Exercise 1:

1. The task is to generate a vector (array) that results from applying the sine function to a given set of values. Specifically, for the values:

   0 0.7000 1.4000 2.1000 2.8000 3.5000 4.2000 4.9000 5.6000 6.3000 7.0000 7.7000

   The results:

   0 0.6442 0.9854 0.8632 0.3350 -0.3508 -0.8716 -0.9825 -0.6313 0.0168 0.6570 0.9882

   Using:

```
b = sin([0:0.7:8]);
```

2. The task is to generate a vector (array) based on the function x3+3x+5x^3 + 3x + 5x3+3x+5 for a given set of values. Specifically, for the values:

   0 0.3000 0.6000 0.9000 1.2000 1.5000 1.8000 2.1000 2.4000 2.7000 3.0000 3.3000 3.6000 3.9000 4.2000 4.5000 4.8000 5.1000 5.4000 5.7000 6.0000 6.3000 6.6000 6.9000 7.2000 7.5000 7.8000 8.1000 8.4000 8.7000 9.0000

   The results

   5.0000 5.9270 7.0160 8.4290 10.3280 12.8750 16.2320 20.5610 26.0240 32.7830 41.0000 50.8370 62.4560 76.0190 91.6880 109.6250 129.9920 152.9510 178.6640 207.2930 239.0000 273.9470 312.2960 354.2090 399.8480 449.3750 502.9520 560.7410 622.9040 689.6030 761.0000

   Using:

```
k = [0:0.3:9];
b = power(k,3) + 3*k +5;
```

3. Generating a Matrix with Random Values: To create a matrix of size **M** × **N** or **N** × **N** with random values in the range [0,1][0,1][0,1], we use the function: `rand(M,N) & rand(N).`

A)
```
A = rand(4);
```
```
    0.3674    0.9133    0.3354    0.1068
    0.9880    0.7962    0.6797    0.6538
    0.0377    0.0987    0.1366    0.4942
    0.8852    0.2619    0.7212    0.7791
```
```
a = A(:,[2 3]);
```
```
a =

    0.9133    0.3354
    0.7962    0.6797
    0.0987    0.1366
    0.2619    0.7212
```

B)
```
A = rand(4,5);
```
```
A =

    0.7150    0.6987    0.5000    0.6177    0.1829
```

```
0.9037    0.1978    0.4799    0.8594    0.2399
0.8909    0.0305    0.9047    0.8055    0.8865
0.3342    0.7441    0.6099    0.5767    0.0287
  b = A([1 3],:);
  b =

    0.7150  0.6987  0.5000  0.6177  0.1829
    0.8909  0.0305  0.9047  0.8055  0.8865
```

Γ)
```
A = rand(4,5);
  A =

0.4899    0.5005    0.0424    0.8181    0.6596
0.1679    0.4711    0.0714    0.8175    0.5186
0.9787    0.0596    0.5216    0.7224    0.9730
0.7127    0.6820    0.0967    0.1499    0.6490
  c = A(:,2:4);
  c =

    0.5005  0.0424  0.8181
    0.4711  0.0714  0.8175
    0.0596  0.5216  0.7224
    0.6820  0.0967  0.1499
```

Δ)
```
A = rand(4,5);
  A =

0.8003    0.0835    0.8314    0.5269    0.2920
0.4538    0.1332    0.8034    0.4168    0.4317
0.4324    0.1734    0.0605    0.6569    0.0155
0.8253    0.3909    0.3993    0.6280    0.9841
  d = A([2 4],:);
  d =

    0.4538  0.1332  0.8034  0.4168  0.4317
    0.8253  0.3909  0.3993  0.6280  0.9841
```

Ε)
```
A = rand(5,6);
  A =

0.1672    0.3395    0.2691    0.9831    0.6981    0.1711
0.1062    0.9516    0.4228    0.3015    0.6665    0.0326
0.3724    0.9203    0.5479    0.7011    0.1781    0.5612
0.1981    0.0527    0.9427    0.6663    0.1280    0.8819
0.4897    0.7379    0.4177    0.5391    0.9991    0.6692
  e = A([1 3 5],:);
  e =

    0.1672  0.3395  0.2691  0.9831  0.6981  0.1711
    0.3724  0.9203  0.5479  0.7011  0.1781  0.5612
    0.4897  0.7379  0.4177  0.5391  0.9991  0.6692
```

4. A script is required to convert the energy gap of a semiconductor into the corresponding wavelength of emitted radiation and vice versa, depending on the user's input.

