'S:

- The engine control unit (ECU) is an essential part of the vehicle which regulates the engine's operations. It depends on the input from sensors and computers around the vehicle to enhance the fuel-air mixture and other characteristics of motor functioning.

Based on LINBUS:

a. Steering Wheel:

- Many controls are going to be positioned on the steering wheel like Cruise control, wiper, turning light, climate control, radio.

b. Roof:

- High amount of wiring is done where it should continuously monitor for every 10 to minutes like Rain sensor, Light sensor, Light control, Sun Roof.

Problem 2

Basic MATLAB Applications:

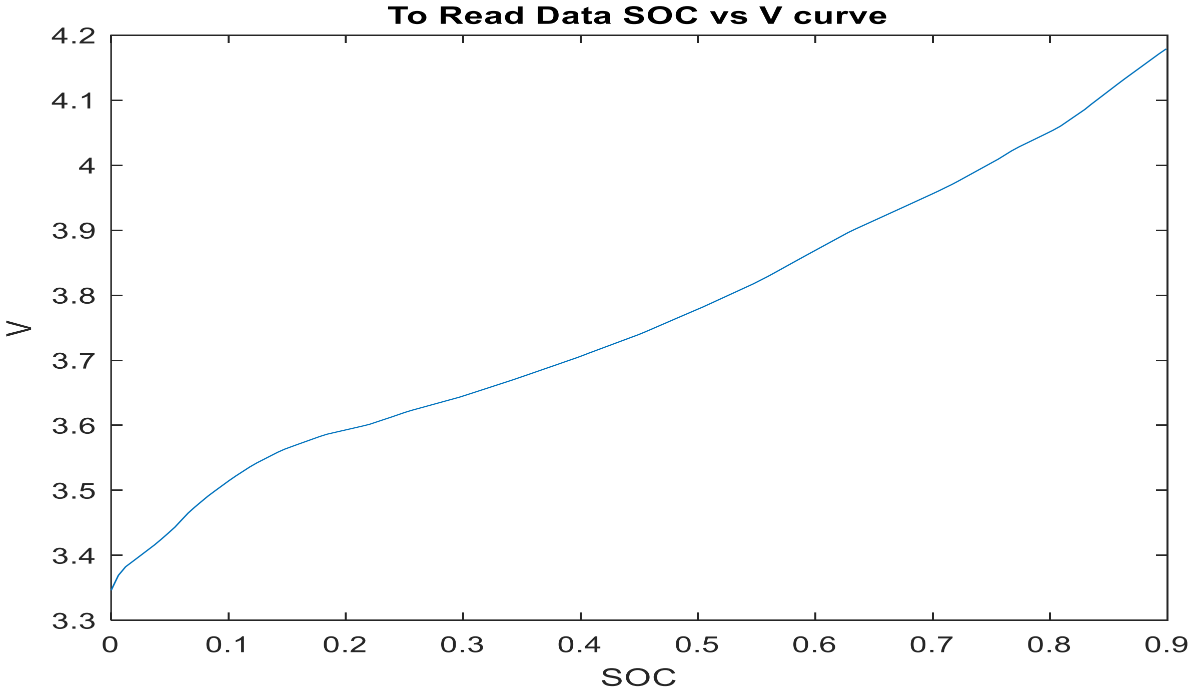
Statement: The internal resistance of a battery is not constant and it constantly changes while charging. If we want to build an accurate simulation model for battery system, we need to find the changing trend of the battery internal resistance.To understand this, we conducted an experiment where we charge a battery at 1 ampere current. As the battery charged, its voltage increased along with its State of Charge (SOC). SOC is the fraction of total capacity held by a battery and it ranges from 0 to 1. A battery is fully charged when SOC is 1 and it is fully discharged when SOC is 0.“Data.mat” contains the measured Voltage, Temperature and SOC data. By opening this file, you can load following data into MATLAB:□Voltage (V) that is applied to the battery to charge it.□Temperature (temp) of the battery while charging.□State of Charge (SOC) of the battery at that instant.We can estimate the battery’s internal resistance by using equation:R=V—E0 is the open circuit voltage of the battery. It can be estimated by using equation:E0E0=0.0026·$temp$+7.9387·10—4·$temp$·SOC+0.0016·$temp$·SOC2+2.5625With those two equations, we can find the relation between internal resistance and SOC.Use MATLAB to conduct following tasks:

I. Read data from Data.mat, plot the curve voltage V vs. SOC. Make sure you have put label to your axis and title to your figure.

MATLAB Code:

```
clc;

plot (SOC,V)

title('To Read V vs soc curve')

xlabel(SOC);

ylabel('V')
```

GRAPH:



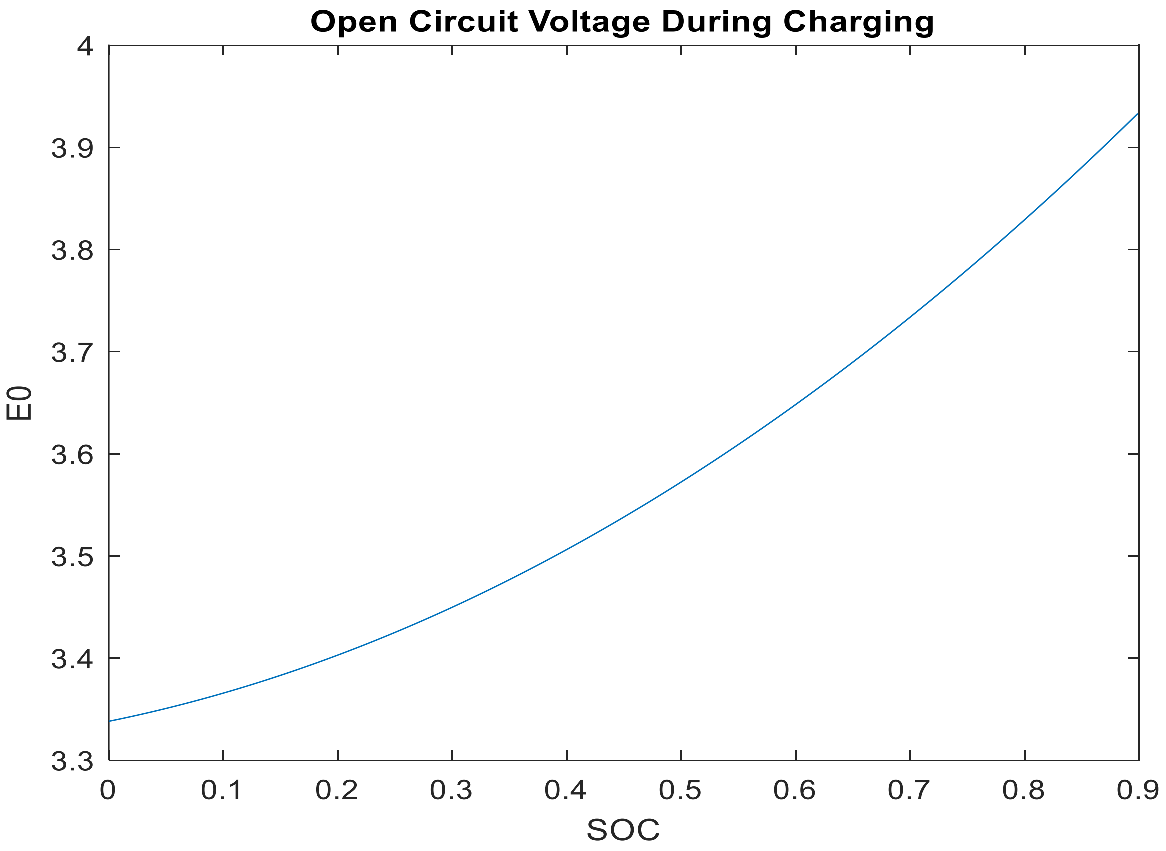
2. Calculate open circuit voltage and internal resistance R during charging. Plot the *E*0 curve vs. SOC and R vs. SOC in separate figures. Make sure you have put label to your *E*0axis and title to your figure.

