Sharif university of technology

Elecrical engineering department

Matlab assignment #2

Stochastic process

Professor behnia

Student: kasra fallah

Student number: 97109987

Apr 2021

Question 1

1.

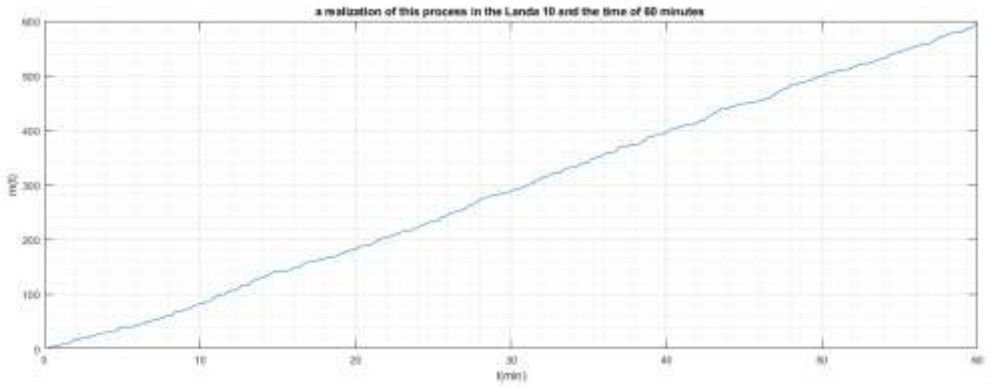
To write this function, all we have to do is generate the Poisson points in one-second intervals and then execute the cumulative function using the cumsum() function.

To generate Poisson numbers, I used the Poisson Rand function, which you can see in the MATLAB code snippet below.

```
function [time,out] = poisoin(lambda,t)
precition = lambda/3600;
time = 0:1:t*3600;
time = time(1:end -1);
out = cumsum(poissrnd(precition,t*3600,1));
poissrnd(precition);
end
```

2.

Currently a realization of this process can be seen in the Lambda 10 experiment and the time of 60 minutes in the diagram below.



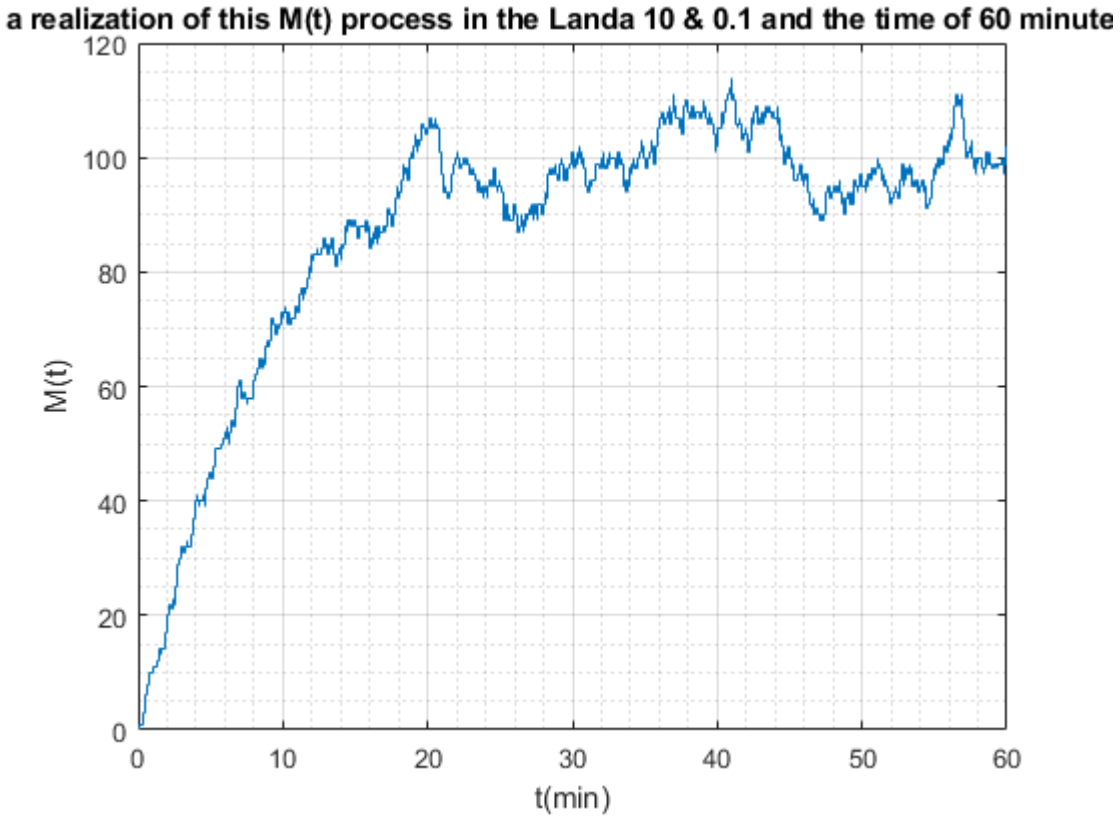
3.