   The relationship between the two is:

$$wavelength = \frac{1240.8}{energy}$$

And the code.

```
prompt = 'Do you want conversion from eV to nm  OR  from nm to eV?
(type eV or nm)';
str = input(prompt,'s');
if strcmp(str,'eV')
    prompt = 'What is the energy gap in [eV]?';
    energy = input(prompt);
    wavelength = 1240.8/energy;
    disp('the wavelength is ')
    disp(wavelength)
    disp(' nm')
else
    prompt = 'What is the wavelength in [nm]?';
    wavelength = input(prompt);
    energy = 1240.8/wavelength;
    disp('the energy is ')
    disp(energy)
    disp(' eV')
end
```

If we test the script with real values, we obtain:

Do you want conversion from eV to nm OR from nm to eV? (type eV or nm)nm
What is the wavelength in [nm]?433.57
the energy is
2.8618
eV

And vice versa:

Do you want conversion from eV to nm OR from nm to eV? (type eV or nm)eV
What is the wavelength in [nm]?2.86
the wavelength is
433.8462
Nm

Exercise 2:

Consider a box that is divided into two equal parts by a wall.

Inside the box, there are nanoparticles.

The wall dividing the box has a hole that allows nanoparticles to pass through.

Only one nanoparticle passes through the hole per unit of time (one second).

Initially, the total number of nanoparticles NN is located entirely in one compartment of the box, and the hole is closed.

When we open the hole, the nanoparticles move randomly back and forth until equilibrium is reached.

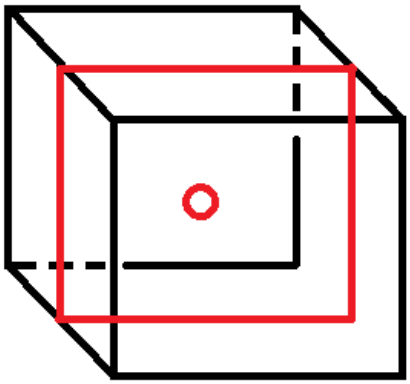
Statistically, the number of nanoparticles in one compartment (A) of the box as a function of time is given by the equation:

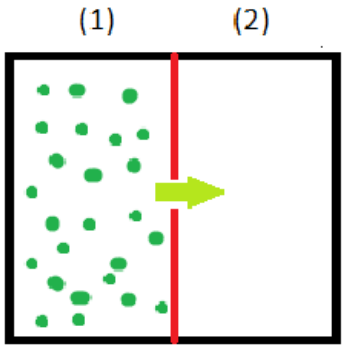
$$N_\alpha = \frac{N}{2}\left(1 + e^{\frac{-2t}{N}}\right)$$

Task:

1. **Simulate the problem using MATLAB code** and plot the change in the number of nanoparticles over time.
2. The **total simulation time is 100,000 seconds**.
3. On the **same graph**, also plot the theoretical equation given above.
4. Perform the simulation for **total nanoparticle numbers** N=1000, N=10,000N, and N=100,000N.
5. **Record the CPU execution time** required for the simulation using your MATLAB code.
6. **Specify the processor model** of the computer on which you ran the simulation.

The problem is schematically represented as follows:





The theoretical equation for the change in the nanoparticle population as a function of time is given in the problem statement:

$$N_\alpha = \frac{N}{2}\left(1 + e^{\frac{-2t}{N}}\right)$$

Where $t \in [0, 100000]$ s, N_α takes values according to the above equation.

To derive experimental results, we assume that for each second, only one nanoparticle passes through the opening until equilibrium is reached, i.e., $N_1 = N_2$. From the moment equilibrium is achieved, for the remainder of the simulation time, the populations remain constant. The experimental equation derived is:

$$N_a = \begin{cases} N - t\,, & t \in [0\,, N/2] \\ \frac{N}{2}, & t \in \left[\frac{N}{2} + 1\,, 100000\right] \end{cases}$$

These equations can be quickly computed for the required range of values using MATLAB with the following code:

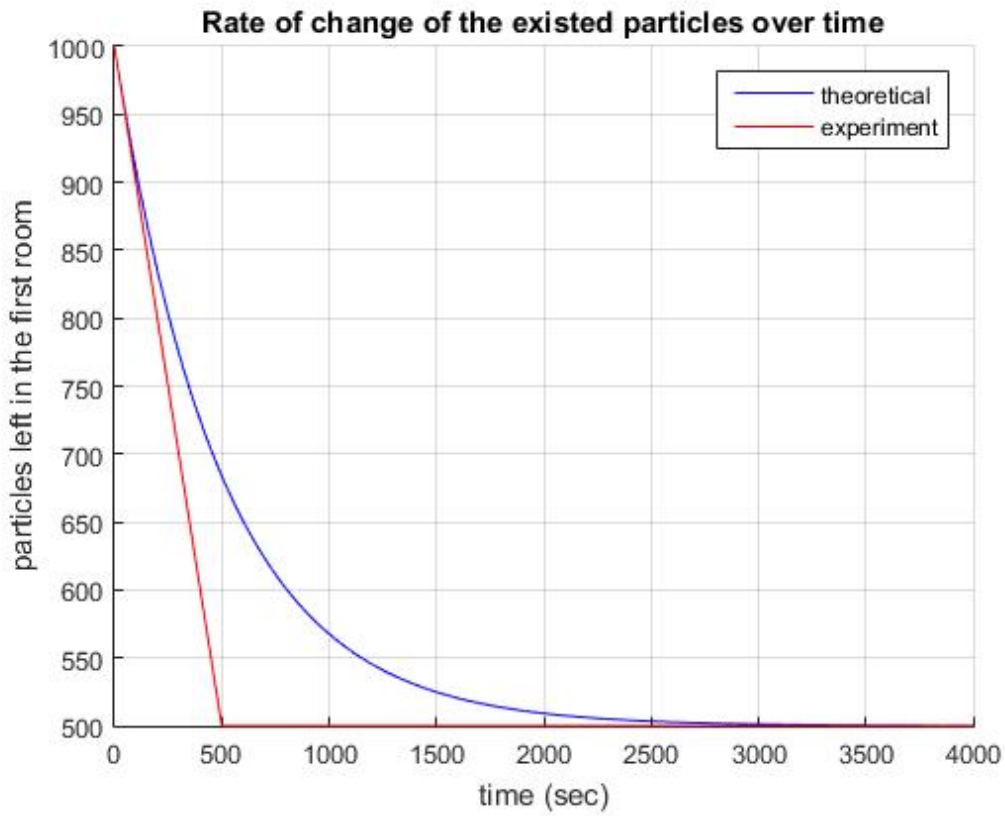
```
start = tic;
N = 1000;
t =  0:100000 ;

%Na_theo is the theoretical Na
Na_theo = N*(1+exp(-2*t/N))/2 ;
%Na_exp is the experiment Na
Na_exp = N - t(1:N/2);
k = zeros(1,100000-N/2+1)+ N/2;
Na_exp = [Na_exp , k];

%comparison plot
plot (t,Na_theo,'b',t,Na_exp,'r');
xlabel('time (sec)');
ylabel('particles left in the first room');
title('Rate of change of the existed particles over time');
legend('theoretical','experiment');
grid;
box;
```

```
%we can zoom in the plot using the command
%axis ([0 t_max N/2 N ]);
%reducing the t_max depending on which second we want to be able to
%see
time = toc(start);
```

The commands start = tic; ... time = toc(start); measure and record the time required for the program to execute and generate the plot. The computer processor used is an Intel i7.