I. *E*0 curve vs. SOC

MATLAB Code:

```
clc;
E0=(0.0026.*temp)+(7.9387.*0.0001.*temp.*SOC)+(0.0016.*temp.*SOC.*SOC)+2.5625;
plot (SOC,E0)
title('Open Circuit Voltage During Charging')
xlabel(‘SOC’)
ylabel('E0')
```

GRAPH:

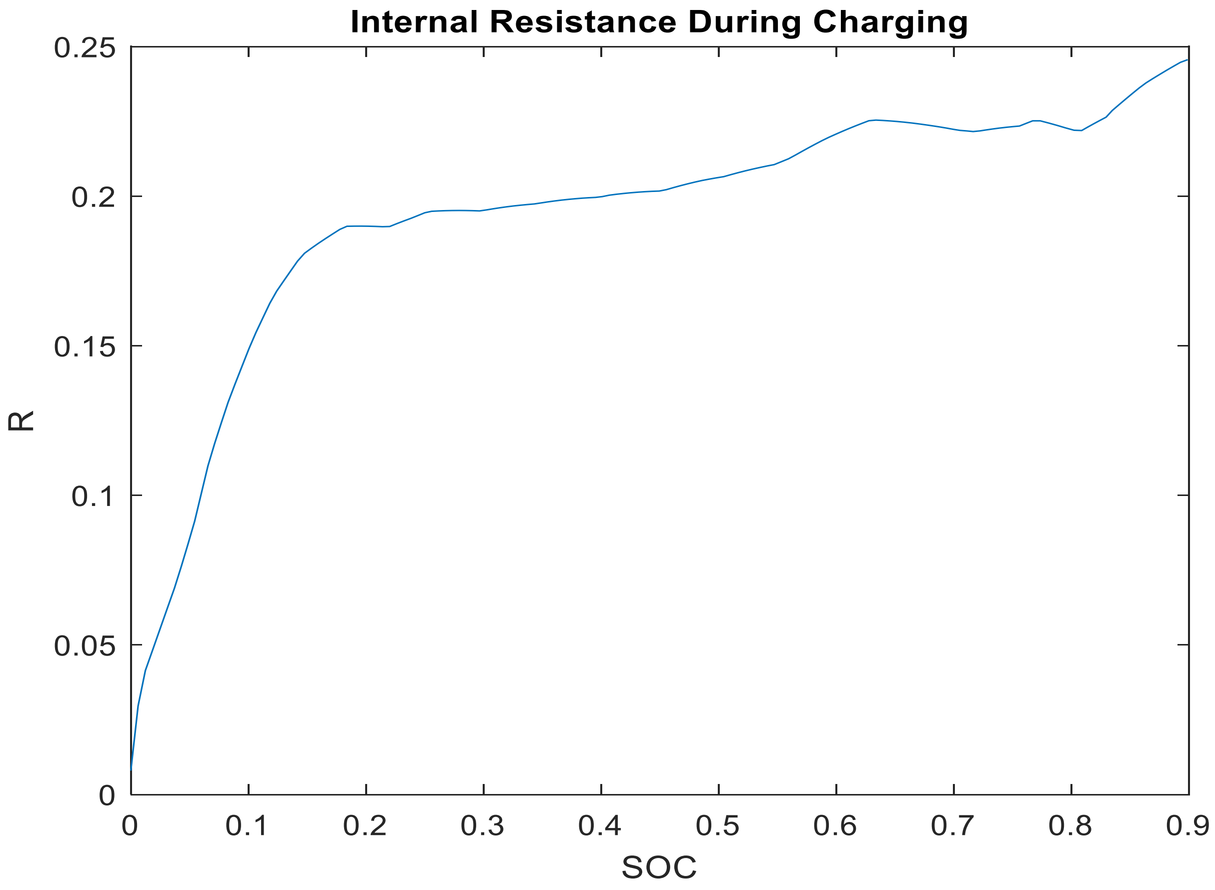


II. R vs. SOC:

MATLAB Code:

```
clc;
E0 =(0.0026.*temp)+(7.9387.*0.0001.*temp.*SOC)+(0.0016.*temp.*SOC.*SOC)+2.5625;
R=V-E0;
plot (SOC,R)
title('Internal Resistance During Charging')
xlabel(‘SOC’)
ylabel('R')
```

GRAPH:

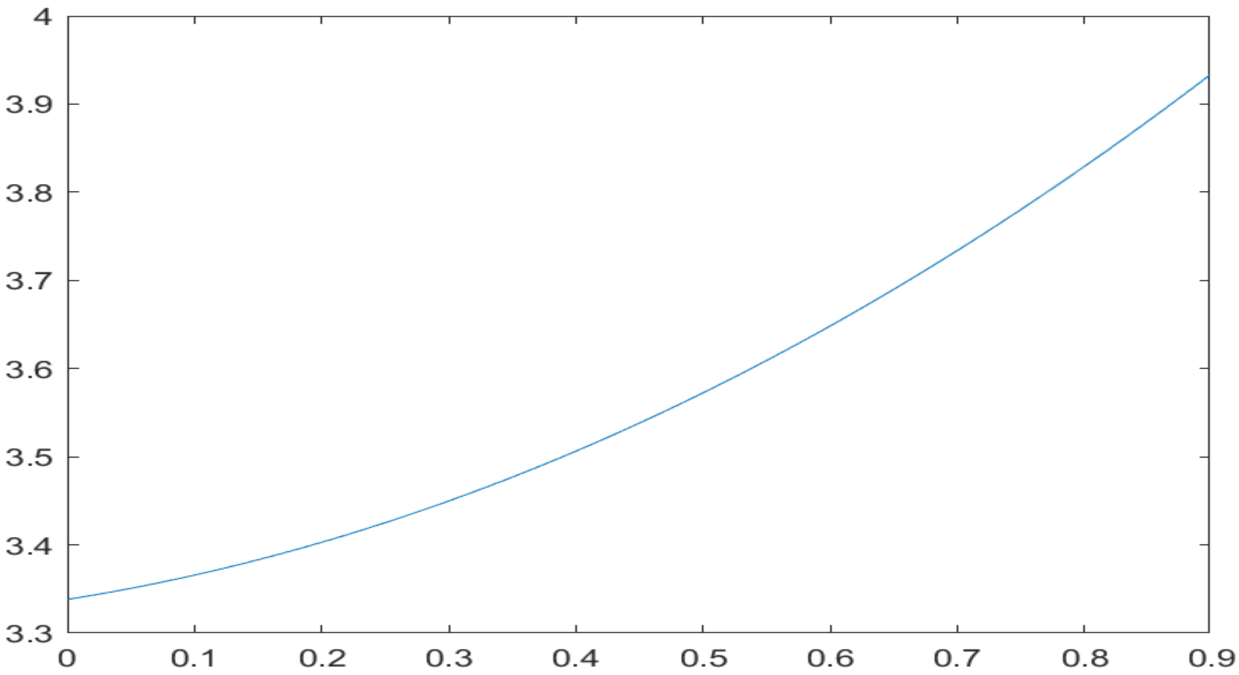


III. By using a loop, evaluate the R matrix and find the greatest R value between SOC 0.6 and 0.8. What is the corresponding SOC at that R value?``

MATLAB Code:

```
clc;
E0 =(0.0026.*temp)+(7.9387.*0.0001.*temp.*soc)+(0.0016.*temp.*soc.*soc)+2.5625;
R=V-E0;
plot (V,soc)
plot (soc,E0);
MAX=0;
for i=1:153
   if soc(i)>0.6
      if soc(i)<0.8
         if R(i)>MAX
            MAX=R(i);
            N=i;
         end
      end
   end
end
   x=soc(N)
   fprintf("soc at RMAX %f",x)
```

Graph:



- The corresponding SOC at that R value

```
x =

   0.6337

soc at RMAX 0.633690>>
```

Problem-3:

Using MATLAB to Implement Low Filter:

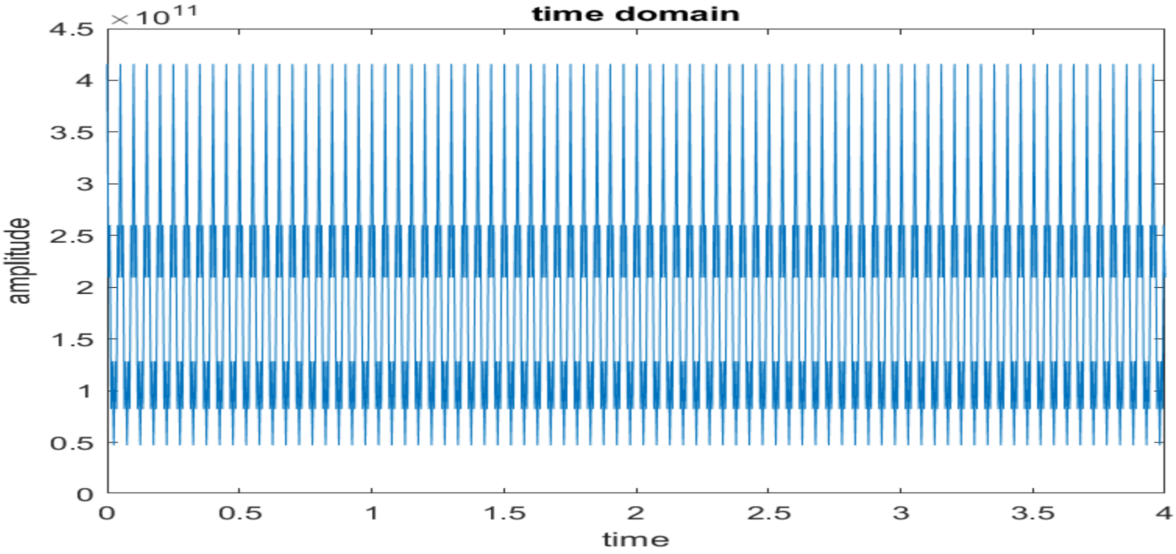
I. Load “r Load “rawdata.mat” and plot it in time domain (the real data has a low frequency and noise Load “rawdata.mat” and plot it in time domain (the real data has a low frequency and noise has a higher frequency). The sampling frequency is 200 Hz. has a higher frequency). The sampling frequency is 200 Hz. awdata.mat” and plot it in time domain (the real data has a low frequency a Load “rawdata.mat” and plot it in time domain (the real data has a low frequency and noise has a higher frequency). The sampling frequency is 200 Hz. nd noise has a higher frequency). The sampling frequency is 200 Hz.

MATLAB CODE:

```
clc;
```

```
load rawdata.mat

% time domain

fs=200

t = linspace(0,length(X)/fs,length(X));

figure;

plot(t,X)

title('time domain')

xlabel('time')

ylabel('amplitude')
```

GRAPH:



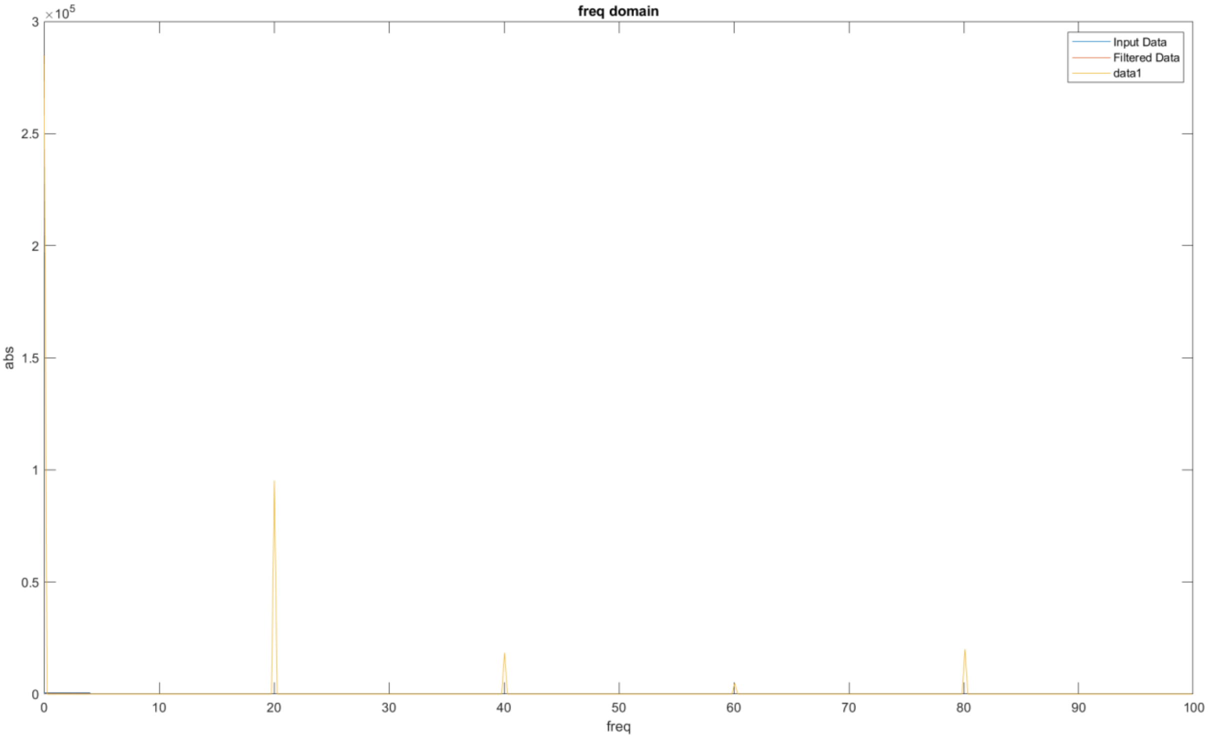
II. Transfer the data to frequency domain using “fft” function and plot its power spectrum

MATLAB Code:

```
% frequency domain

nfft=800;

f= linspace(0,fs,nfft);

X= abs(fft(X,nfft));

figure;

plot(f(1:nfft/2), X(1:nfft/2));
```

```
title('freq domain')

xlabel('freq')

ylabel('abs')
```

GRAPH:



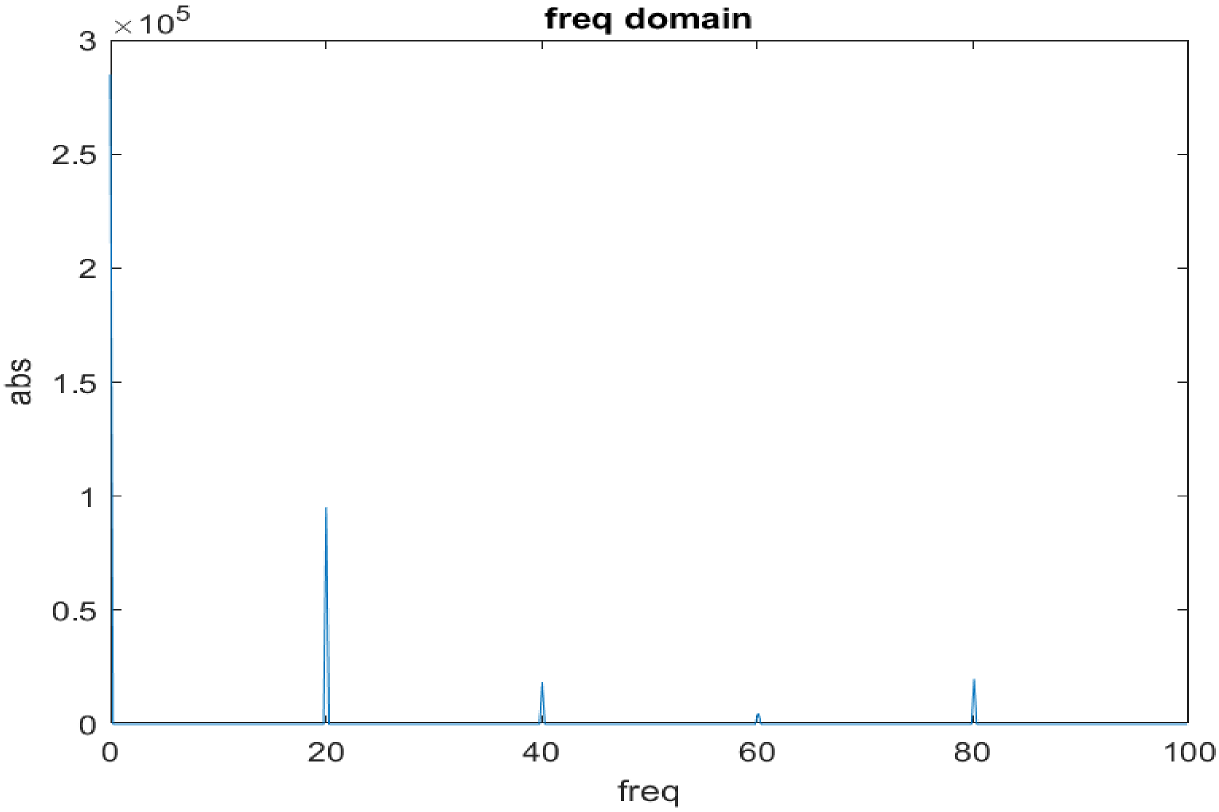
III. Design a low-pass filter using “butter” function and plot it in frequency domain

MATLAB Code:

```
%%filter the data using low pass butterworth filter

fc=10;

fs=200;

n_order=6; %filter order

[b,a]=butter(n_order,fc/(fs/2))

figure(3);
```

```
plot(f(1:nfft/2), X(1:nfft/2));
title('freq domain')
xlabel('freq')
ylabel('abs')
```

GRAPH:



IV. Filter the high-frequency noise using “filter” function with the designed low-pass filter using “butter” and plot the filtered data in time domain.