To create an M(T) process, we need to run one process per second for Lambda 10 and add the result to the previous values, and at the same time, for the established calls (the value of "M(t)" in the previous moment) we need the value of a process. Decrease with Lambda 0.1, which is mentioned in the MATLAB code cutout below.

```
m = zeros(1, 2160000);
m(1) = 0;
for i =2:length(m)
    [a,b] = poisoin(10,1/3600);
    m(i) = m(i-1)+b - calltime(0.1,m(i-1));
end
figure;
plot(m)
end
```

```
function [out] = calltime(lambda,number)
precition = lambda/3600;
out = sum(poissrnd(precition,number,1));
end
```

Now I plotted the result in below diagram.



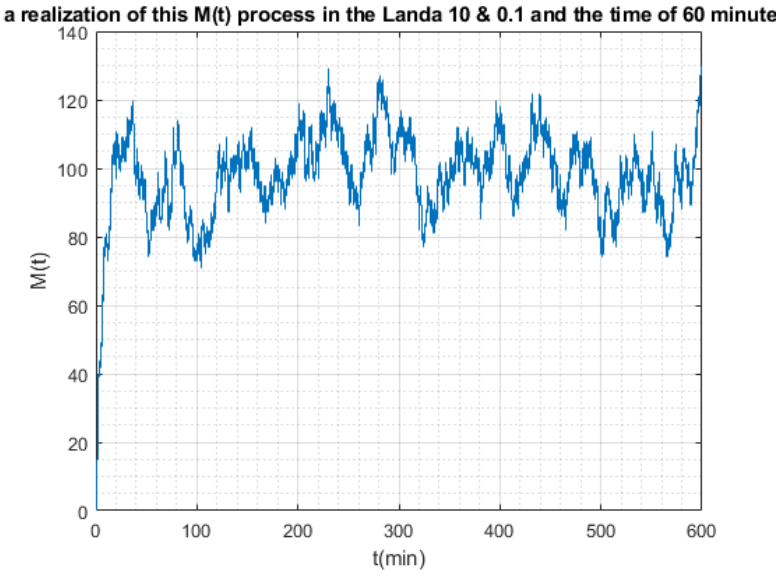
4.

Our process tends to be about 100 simultaneous calls, which is very clear to us because the number of simultaneous calls multiplied by the completion rate must be equal to the value of the call rate to have a balanced system:

$$\lambda_{\text{completion}} * n = \lambda_{adition}$$

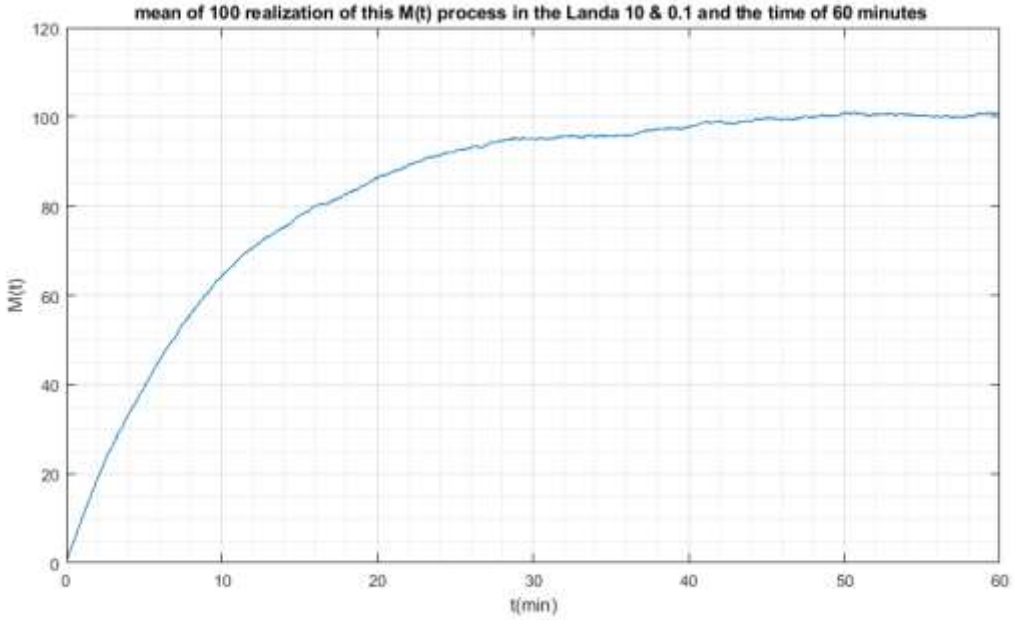
$$\rightarrow n = \frac{10}{0.1} = 100$$