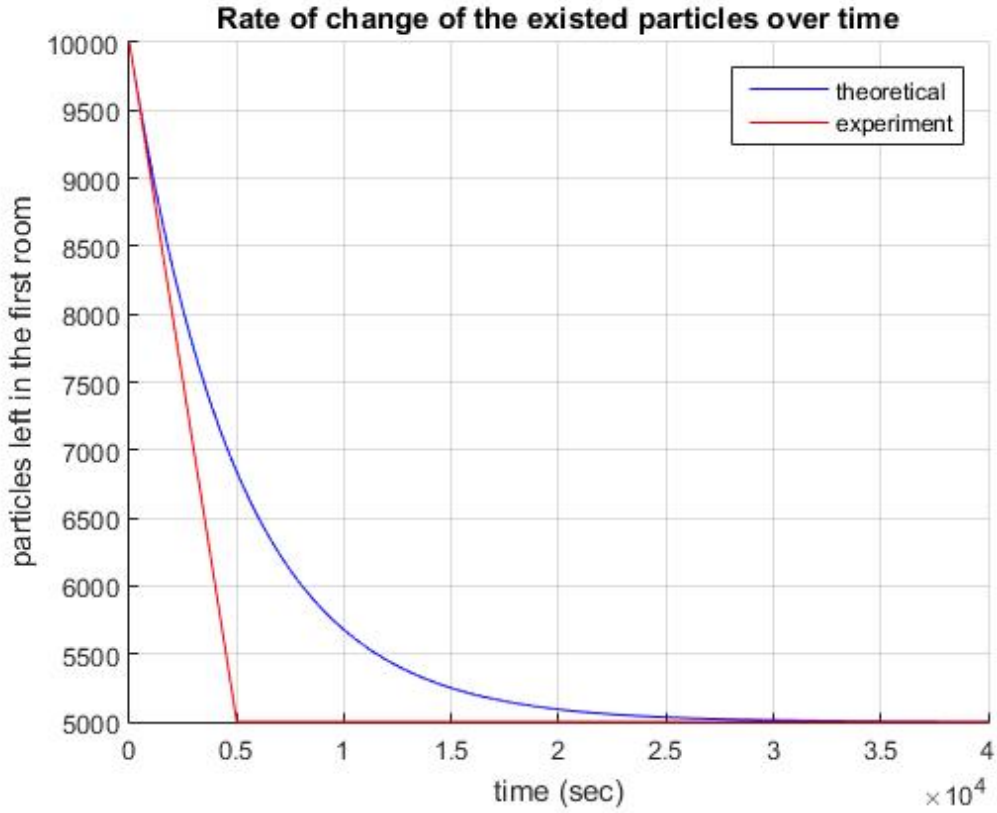
For different values of NNN:

- Για N = 1000



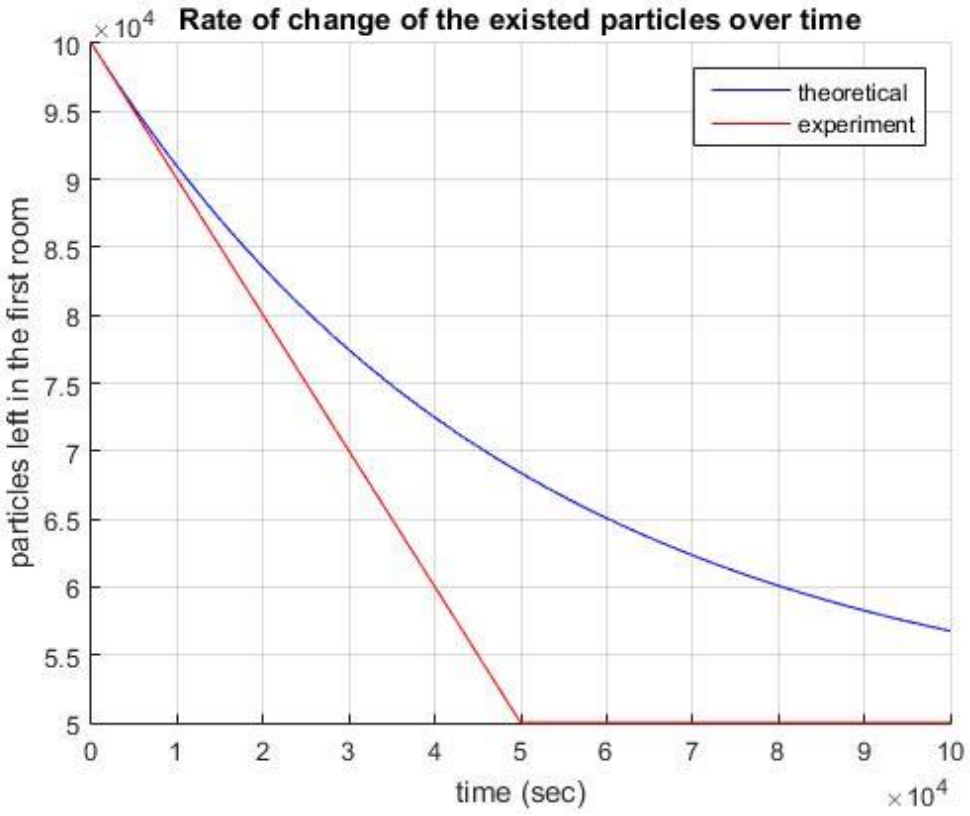
Execution time : 0.112214564961347 s

- N = 10000



Execution time : 0.357866764755508 s

- N = 100000



Execution time : 0.132572652179678 s

From all the graphs, we observe that:

1. The slope of the experimental equation is significantly steeper and deviates from the theoretical model as time increases (before equilibrium is reached).
2. As the number of nanoparticles increases, the slope of the graphs becomes less steep, meaning the system takes longer to reach equilibrium.
3. In the last case (N=100000), equilibrium is not reached before the simulation ends.
4. Execution times do not show significant differences across different values of NNN.