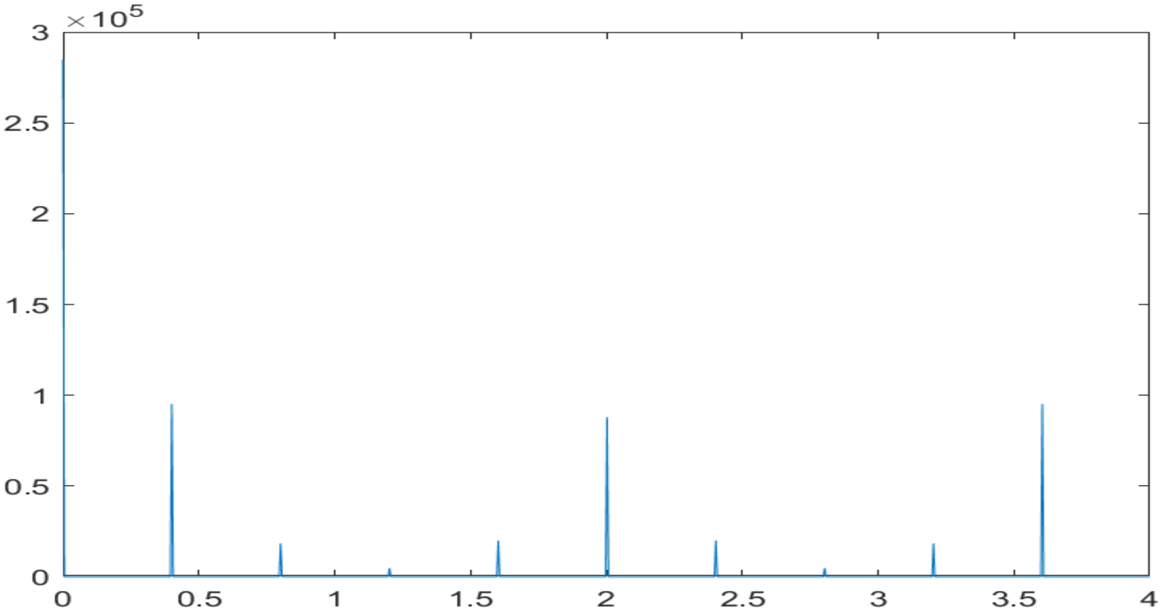
MATLAB Code:

```
%% filter the data using filter function
y = filter(b,a,X);
figure(4);
plot(t,X)
figure(5);
plot(t,y)
```

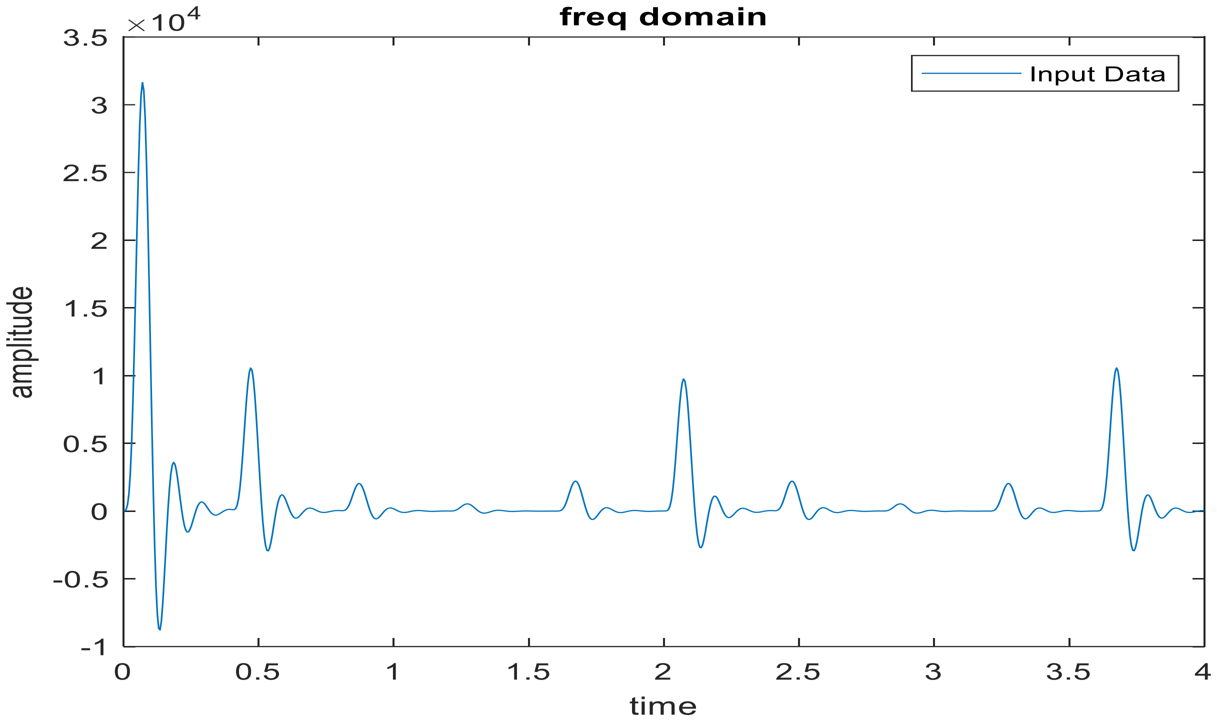
```
legend('Input Data','Filtered Data')

title('freq domain')

xlabel('time')

ylabel('amplitude')
```

GRAPH:

IV.a.



IV.b.



Problem 4

Kalman Filter:

I. Use MATLAB to implement a Kalman filter for removing noises in a vehicle position sensor

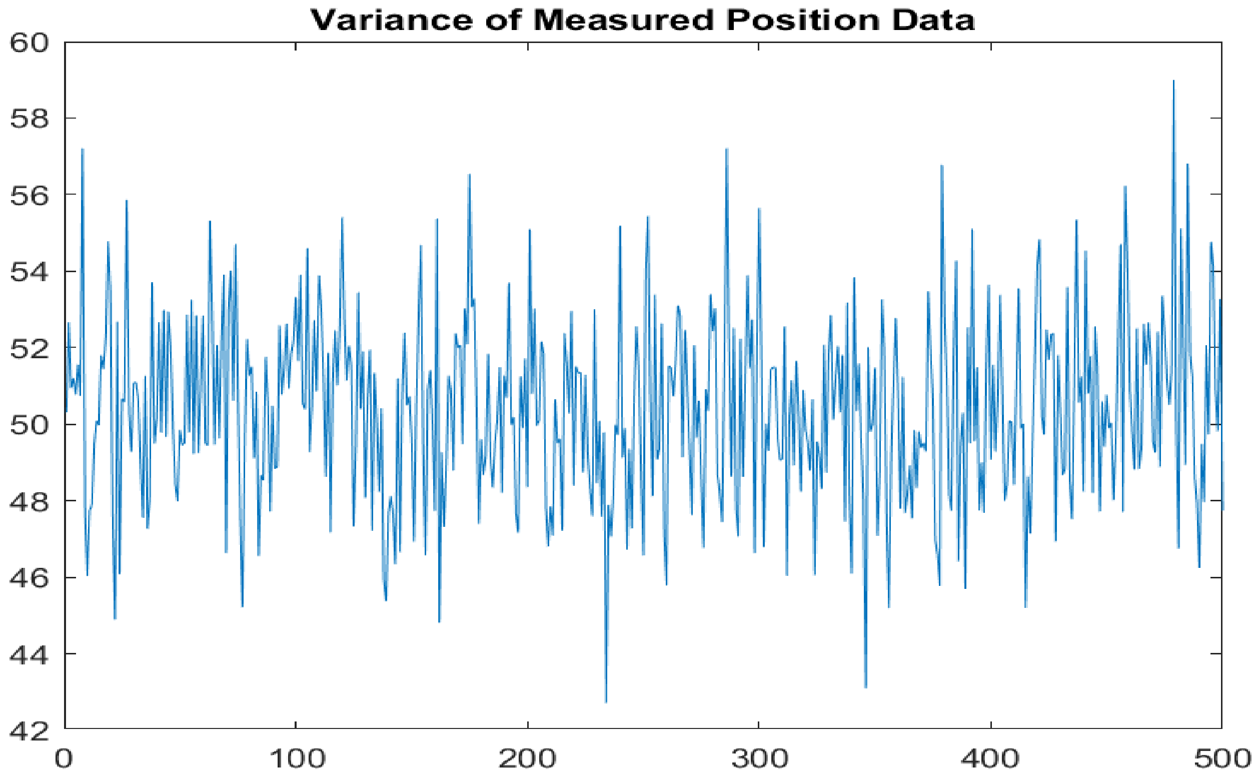
a. Use MATLAB to implement a Kalman filter for removing noises in a vehicle position sensora. Load “pos.mat” and plot it. Calculate variance of the measured position data.

MATLAB Code:

```
clc;
V=var(z);
plot (z)
```

GRAPH:

x



b. Write down the constant process model. Design a Kalman Filter to filter the noises and estimate the actual vehicle position (including prediction and correction equations).

A. Constant process model:

$Y_k = Ay_{k-1}+Bu_k+w_k$

$Z_k=Hy_k+v_k$

Where,

- K=Discrete Time
- Y=System data to be estimated
- U=System input
- Z=System Measurement
- A, B, H=Constant Matrices
- w=Process noise w approx. N (0, Q)
- v=measurement noise with y approx. N (0, R)

i. When the vehicle position is stationary:
$Y_k = Ay_{k-1}+Bu_k+w_k$
$Y_k = 1*y_{k-1}+0(u_k)+w_k$
$Z_k=1*y_k+v_k$

ii. When the vehicle is moving with speed “u”:

$Y_k = Ay_{k-1}+Bu_k+w_k$
$Y_k = 1*y_{k-1}+T*u_k+w_k$ (constant process)
$Z_k=1*y_k+v_k$ (measurement process)

iii. When the vehicle is moving with speed "y":

$Y_k = Ay_{k-1}+Bu_k+w_k$
$Y_k = y_{k-1}+T\ y_{k-1}$
$Y_k = y_{k-1}$

Process model:

$Y= [y: y^{-}]$

$Y_k = Ay_{k-1}+w_k$

$Z_k=Hy_k+v_k$

$Y_k=[1, T : 0, 1]*y_{k-1}+w_k$

$Z_k=[1\ 0]*y_k+v_k$

B. At vehicle constant:

- Process Model:

- $Y_k = Ay_{k-1}+Bu_k+w_k$
- $Z_k=Hy_k+v_k$

- Vehicle at constant position:

  - $Y_k = 1*y_{k-1}+w_k$
  - $Z_k=1*y_k +v_k$

- At prediction Step:

  - $\hat{y}\ ^{-}{}_k=Ay_{k-1}+Bu_k$
  - $P^{-}{}_k=AP_{k-1}A^T+Q$

Solution-

- $\hat{y}^{-}_{k}=y_{k-1}$
- $P^{-}_{k}=P_{k-1}$

➢ At correction step:

- $K=P^{-}_{k} H^{T} (HP^{-}_{k}H^{T} + R)^{-1}$
- $\hat{y}_{k}= \hat{y}^{-}_{k} +K (z_{k} - H \hat{y}^{-}_{k})$
- $Pk= (I -K H) P^{-}_{k}$

Solution-

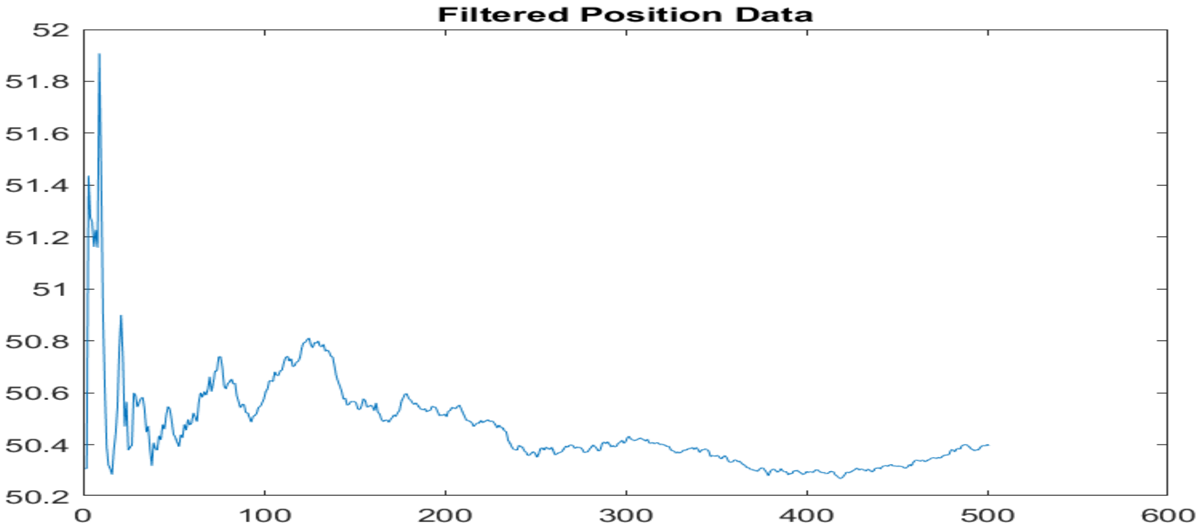
- $K=P^{-}_{k} (P^{-}_{k} +R)^{-1}$
- $\hat{y}_{k}= \hat{y}^{-}_{k} + K (z_{k}- \hat{y}^{-}_{k})$
- $Pk= (I -KH)P^{-}_{k}$

c. Implement the Kalman filter in MATLAB through coding all the prediction and correction equations and plot the filtered position data.