To confirm the calculations, run the system for 600 minutes and see the results, which is completely in confirmation of the theoretical results.

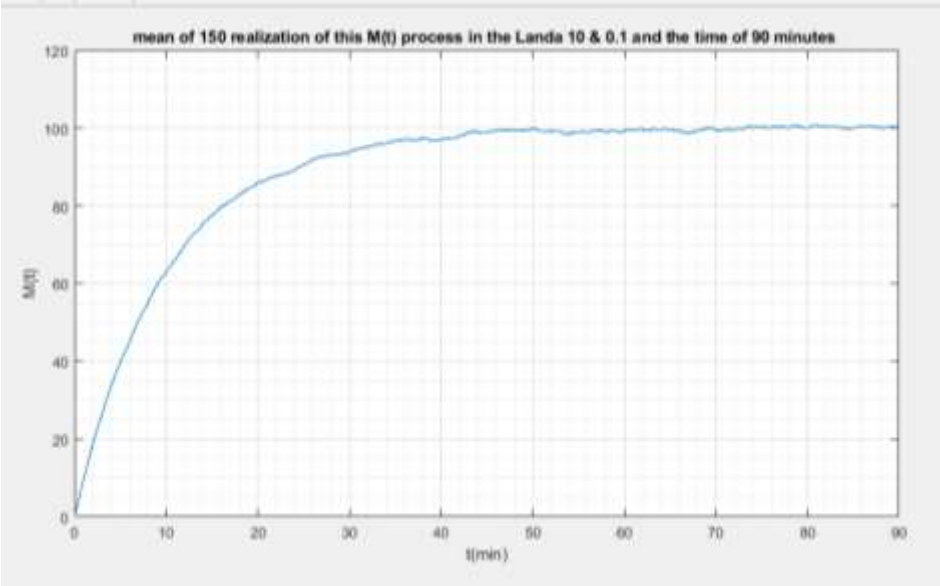


5.

I ran the code for 600 realizations from the system and averaged them on the chart for up to 60 minutes, which, as you can see, has become very functionally smooth, eventually reaching about 100.



To ensure the answer obtained in the previous sections, I drew a graph for the time of 90 minutes and 150 realizations of the process, so that we can clearly see the rate of simultaneous conversations in the channel.



Question 2

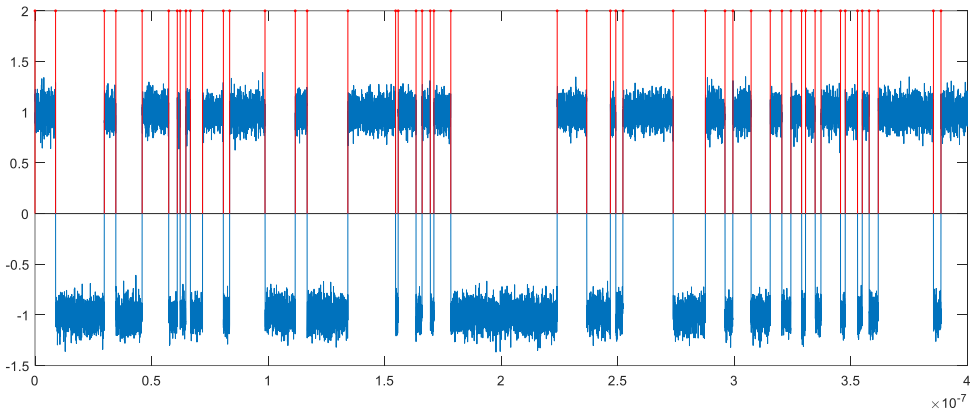
6.

To do this, just take the function transitions between -1 and 1 and specify these boundaries.

To do this, I used the following MATLAB code and you can see a display of these transfers below the code

```
load("Q2_DATA.csv");
format long
a = Q2_DATA;
time = a(:,1);
x = a(:,2);
plot(time,x)
xlim([0,4*10^-6])
%zero_crossing
zero_crossing(1) = 0;
k = 2;
for i = 2: length(x)
    if x(i)<0
        if x(i-1)>0
            zero_crossing(k) = time(i);
            k =k+1;
        end
    end
    if x(i)>0
        if x(i-1)<0
            zero_crossing(k) = time(i);
            k =k+1;
        end
    end
end
hold on;
stem(zero_crossing,ones(1,length(zero_crossing))*2,'r.')
xlim([0,4*10^-7])
mean_length_of_pulse = sum(diff(zero_crossing))...
    /length(diff(zero_crossing))
```

Now graph:



The values of the pulse lengths are easily obtained by calculating the difference between the red dots

```
mean_length_of_pulse =

   1.076944444444445e-08
```

We can use 3 methods to estimate Lambda:

1. Calculate the inverse value of the average pulse length

2. Calculate the number of segment pulses per time

3. Calculate the standard deviation of the data

```
lambda_est = 1/mean_length_of_pulse

lambda_est = 145/time(end)
```

so we have :

```
lambda_est =

    9.285530049006963e+07


lambda_est =

    9.154040404040404e+07
```

7.

Given the description of the problem of a random process, it is easy to see that we have a random process of a semi random telegraph signal.

First of all, we are going to find the expected value of this stochastic process:

Semi random telegraph signal

$$\mathrm{x(t)} = \begin{bmatrix} 1 & if\ n_{(0,t)} \text{ is even} \\ -1 & if\ n_{(0,t)} \text{ is odd} \end{bmatrix}$$

$$E\{X(t)\} = 1 \times P_{(x_{(t)}=1)} - 1 \times P_{(x_{(t)}=-1)}$$

$$1 * \sum_{k=0}^{\infty} P\{n_{(0,t)} = 2k\} + (-1) \sum_{k=0}^{\infty} P\{n_{(0,t)} = 2k+1\} =$$

$$for\ one\ of\ the\ terms\ \rightarrow \sum_0^{\infty} \frac{e^{-\lambda t}(\lambda t)^{2k}}{(2k)!} = e^{-\lambda t} \sum_0^{\infty} \frac{(\lambda t)^{2k}}{(2k)!} = e^{-\lambda t}\left[1 + \frac{(\lambda t)^2}{2!} + \frac{(\lambda t)^4}{4!} \cdots\cdots\right]$$

$$= e^{-\lambda t}\cosh(\lambda)$$

So we will have

$$E\{x(t)\} = e^{-\lambda t}\cosh(\lambda) - e^{-\lambda t}\sinh(\lambda) = e^{-\lambda t}(\cosh(\lambda) - \sinh(\lambda))$$

$$E\{x(t)\} = e^{-\lambda t}(e^{-\lambda t}) = e^{-2\lambda t}$$

Now in this problem we have parameter that called I for that we have:

$$E\{I(t)\} = E\left\{\frac{\int_0^t x(T)dT}{t}\right\} = \frac{\int_0^t E\{x(T)\}dT}{t} = \frac{\int_0^t e^{-2\lambda T}dT}{t}$$

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

So obviously we show question requirement:

$$E\{I(t)\} = \frac{\left(1 - e^{-2\lambda t}\right)}{2\lambda t}$$

8.

For this part first of all we must calculate the integral of x through zero to t and then divide it to t and then solve the equation that we have to calculate the lambda value:

```
dt = diff(time(1:843));
xave = x(1:842);
a = dt.*xave;

integration = sum(a)/time(843)

lambda = (1-integration)/(2*time(843))

syms f(x)
f(x) = ((1-exp(-x))/x)-integration;
sol = vpasolve(f)
a = sol/(2*time(843))
```

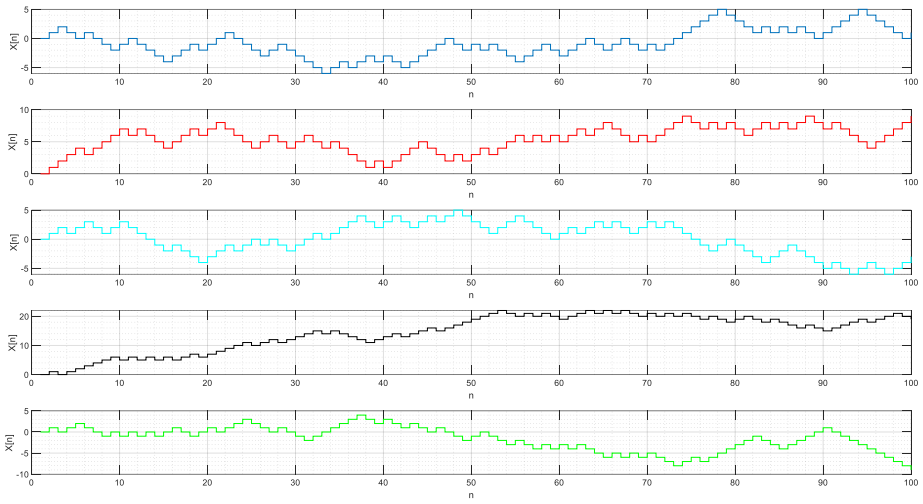
```
sol =

14.982618620782676867738943712167


lambda =

449334771.49660142927181950732999
```

Question 3

9. I made random walk patterns in a matrix and plotted them for 5 runs. I used the rand function to construct each and set the values above 0.5 to 1 and the values below it to -1.

See the MATLAB code and the resulting graph below

```
random_walk = zeros(100,5)
for i = 2:100
    a = rand(1,5)
    random_walk(i,:) = random_walk( i-1,:)  -1*(a<0.5) +1*(a>0.5);
end
```



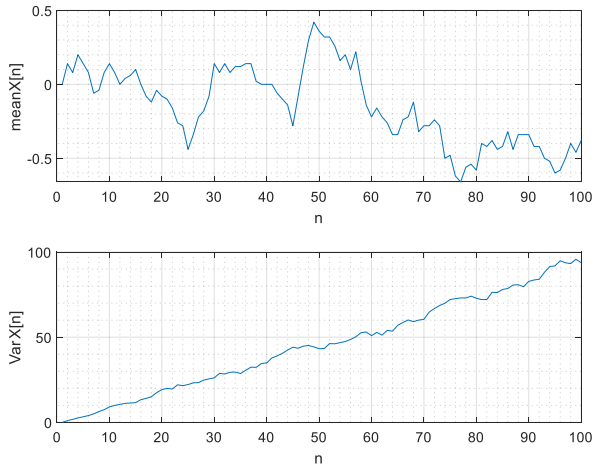
10.for this part first of all I made 100 random walk vectors and i make two vectors from that

1. mean values

2. standard deviation values

```
random_walk = zeros(100,100);
for i = 2:100
    a = rand(1,100);
    random_walk(i,:) = random_walk( i-1,:)  -1*(a<0.5) +1*(a>0.5);
end
mean = sum(random_walk,2)/100;
var_ran = var(random_walk,0,2);
```

and plot this to vectors to see who it works



It is clear that the mean value according to it is a value close to zero, which we also knew by theory, which is equal to 0.

We also know that the value of variance is proportional to n, which is almost the same as the line of origin of the passage on the obtained figure.

11.

In the same way that we mentioned earlier, I made 500 random walk samples and determined the number of cases that were less than 8, and thereby obtained the probability.

```
andom_walk = zeros(100,500);
for i = 2:100
    a = rand(1,500);
    random_walk(i,:) = random_walk( i-1,:)  -1*(a<0.5) +1*(a>0.5);
end
k = 0
for i = 1:500
    if random_walk(100,i)< 8
        k = k + 1;
    end
end
k/500
```

```
ans =

   0.788000000000000
```

$$p\{x[n] \leq m\} = G(\frac{m}{\sqrt{n}})$$

From above equation we have:

$$p = 0.758$$

Results of theory & simulation are so close to each other.

End