Second Part:

Repeat the above process, but now assume that the total number of nanoparticles **N is NOT initially located in one compartment**. Instead, they are **distributed** across both compartments, so that:

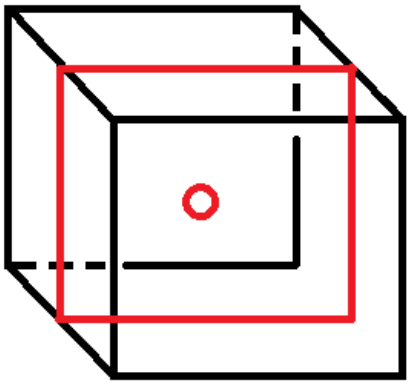
$$N_1 + N_2 = N$$

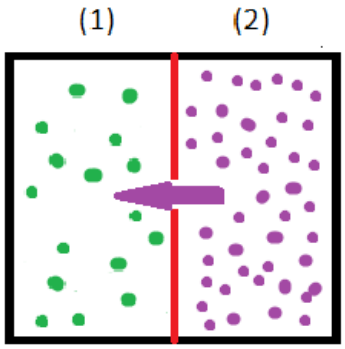
The populations N1N_1N1 and N2N_2N2 will be determined as follows:

1. Your MATLAB code will **read the first name and last name** of each student.
2. Each letter of the name will be **converted into ASCII numbers**.
3. The sum of all ASCII values of the **first name** is stored as the variable **ON**.
4. The sum of all ASCII values of the **last name** is stored as the variable **EP**.
5. The program will determine **which value (ON or EP) is larger** and calculate the ratio of the smaller number to the larger one.
6. This ratio will be **equal to N1/N2**

Then, repeat the previously mentioned study using this new distribution of nanoparticles.

In the second part of the assignment, the problem is modified so that nanoparticles are **unevenly distributed** between the two compartments of the box. The updated problem is represented schematically as follows:





The populations N1 and N2 are computed using a MATLAB script according to the problem's instructions:

```
%Compute N1 and N2
name= 'george';
name = double(name);
ON = sum(name);

lastname= 'vlachos';
lastname = double(lastname);
EP = sum(lastname);

%The computation of rate makes it always minor to 1
if EP == max(ON,EP)
    rate = ON/EP;
else
    rate = EP/ON;
end

%We set rate = N1/N2 so N1<N2 is always valid
N2 = round(N/(rate+1));
N1 = N - N2;
```

We assume that the particles remaining in **Compartment 1** have a population of N1N_1N1, while those in **Compartment 2** have a population of N2N_2N2. It is logical to conclude that the particles in **Compartment 2**, which are more numerous, will move toward **Compartment 1**.

The relationships for the two populations as a function of time are:

$$N_{a1} = \begin{cases} N_1 + t\,, & t \in \left[0\,, \frac{N}{2} - N_1\right] \\ \frac{N}{2}, & t \in \left[\frac{N}{2} - N_1 + 1\,, 100000\right] \end{cases}$$

$$N_{a2} = \begin{cases} N_2 - t\,, & t \in \left[0\,, \frac{N}{2} - N_1\right] \\ \frac{N}{2}, & t \in \left[\frac{N}{2} - N_1 + 1\,, 100000\right] \end{cases}$$