MATLAB Code:

```
%% kalman filter through prediction and correction equations

A=1;

B=0;

H=1;

R=0.5;

yk(1)=z(1);

for i=1:500

   yk(i+1)=A*yk(i);

   V=A*V;

   k=V*H/((V*H+R));
   yk(i+1)=yk(i+1)+k*(z(i)-(H*yk(i+1)));
   V=(1-(k*H))*V;
end
figure (2);
plot (yk)
```

GRAPH:



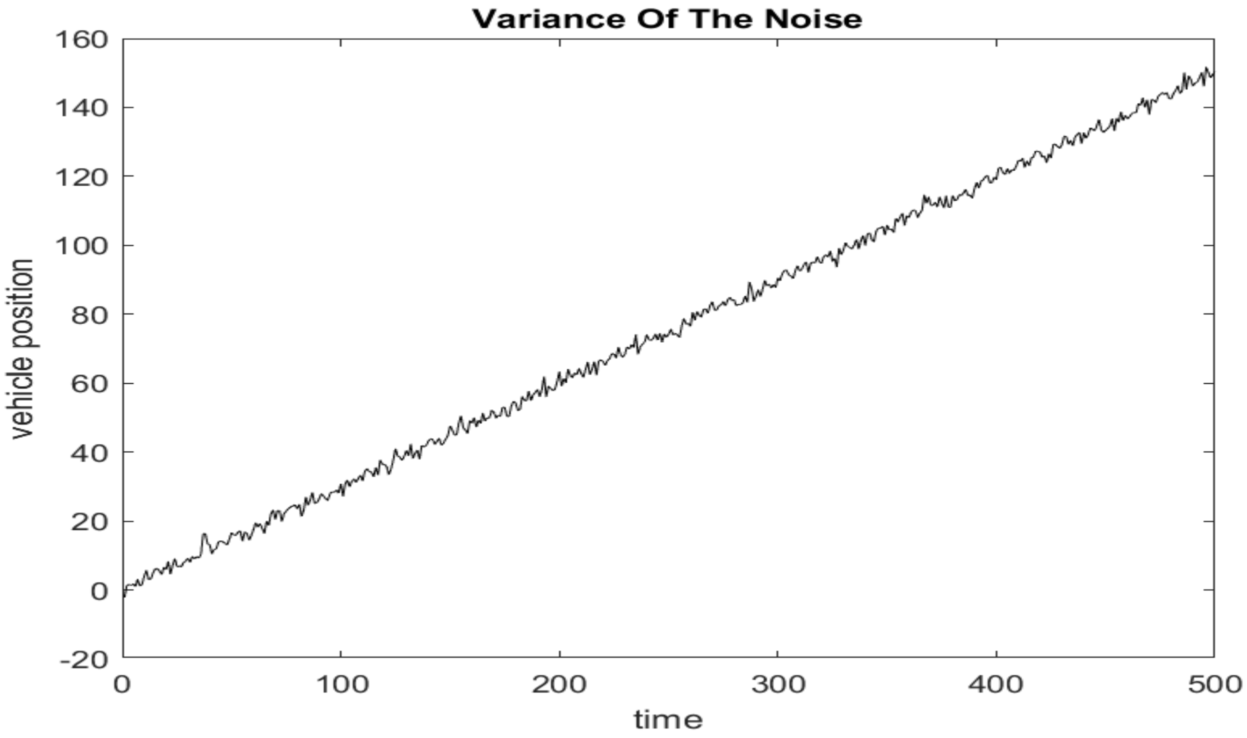
II. Use MATLAB to implement a Kalman filter to obtain the vehicle position by fusing a vehicle speed sensor and a GPS sensor

a. Load “gps.mat” and plot the vehicle positions vs. time. Use “polyfit” to fit the signal and then calculate the variance of the noise.

MATLAB Code:

```
clc;
load gps.mat
t=1:500;
figure (1)
plot (t,z,'r')
title ( 'Variance Of The Noise')
xlabel('time')
ylabel('vehicle position')
L=1:length(z);
p=polyfit(L,z,1);  %% Syntax of polyfit (p)=polyfit(x,y,n)
p_v=polyval(p,L);
variance=var(p_v)
```

GRAPH:

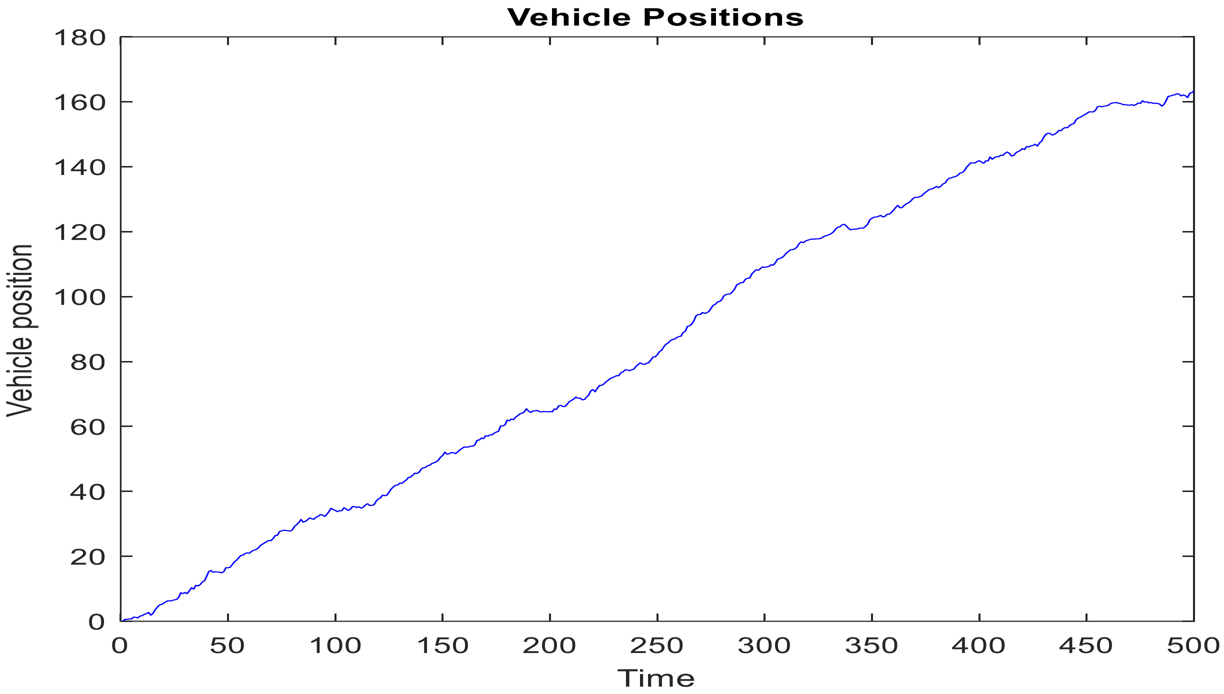


b. Load “speed.mat” and calculate standard deviation of the measured data. Use process model “x(k) = x(k-1) + 1*speed(k)” to calculate the vehicle positions. Plot the positions vs. time.

MATLAB Code:

```
clc;
load speed.mat
x(1)=0;
S=std(u);    %% "standard deviation syntax S = std(A)"
for i=1:(length(u)-1)
    x(i+1)=x(i)+1*u(i);
end
figure (2)
plot (x,'b')
title('Vehicle Position')
xlabel('Time')
ylabel('Vehicle position')
```

GRAPH:



c. Taking speed as system input and GPS position as system measurement, write down the whole process model. Design a Kalman filter to estimate the position x.

. Constant process model:

$x_k = Ax_{k-1}+Bu_k+w_k$

$Z_k=Hx_k+x_k$

Where,

- K=Discrete Time
- x=System data to be estimated
- Speed(U)=System input
- GPS(Z)=System Measurement
- A, B, H=Constant Matrices
- w=Process noise w approx. N (0, Q)
- v=measurement noise with y approx. N (0, R)

iv. When the position is stationary:
$x_k = Ax_{k-1}+Bu_k+w_k$
$x_k = 1*x_{k-1}+0(u_k)+w_k$
$Z_k=1*x_k+x_k$

v. When the x position is moving with speed “u”:

$x_k = Ax_{k-1}+Bu_k+w_k$
$x_k = 1*x_{k-1}+T*u_k+w_k$ (constant process)
$Z_k=1*x_k+v_k$ (measurement process)

vi. When the position x is moving with speed “y”:

$x_k = Ax_{k-1}+Bu_k+w_k$
$x_k = x_{k-1}+T\ x_{k-1}$
$x_k = x_{k-1}$

Process model:

$x= [x: x^{-}]$

$x_k = Ax_{k-1}+w_k$

$Z_k=Hx_k+x_k$

$x_k=[1, T : 0, 1]*x_{k-1}+w_k$

$Z_k=[1\ 0]*x_k+x_k$

B. At vehicle constant:

- Process Model:

- $x_k = Ax_{k-1}+Bu_k+w_k$
- $Z_k=Hy_k+x_k$

- Vehicle at constant position:

    - $x_k = 1*x_{k-1}+w_k$
    - $Z_k=1*x_k +v_k$

- At prediction Step:

    - $x^{-}_{k}=Ax_{k-1}+Bu_k$
    - $P^{-}_{k}=AP_{k-1}A^{T}+Q$

Solution-

- $x^{-}_{k}=x_{k-1}$

- P^-_k=P_{k-1}

- At correction step:

- K=P^-_k H^T ($HP^-_kH^T$ + R)$^{-1}$
- x_k= x_k +K (z_k - H x^-_k)
- Pk= (I -K H) P^-_k

Solution-

- K=P^-_k (P^-_k +R)$^{-1}$
- x_k= x^-_k + K (z_k- x^-_k)
- Pk= (I -KH)P^-_k

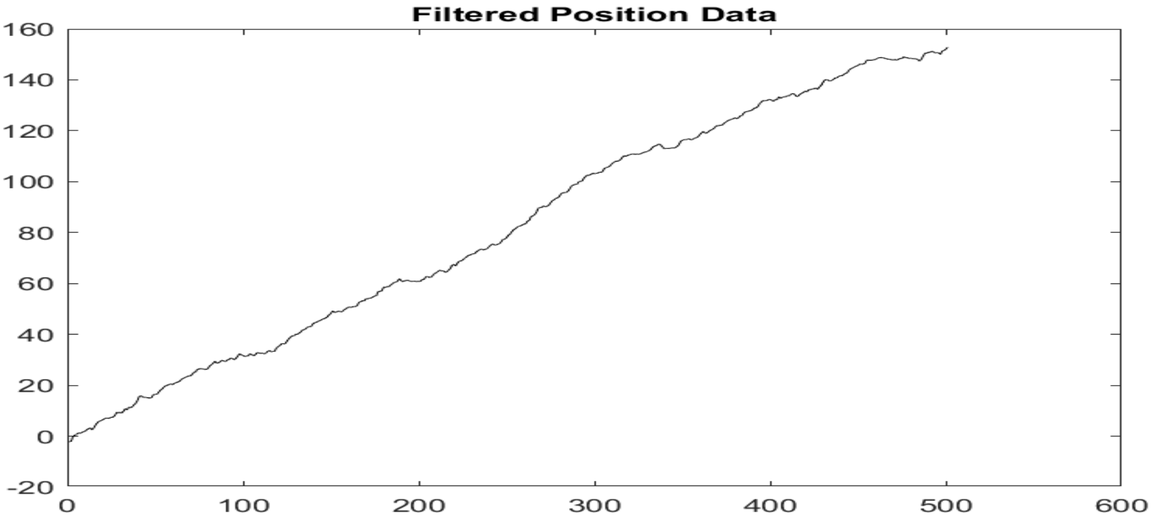
d. Implement the designed Kalman filter in MATLAB and plot the filtered position data.

MATLAB Code:

```
%% kalman filter through prediction and correction equations

A=1;

B=0;

H=1;

R=0.5;

t=1;

yk(1)=z(1);

for i=1:500

   yk(i+1)=A*yk(i)+t*u(i);

   V=A*V;


   k=V*H/((V*H+R));

   yk(i+1)=yk(i+1)+k*(z(i)-(H*yk(i+1)));
```

```
    V=(1-(k*H))*V;
end
figure (3);
plot (yk,’r’)
title ('Filtered Position Data')
```

GRAPH:


Filtered Position Data