6.1 Power spectral densities and transfer functions
Take the task description of exercise 5.1 (previous exercise).
Calculate the pdf $f_y(y)$. Remember that the input signal u(e, t) is Gaussian noise.

6.2 Autocorrelation function
Let a discrete stationary random process x (ζ, t).
The outcomes of the process are the values $x_1 = -1, x_2 = 0$ and $x_3 = 1$
The probabilities of the occurrence of those outcomes are

$$P(\{x(\zeta, t+\tau) = x_i\} | \{x(\zeta, t) = x_j\}) = \begin{cases} \frac{1}{3}\left(1 + 2\, e^{-|\tau|}\right) & for\ i = j \\ \frac{1}{3}\left(1 - e^{-|\tau|}\right) & for\ i \neq j \end{cases} \qquad i, j = 1,2,3$$

a) Calculate the probabilities

$$P(\{x(\zeta, t) = x_i\}) \qquad for\ i = 1,2,3.$$

b) Calculate the ACF $s_{xx}(\tau)$. Solve this part in Matlab.

6.3 Kalman Filter
Run the following Matlab demo (also provided in a Word file) for a Kalman filter with two different noise levels

a)
```
car_accel_noise_mag = 0.05;
robot_noise_mag = .10;
```

b)
```
car_accel_noise_mag = OWN CHOICE;
robot_noise_mag = OWN CHOICE;
```

and provide the plots of a) and b)

```
clear all
% duration and how often we sample
duration = 10; %car ride duration
dt = .1;  % sampling distance

% Define update equations
Fk = [1 dt; 0 1] ; %State Transition Matrix
Bk = [dt^2/2; dt]; %Input Control Matrix
Hk = [1 0]; % Measurement matrix
%we are only measuring position, so velocity variable is set to zero.

% main variables
u = 1.5; % acceleration mag
x= [0; 0]; %initial state vector, car has two components: [position; velocity]
xhat = x;  %initial state estimation of where the car is (what we are updating)
car_accel_noise_mag = 0.05; %process noise -standard deviation of acceleration
robot_noise_mag = .10;  %measurement noise -standard deviation of location
sigmaw = car_accel_noise_mag^2 * [dt^4/4 dt^3/2; dt^3/2 dt^2]; % Process noise covariance
matrix
Rk = robot_noise_mag^2;% measurement noise covariance matrix
Pk = sigmaw; % initial estimation of car position covariance

% result variables
pos = []; % Actual car ride trajectory
vel = []; % Actual car velocity
Zk = []; % car trajectory that the robot sees (measured) robots perception

% simulate what robot sees over time
for t = 0 : dt: duration

    % Generate the car ride
    processNoise = car_accel_noise_mag * [(dt^2/2)*randn; dt*randn];
    x= Fk * x+ Bk * u + processNoise;
    % Generate what the robot sees
    measurementNoise = robot_noise_mag * randn*100;
    y = Hk * x+ measurementNoise;
```

```
    pos = [pos; x(1)];
    Zk = [Zk; y];
    vel = [vel; x(2)];
end

% Plot the results
figure(1);
tt1=0:dt:t;
% Actual ride of car % what robot sees contineously %theoretical trajectory of robot that 
doesn't use kalman,but using moving average summing in window
plot(tt1, pos, '-r.',tt1, Zk, '-k.',tt1, smooth(Zk), '-g.'),title ('without kalman filter'),
axis([0 10 -20 80]),legend('Actual trajectory of car','what robot sees','estimate' );


% using kalman filtering
% estimation variables
pos_estimate = []; % car position estimate
vel_estimate = []; % car velocity estimate
x= [0; 0]; % reinitialize the state

P_mag_estimate = [];
predict_state = [];
predict_var = [];
for t = 1:length(pos)
    % Predict next state of the car with the last state and predicted motion.
    xhat = Fk * xhat + Bk * u;
    predict_state = [predict_state; xhat(1)] ;

    %predict next covariance
    Pk = Fk * Pk * Fk' + sigmaw;
    predict_var = [predict_var; Pk] ;

    % predicted robot measurement covariance
    % Kalman Gain
    K = Pk*Hk'*inv(Hk*Pk*Hk'+Rk);
    % Update the state estimate.
    xhat = xhat + K * (Zk(t) - Hk * xhat);
    % update covariance estimation.
    Pk =  (eye(2)-K*Hk)*Pk;

    %Store result for plotting
    pos_estimate = [pos_estimate; xhat(1)];
    vel_estimate = [vel_estimate; xhat(2)];
    P_mag_estimate = [P_mag_estimate; Pk(1)];
end

% Plot the results
figure(2);
tt2 = 0 : dt : duration;
plot(tt2,pos,'-r.',tt2,Zk,'-k.', tt2,pos_estimate,'-g.'),title ('with kalman filter'),
axis([0 10 -20 80]),legend('Actual trajectory of car','what robot sees','kalman filter 
estimate' );


%plot the evolution of the distributions
figure(3);
for T = 1:length(pos_estimate)
clf
    x = pos_estimate(T)-5:.01:pos_estimate(T)+5; %  x axis range

    %predicted next position of the car
    hold on
    mu = predict_state(T); % mean
    sigma = predict_var(T); % standard deviation
    y = normpdf(x,mu,sigma); % pdf
    y = y/(max(y));
    hl = line(x,y,'Color','m');

    %data measured by the robot
    mu = Zk(T); % mean
    sigma = robot_noise_mag; % standard deviation
    y = normpdf(x,mu,sigma); % pdf
    y = y/(max(y));
    hl = line(x,y,'Color','k'); % or use hold on and normal plot

    %combined position estimate
    mu = pos_estimate(T); % mean
    sigma = P_mag_estimate(T); % standard deviation
```

```
    y = normpdf(x,mu,sigma); % pdf
    y = y/(max(y));
    hl = line(x,y, 'Color','g');
    axis([pos_estimate(T)-5 pos_estimate(T)+5 0 1]);


    %actual position of the car
    plot(pos(T));
    ylim=get(gca,'ylim');
    line([pos(T);pos(T)],ylim.','linewidth',2,'color','b');
    legend('state predicted','measurement','state estimate','actual car position')
    pause
end
```