The corresponding theoretical equations are:

$$N_{a1} = N - \frac{N}{2}\left(1 + e^{\frac{-2(t+t1)}{N}}\right)$$

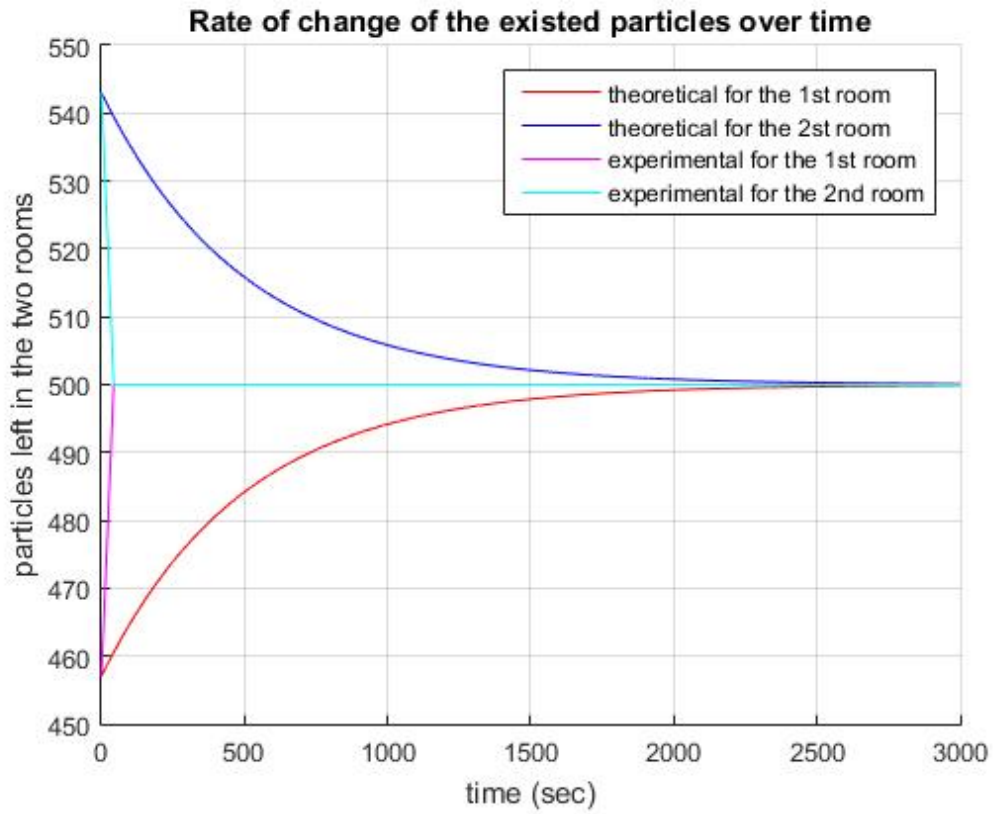
$$N_{a2} = \frac{N}{2}\left(1 + e^{\frac{-2(t+t1)}{N}}\right)$$

Where $t_1 = -\frac{N}{2}\ln(\frac{2N2}{N} - 1)$

The following script calculates the values for the requested variables:

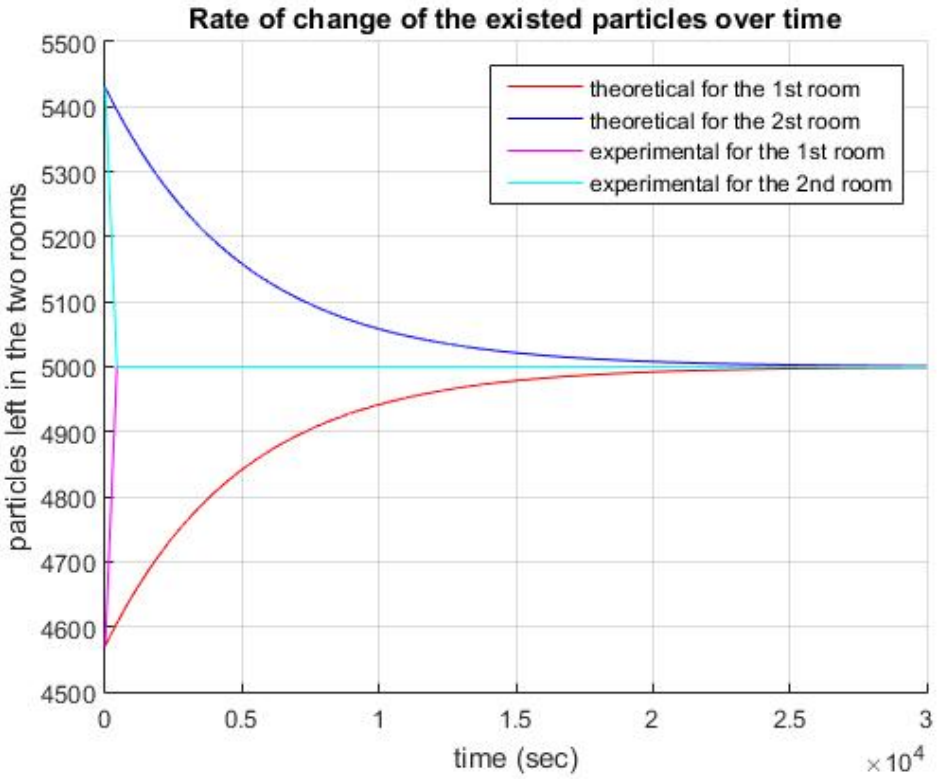
```
%Na_exp1 are the particles left in space 1
Na_exp1 = N1 + t(1: N/2-N1);
k1 = zeros(1,100000-(N/2 - N1)+1)+ N/2;
Na_exp1 = [Na_exp1 , k1];
%Na_exp2 are the particles left in space 2
Na_exp2 = N2 - t(1: N/2-N1);
k2 = zeros(1,100000-(N/2 - N1)+1)+ N/2;
Na_exp2 = [Na_exp2 , k2];
%Na_theo1 kai Na_theo2 express the theory for each space of the box
t1 = -N*log(2*N2/N -1)/2;
Na_theo1 = -N*(1+exp(-2*(t+t1)/N))/2 +N;
Na_theo2 = N*(1+exp(-2*(t+t1)/N))/2;


%Comparison plot
plot (t,Na_theo1,'r',t,Na_theo2,'b',t,Na_exp1,'m',t,Na_exp2,'c'
);
xlabel('time (sec)');
ylabel('particles left in the two rooms');
title('Rate of change of the existed particles over time');
legend('theoretical for the 1st room','theoretical for the 2st
room','experimental for the 1st room','experimental for the 2nd room'
);
grid;
box;
time = toc(start);
```

- N = 1000



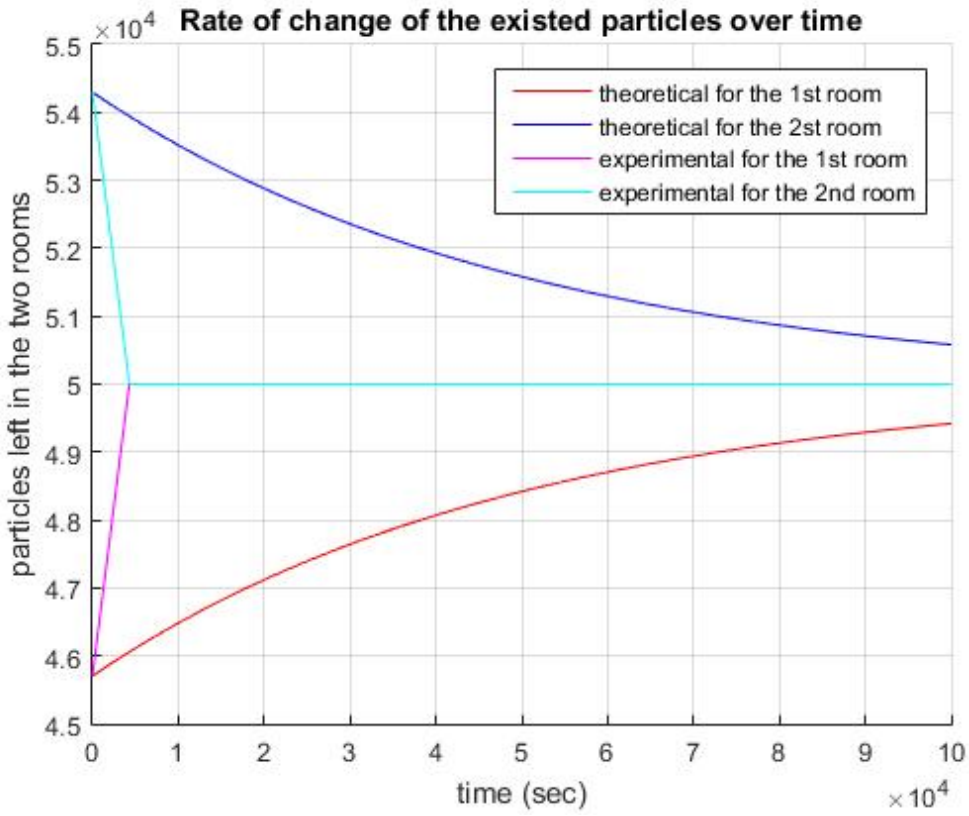
Execution time : 7.471486760874688 s

- N = 10000



Execution time : 8.43151937323832 s

- N = 100000



Execution time : 7.139882983029583 s

In the first two graphs, zooming was applied using the axis() command to better visualize the results, following the explanation given earlier.

From all the graphs, we observe that:

1. The slope of the experimental curve is significantly steeper and deviates from the theoretical curve as time increases (before equilibrium is reached).
2. The ratio of slopes for the curves corresponding to each compartment is **preserved**, and the curves are **mirror images** of each other.
3. As the number of nanoparticles **increases**, more time is required to achieve equilibrium.

Exercise 3 (Weibull):

Calculation of Wind Energy Potential Using the Weibull Distribution

1. Objective

To estimate the wind energy potential using the Weibull distribution and construct a wind rose diagram.

2. Weibull Distribution

For the calculation of the available wind energy potential in a region, knowledge of the mean wind speed alone is not sufficient. Detailed information about the probability distribution of various wind speed values over time is required. Existing experimental data show that wind characteristics in temperate regions and at heights up to 100 meters from the ground are satisfactorily described by the Weibull distribution. For this reason, the Weibull distribution is the most widely used analytical probability density distribution for wind energy potential calculations.

The Weibull distribution expresses the probability that the wind speed V falls within the range V - dV/2 and V + dV/2 and is given by the equation:

$$f(V) = \frac{k}{C}\left[\frac{V}{C}\right]^{k-1} e^{-\left(\frac{V}{C}\right)^{k}} \qquad (1)$$

(1) Weibull probability density function equation

Where:

C: (scale/location parameter) related to the mean wind speed.

k: (shape parameter) inversely proportional to the variance of wind speed relative to the mean speed.

To determine the wind speed duration curve, we need to specify the time period during which the measured speed exceeds a predetermined value. For the Weibull distribution, the duration curve can be derived from the total probability function, expressed as:

$$F(V \le V_0) = 1 - e^{-\left(\frac{V_0}{C}\right)^{k}} \qquad (2)$$

(2) Cumulative probability function equation

This equation is complementary to the duration curve and provides the probability F that the wind speed V is less than a given value Vo.

After logarithm transformation, equation (2) can be rewritten as:

$$\ln(-\ln(1 - F(V \le V_0))) = -k \ln C + k \ln V_0 \quad (3)$$

(3) Linear form equation

Thus, the parameters k and C corresponding to measurements in a region can be determined using the least squares method, fitting a line to the measurements as follows:

$$Y = A + B \cdot X \quad (4)$$

Where:

$$\mathrm{Y} = \ln(-\ln(1 - F(V \leq V_0)))$$

$$\mathrm{X} = \ln V_0$$

After computing A and B using the least squares method on the existing wind data, the parameters k and C can be calculated using:

$$C = e^{-\frac{A}{B}}$$

$$k = B$$

3. Wind Rose Diagram

The wind rose diagram is a polar plot that displays the relative frequency of wind directions at a location. In the diagram, concentric circles represent the relative frequency of each wind direction, and color shading is often used to indicate wind speed distribution per direction. The percentage of calm conditions is displayed at the center of the diagram.

4. Data

Wind data from a meteorological station in the Aegean region will be used for calculations. The data files are in Excel format and contain wind speed measurements recorded every three hours over multiple years.

5. Required Actions

a. Each student downloads a data file using the hyperlink provided in the attached Excel file. The same Excel file specifies the years for which each student must perform calculations.

b. Develop a MATLAB program that performs the following steps:

(i) Reads the data file.

(ii) Selects only the data for the years specified in the Excel file.

(iii) Fits a Weibull distribution to the data according to the equations above.

(iv) Computes and displays the Weibull distribution parameters C and k.

(v) Plots the wind speed distribution and the Weibull curve (examples provided at the end of the instructions).

(vi) Generates the wind rose diagram (examples provided at the end of the instructions).

c. The assignment must be submitted via email by January 22. Specifically, students must submit three files:

- A .m file containing the MATLAB program.

- Two .jpg files displaying the generated plots.

In the SUBJECT line of the email, write: CP2017.

d. The assignment is individual. Submitting identical files will result in rejection.

Resulted Diagrams: