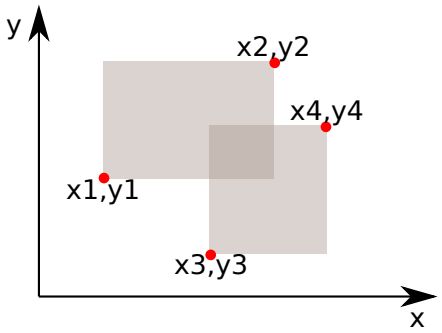
Assignment 2G Box area

Two rectangular boxes are defined by the coordinates of their lower left and upper right corners as seen below.



The area of the two boxes are given by

$$A_1 = (x_2 - x_1)(y_2 - y_1), \qquad A_2 = (x_4 - x_3)(y_4 - y_3). \tag{2.6}$$

The area of the intersection between the two boxes is given by

$$A_o = \max[0, \min(x_2, x_4) - \max(x_1, x_3)] \cdot \max[0, \min(y_2, y_4) - \max(y_1, y_3)]. \tag{2.7}$$

Problem definition

Create a function named `boxArea` that takes as input the coordinates defining the two boxes as a vector $[x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4]$ as well as a string `area` that specifies what to compute (written exactly as below). The function must return the computed area.

`area`	Area to compute
`Box1`	Area of the first box.
`Box2`	Area of the second box.
`Intersection`	Area of the overlap between the boxes.
`Union`	The sum of the areas of the two boxes minus the area of the overlap.

Solution template

```
function A = boxArea(boxCorners, area)
% Insert your code here...
% Use a switch statement to choose between the different areas to compute.
switch area
    case 'Box1'
        % Insert code to compute area of box one
        % A = ...
    case 'Box2'
        % Insert code to compute area of box two
        % A = ...
    case 'Intersection'
        % Insert code to compute area of intersection
        % A = ...
    case 'Union'
        % Insert code to compute area of union
        % A = ...
end
```

Input	
`boxCorners`	Corners of the boxes (vector of length 8).
`area`	Which area to compute (string).
Output	
`A`	The computed area (number).

Example

Consider the following coordinates: $x_1 = 5, x_2 = 20, x_3 = 14, x_4 = 25, y_1 = 12, y_2 = 23, y_3 = 5, y_4 = 17$. The input `boxCorners` would be the vector $[5, 20, 14, 25, 12, 23, 5, 17]$. If the input string `area` is equal to `intersection` the area of the overlap must be computed: $A_o = \max[0, \min(20, 25) - \max(5, 14)] \cdot \max[0, \min(23, 17) - \max(12, 5)] = \max[0, 20 - 14] \cdot \max[0, 17 - 12] = 6 \cdot 5 = 30$.

Example test case

Remember to thoroughly test your code before handing it in. You can test your solution on the example above by running the following test code and checking that the output is as expected.

Test code	Expected output
`disp(boxArea([5,20,14,25,12,23,5,17], 'Intersection'))`	`30`

Hand in on CodeJudge

The assignment must be handed in on CodeJudge.

2G

```
function A = boxArea(boxCorners, area)
% Use a switch statement to choose between the different areas to compute.
x = boxCorners;
switch area
case 'Box1'
    index = [1,2,5,6];
    A = AA(x(index));
case 'Box2'
    index = [3,4,7,8];
    A = AA(x(index));
case 'Intersection'
    A = max(0, (min(x(2),x(4))-max(x(1),x(3))))*max(0, (min(x(6),x(8))-max(x(5),x(7))));
case 'Union'
    index1 = [1,2,5,6];
    index2 = [3,4,7,8];
    A = AA(x(index1)) + AA(x(index2)) - (max(0, (min(x(2),x(4))-max(x(1),x(3))))*max(0, (min(x(6),x(8))-max(x(5),x(7)))));
end
end


function X = AA(a)
    X = (a(2)-a(1))*(a(4)-a(3));
end
```

Optional challenge 3F Football goal tracker

In the 2013–2014 Premier League season was the first seasonuse Hawk-Eye system for tracking the ball to detect whether a goal has been scored. The system uses a number of high-performance cameras to track the ball from different angles.

We might imagine a simple version of the system that, once the ball is kicked, tracks whether or not the ball would pass the goalline if it continued in a straight line. Suppose the system is already able to compute the point where the ball is kicked and the direction vector of the kick.

The standard measurement of a football field is 105 [m] × 68 [m] and the goal is 7.32 [m] wide. Suppose the ball is shot from a point $p = [x, y]$ on the field and goes in a straight line in some direction given by a vector $v = [v_x, v_y]$. (The magnitude of the vector indicates the speed of the ball, which we will not use in this exercise.)

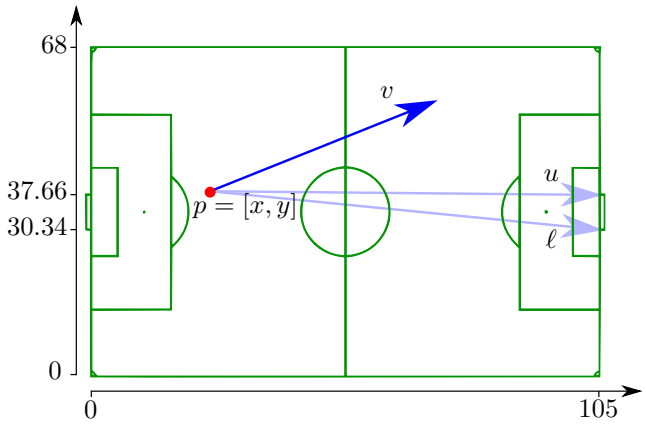


Figure 3.2: A football field.

Problem definition

Create a function that tells whether or not the ball will pass the goalline (in either of the two goals) if it continues from the initial position in a straight line along the direction vector.

Solution template

```
function score = computePassesGoalLine(point, directionVector)
% Insert your code here
```

Input	
`point`	The (x, y) coordinates of the initial position of where the football is kicked from (vector).
`directionVector`	The vector describing the direction in which the football is kicked.
Output	
`score`	A boolean telling whether or not the ball will pass the goalline.

Example

Assume that the position of the ball is given by the point $p = [x, y] = [30, 20]$ and the direction is given by the vector $v = [v_x, v_y] = [10, 2]$.

Firstly, we note that since the x-component of the direction vector is positive, in this case the ball can only pass the goal line in the goal on the right. The goal posts of the goal on the right are at the positions $[105, 30.34]$ and $[105, 37.66]$.

The ball will cross x-coordinate $x_{\text{goal}} = 105$ at the right at position $p + \alpha v$ where α is a number denoting how far we should travel along v. Since we must have $x + \alpha v_x = 105$ we can compute α as

$$\alpha = \frac{x_{\text{goal}} - x}{v_x} = \frac{105 - 30}{10} = 7.5. \tag{3.7}$$

The y-coordinate where the ball passes the goal line is then given by

$$y_{\text{goal}} = y + \alpha v_y = 20 + 7.5 \cdot 2 = 35. \tag{3.8}$$

Then the condition to see if the ball ends up in the goal is equivalent to testing if

$$30.34 < y_{\text{goal}} < 37.66, \tag{3.9}$$

which is true for $y_{\text{goal}} = 35$ so the return value `score` should be set true.

Example test case

Remember to thoroughly test your code before handing it in. You can test your solution on the example above by running the following test code and checking that the output is as expected.

Test code	Expected output
`disp(computePassesGoalLine([30, 20], [10, 2]))`	`1`

Hand in on CodeJudge

This challenge can be be handed in on CodeJudge.

3F

```
function score = computePassesGoalLine(point, directionVector)
    v = directionVector;
    p = point;
    if v(1) > 0
        yf = p(2) + (v(2)/v(1)*(105 - p(1)));
        if (yf < 37.66 && yf > 30.34)
            score = 1;
        else
            score = false;
        end
    elseif v(1) < 0
        yf = p(2) - (v(2)/v(1)*p(1));
        if (yf < 37.66 && yf > 30.34)
            score = 1;
        else
            score = 0;
        end
    else
        score = 0;
    end
end
```

Assignment 4F Removing incomplete experiments

You are working on a data set from a series of experiments, each of which consists of three parts. The experiments have been performed in a randomized order. Each experiment has been assigned an experiment-number and a part-number, which are joined into one decimal number called an id-number. The id-number is formed by separating the experiment-number and part-number by a decimal point. For example, the experiment number 17 part 3 has been assigned the id-number 17.3. Note, that you can compute the experiment-number from the id-number by rounding down to the nearest integer.

You notice that due to errors, for some of the experiments all three parts have not been completed. For further analysis, you need to exclude all experiments where one or more parts are not available. You can safely assume that if there are 3 id-numbers with the same experiment-number, the experiment is complete.

Problem definition

Create a function that takes as an input a vector of id-numbers and returns a vector of id-numbers where all incomplete experiments have been removed. The id-numbers that are not removed must remain in the same order as in the original vector.

Solution template

```
function idComplete = removeIncomplete(id)
% Insert your code here
```

Input	
`id`	Id-numbers (vector of decimal numbers)
Output	
`idComplete`	Id-numbers of complete experiments (vector of decimal numbers)

Example

Consider the following id-numbers:

1.3 2.2 2.3 4.2 5.1 3.2 5.3 3.3 2.1 1.1 5.2 3.1

In experiment 1, part 2 is missing, and in experiment 4, parts 1 and 3 are missing. Thus, experiment 1 and 4 are incomplete. After removing the incomplete experiments, the result should be:

2.2 2.3 5.1 3.2 5.3 3.3 2.1 5.2 3.1

Example test case

Remember to thoroughly test your code before handing it in. You can test your solution on the example above by running the following test code and checking that the output is as expected.

Test code	Expected output

`disp(removeIncomplete([1.3, 2.2, 2.3, 4.2, 5.1, 3.2, 5.3, 3.3, 2.1, 1.1, 5.2, 3.1]))`	`Columns 1 through 7`
	`2.2000 2.3000 5.1000 3.2000 5.3000 3.3000 2.1000`
	`Columns 8 through 9`
	`5.2000 3.1000`

Hand in on CodeJudge

This assignment must be handed in on CodeJudge.

4F

```
function idComplete = removeIncomplete(id)
    v = fix(id); %Extracts only experiments id.
    M = max(v); %Now I know how many experiments there are!

    %Counting how many parts each experiments has
    % NB: this doesn't work in the command window: v(i) must be converted
    % into integer there!
    C = zeros(1, M);
    for i=1:length(v)
        C(v(i)) = C(v(i)) + 1;
    end

    % The indeces of C correspondent to its elements that are equal to 3
    % are the only completed experiment id ;)!
    ce = find(C == 3);
    ind = ismember(v, ce);
    % ind is a vector as long as v (and therefore id); containing 1
    % (logical) when the correspondant element in v is in ce and so
    % a completed experiment id
    idComplete = id(ind);
end
```

Optional challenge 4G Cluster analysis

Using an image sensor on a satellite, you can measure the reflectance at the ground level in a narrow band of the near-infrared frequency spectrum. It is known that this reflectance is highly sensitive to the type of vegetation; however, there is some measurement noise and the overall magnitude of the reflectance also varies with the time of day, sensor drift, etc. You want to create an automated system that can distinguish between two types of vegetation, trees and grass, which are known to have relatively high and low reflectance respectively.

To distinguish between trees and grass, you decide to use *cluster analysis*. "Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters)." [Wikipedia] In particular, you will use the following algorithm (called k-means) to cluster the reflectance data into two groups.

The clustering algorithm

Given a set of N measurements, $(r_1, r_2, \ldots, r_N)$, we will initially assign the odd-numbered measurements to class 1 and the even numbered measurements to class 2. Then the following two steps are repeated:

- *Update step:* Compute the mean value (average) of the measurements within each cluster.
- *Assignment step:* Assign each measurement to the cluster with the closest mean value. In case of a tie, assign the measurement to cluster 1.

Repeat the above steps until the cluster assignments do not change. It can not be determined in advance how many steps will be needed before the clustering assignment stabilizes.

■ Problem definition

Create a function that takes as an input a vector of reflectance measurements and returns a vector of cluster assignments computed using the algorithm described above.

■ Solution template

```
function clusterAssignments = clusterAnalysis(reflectance)
% Insert your code here
```

Input	
`reflectance`	Reflectance measurements (vector of decimal numbers)
Output	
`clusterAssignments`	Final cluster assignments (vector of numbers, 1 or 2)

■ Example

Consider the following set of reflectance measurements where the color and style of the number indicates the initial cluster assignment,

1.7 *1.6* **1.3** *1.3* **2.8** *1.4* **2.8** *2.6* **1.6** *2.7.*

In the update step, the two mean values can be computed as

$$m_1 = \frac{1.7 + 1.3 + 2.8 + 2.8 + 1.6}{5} = 2.04, \qquad m_2 = \frac{1.6 + 1.3 + 1.4 + 2.6 + 2.7}{5} = 1.92. \tag{4.3}$$

In the assignment step, each measurement is reassigned to cluster with the closest mean,

1.7 *1.6* *1.3* *1.3* **2.8** *1.4* **2.8** **2.6** *1.6* **2.7.**

In the next update step, the two mean values can be computed as

$$m_1 = \frac{2.8 + 2.8 + 2.6 + 2.7}{4} = 2.725, \qquad m_2 = \frac{1.7 + 1.6 + 1.3 + 1.3 + 1.4 + 1.6}{6} = 1.483. \tag{4.4}$$

In the next assignment step, each measurement is again reassigned to cluster with the closest mean,

1.7 *1.6* *1.3* *1.3* **2.8** *1.4* **2.8** **2.6** *1.6* **2.7**.

Since this assignment is identical to the previous assignment, the algorithm stops. The output of the algorithm is a vector of cluster assignments, which in this example should be

2 2 2 2 1 2 1 1 2 1

Example test case

Remember to thoroughly test your code before handing it in. You can test your solution on the example above by running the following test code and checking that the output is as expected.

Test code	Expected output
`disp(clusterAnalysis([1.7, 1.6, 1.3, 1.3, 2.8, 1.4, 2.8, 2.6, 1.6, 2.7]))`	2 2 2 2 1 2 1 1 2 1

Hand in on CodeJudge

This challenge can be be handed in on CodeJudge.

4G

```
function clusterAssignments = clusterAnalysis(reflectance)
    v = reflectance; % for conciseness
    L = length(v); % turned out to be necessary because used quite often
    i = 1:L; % for indexing... I can't find a better way anywhere!

    % Later on I will need a vector to store the assignments at the
    % beginning of an iteration.
    Ci = ones(1, L);

    % Forming Initial Clusters:
    % cluster 1: odd numbered
    v1 = v(mod(i,2) ~= 0);
    m1 = mean(v1);

    %cluster 2: even numbered
    v2 = v(mod(i,2) == 0);
    m2 = mean(v2);

    % Initial Assignment
    C(mod(i,2) == 0) = 2;
    C(mod(i,2) ~= 0) = 1;

    % Iterative procedure: goes on until cluster assignment remains
    % unchanged
    while (~isequal(C, Ci))
        % right after the first check, the present assignment must be
        % stored for the next check (the previous is no longer significant)
        Ci = C;

        % Reset of the clusters:
        % once the mean values are computed for the first time or after the
        % end of an iteration, I don't really need the clusters themselves
        % but it can happen that when forming new clusters some old values
        % remain (if less values are assigned to a certain clusters
        % compared with the previous iteration) and those values will
        % affect the mean values at the end of the present iteration!
        % Therefore, the 2 clusters are turned into 2 empty vectors

        v1 = [];
        v2 = [];

        i1 = 1;
        i2 = 1;

        % Every single value of reflectance must be compared with the mean
        % values of the 2 clusters and the 2 differences must be compared
        % as well. I can't see the possibility of vectorising.
        % I chose to assign a value that is equally close to both the mean
        % values to the first cluster.
        % Note that the original order of v is never changed so I can
        % easily compute the new assignments.
        for j = 1:L
            if (abs(v(j) - m1) <= abs(v(j) - m2))
                v1(i1) = v(j);
                i1 = i1 + 1;
```

```
                C(j) = 1;
            else
                v2(i2) = v(j);
                i2 = i2 + 1;
                C(j) = 2;
            end
        end

        % once 2 new clusters are formed
        m1 = mean(v1);
        m2 = mean(v2);
    end

    clusterAssignments = C;

end
```

Assignment 5A Temperature conversion

Temperature can be measured in degrees Fahrenheit, degrees Celsius, or Kelvin. We can convert between different units of temperature according to the following equations

	Celsius	Fahrenheit	Kelvin
Celsius	—	$F = 1.8 \cdot C + 32$	$K = C + 273.15$
Fahrenheit	$C = \frac{F - 32}{1.8}$	—	$K = \frac{F + 459.67}{1.8}$
Kelvin	$C = K - 273.15$	$F = 1.8 \cdot K - 459.67$	—

where C is the temperature in degrees Celsius, F is the temperature in degrees Fahrenheit, and K is the temperature in Kelvin.

Problem definition

Implement a function that converts temperatures between the three different units of measurement. The function must take three inputs: A decimal number `T`, which specifies the temperature and two strings, `unitFrom` and `unitTo` which specify the units to convert from and to. The strings must take one of the values `Celsius`, `Fahrenheit`, or `Kelvin`.

> **Hint**
> When comparing two strings, remember to use the `strcmp` function.

Solution template

```
function T = convertTemperature(T, unitFrom, unitTo)
% Insert your code here
```

Input	
`T`	The input temperature (decimal number)
`unitFrom`	The unit of temperature of the input (string, `Celsius`, `Fahrenheit`, or `Kelvin`)
`unitTo`	The unit of temperature of the output (string, `Celsius`, `Fahrenheit`, or `Kelvin`)
Output	
`T`	The converted temperature (decimal number)

Example

If the input temperature `T` is 50 and `unitFrom` is the string `Fahrenheit` and `unitTo` is the string `Celsius`, we need to convert 50 degrees Fahrenheit to degrees Celsius. The converted temperature can be computed as follows:

$$C = \frac{F - 32}{1.8} = \frac{50 - 32}{1.8} = 10. \tag{5.1}$$

Thus, the output temperature `T` should be set to 10.

Example test case

Remember to thoroughly test your code before handing it in. You can test your solution on the example above by running the following test code and checking that the output is as expected.

Test code	Expected output
`disp(convertTemperature(50, 'Fahrenheit', 'Celsius'))`	`10`

Discussion and futher analysis

Take some time to reflect on and discuss the way that you have commented your code. Do you follow all the suggestions given in section 5.2?

5A

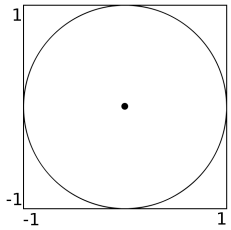
```
%It is so simple that comments are futile...
function T = convertTemperature(T, unitFrom, unitTo)
    while 1
       if strcmp(unitFrom, 'Celsius')
           if strcmp(unitTo, 'Fahrenheit')
                T = 1.8*T + 32;
               break
           elseif strcmp(unitTo, 'Kelvin')
                T = T + 273.15;
               break
           else
                disp('Error');
           end

       elseif strcmp(unitFrom, 'Fahrenheit')
           if strcmp(unitTo, 'Celsius')
                T = (T - 32)/1.8;
               break
           elseif strcmp(unitTo, 'Kelvin')
                T = (T + 459.67)/1.8;
               break
           else
                disp('Error');
           end
       elseif strcmp(unitFrom, 'Kelvin')
           if strcmp(unitTo, 'Celsius')
                T = T - 273.15;
               break
           elseif strcmp(unitTo, 'Fahrenheit')
                T = 1.8*T - 459.67;
               break;
           else
                disp('Error');
           end
       else
            disp('Error');
       end
    end
end
```

Assignment 5E Monte Carlo estimation

The Monte Carlo method is a technique that can be used to approximate the distribution of random outcomes. This is done by recording the outcomes of running the same simulation multiple times. In this way, the method resembles repeated playing the same gambling game at a casino.

The Monte Carlo method is very useful, especially to approximate solutions to problems that are analytically difficult. In this assignment, we will use the Monte Carlo method in a simple scenario, to numerically estimate the area of a shape.

Consider a circle with radius 1, centered on $(0, 0)$.



By generating N random points uniformly on the the circumscribing square and counting how many points n that fall within the circle, we can estimate the area of the circle by multiplying the area of the square by the fraction of points inside the circle

$$A_{\text{circle}} \approx A_{\text{square}} \cdot \frac{n}{N}, \tag{5.5}$$

where in this case $A_{\text{square}} = 4$. To test if a point (x, y) is inside the circle, we can simply check if it magnitude of the vector from the center of the circle to (x, y) is less than one.

Problem definition

Write a function that estimates the area of a circle by Monte Carlo simulation. As input the function must receive two vectors containing the coordinates of the randomly drawn points. The function must return the estimated value of the area as output.

To test your function, you can use the function implemented in exercise 5D to generate a vector for the random x-values and a vector for the random y-values.

Solution template

```
function A = circleAreaMC(xvals, yvals)
```

Input	
`xvals`	The x-coordinates of points (vector of decimal numbers)
`yvals`	The y-coordinates of points (vector of decimal numbers)
Output	
`A`	Estimated value for the area of the circle (scalar decimal number)

Example

If we have randomly have drawn the following $N = 5$ points

$$(-0.1, 0.3)\ ,\ (0.7, -0.1)\ ,\ (0.8, 0.9)\ ,\ (0.5, 0.6)\ ,\ (-0.4, -0.3) \tag{5.6}$$

four of the points lies within the circle, and the area would be estimated as $A \approx 4 \cdot \frac{4}{5} = 3.2$.

Example test case

Remember to thoroughly test your code before handing it in. You can test your solution on the example above by running the following test code and checking that the output is as expected.

Test code	Expected output
`disp(circleAreaMC([-0.1, 0.7, 0.8, 0.5, -0.4], [0.3, -0.1, 0.9, 0.6, -0.3]))`	`3.2000`

Hand in on CodeJudge

The assignment must be handed in on CodeJudge.

Discussion and futher analysis

- Try calling your function for different number of points (like 10, 1 000, and 1 000 000).
- Try calling your function multiple times with the same number of points.
- Try running the following code to plot an image of a circle along with your points.

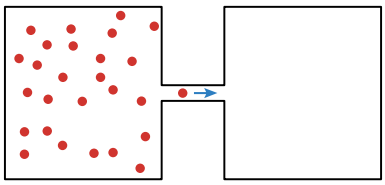
```
p = 0:0.01:2*pi;
plot(sin(p), cos(p));
axis equal; hold on;
plot(xvals, yvals, 'o');
```

5E

```
function A = circleAreaMC(xvals, yvals)
    N = length(xvals);
    if  (N ~= length(yvals))
        disp('Error: coordinates do not match');
        return
    end
    v = zeros(1,N);
    v(sqrt(xvals.^2 + yvals.^2) < 1) = 1;
    n = sum(v);
    A = 4*n/N;
end
```

Optional challenge 5F Thermodynamic simulation

Consider a physical experiment with two connected closed chambers of equal volume. The right chamber is initially empty, while the left contains N identical gas particles. We open a small hole between the chambers, such that the gas particles can move between the two chambers.



After some time, we expect that the system will reach an equilibrium, where the number of gas particles in the left chamber N_L is the same as the number of gas particles in the right chamber N_R. In this challenge we will use the Monte Carlo method to simulate the behaviour of the gas as it approaches this equilibrium.

The evolution of the system is considered as a series of time-steps, beginning at $t = 1$. At each time-step exactly one particle will pass through the hole, and we assume that the particles do not interact. The probability that a particle will move from the left to the right chamber is $p_{LR} = N_L/N$, and the probability of a particle will move from the right to the left chamber is $p_{RL} = 1 - p_{LR} = (N - N_L)/N$.

The simulation will iteratively proceed as follows:

1. Get a random number r from the interval $0 \leq r \leq 1$.
2. If $r \leq p_{LR}$, move one particle from the left to the right chamber. Otherwise move one particle from the right to the left chamber.
3. Repeat step 1 and 2 until $N_L = N_R$. Report back, how many time-steps it took to reach this equilibrium.

Problem definition

Write a function that performs the Monte Carlo simulation as described in the above algorithm. As input to the function, you have the number of particles (which must be an even number in order for an equilibrium to exist) and a vector of N random numbers between 0 and 1. The function must use the first random number in the vector as r at time $t = 1$, the second random number at time $t = 2$ etc. If the function runs out of random numbers before an equilibrium is reached, it must return the value $t = 0$ to indicate this.

Solution template

```
function t = thermoEquilibrium(N, r)
% Insert your code here
```

Input	
`N`	Initial number of gas particles in the left chamber (even integer number)
`r`	Sequence of random numbers (vector of decimal numbers between 0 and 1)
Output	
`t`	Time-steps used to reach equilibrium (integer number) Return zero if equilibrium is not reached.

Example

Consider the simple case where there are only $N = 2$ particles. At $t = 1$ a particle will move from left to right (i.e., with probability $p_{LR} = \frac{2}{2} = 1$). When one particle moves to the right chamber, there will be exactly one particle in each chamber, and equilibrium has been reached. The number of time-steps to reach equilibrium is thus 1.

Example test case

Remember to thoroughly test your code before handing it in. You can test your solution on the example above by running the following test code and checking that the output is as expected.

Test code	Expected output
`disp(thermoEquilibrium(2, [0.16, 0.04, 0.72, 0.09, 0.17, 0.60, 0.26, 0.65, 0.69, 0.74, 0.45, 0.61, 0.23, 0.37, 0.15, 0.83, 0.61, 1.00, 0.08, 0.44]))`	`1`

Hand in on CodeJudge

This challenge can be handed in on CodeJudge.

Discussion and futher analysis

Test your function for different numbers of N (remember that N must be even in order for a equilibrium state to exist.)

- Does the number of time steps to reach equilibrium depend on the number of gas particles?
- Experiment with a large numbers of particles. Will the system always reach equilibrium?

5F

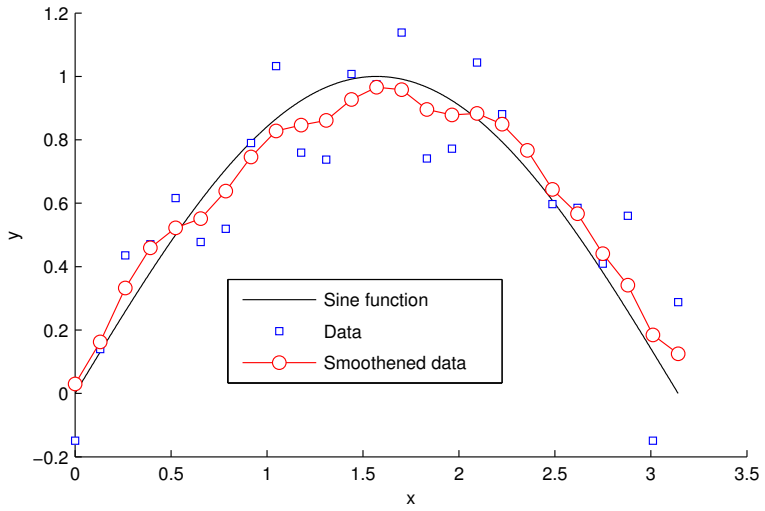
```
% 1. Get a random number r from the interval [0, 1].

% 2. If r <= pLR, move one particle from the left to the right chamber.
%    Otherwise move one particle from the right to the left chamber.

% 3. Repeat step 1 and 2 until NL = NR. Report back, how many time-steps
%    it took to reach this equilibrium.
function t = thermoEquilibrium(N, r)
    t = 0;
    NL = N;
    while ((NL ~= (N/2)) && (t < length(r)))
        p = NL / N;
        t = t + 1;
        if (r(t) <= p)
            NL = NL - 1;
        else
            NL = NL + 1;
        end
    end
    if ((NL ~= (N/2)) && (t >= length(r)))
        t = 0;
    end
end
```

Assignment 6C Moving average

When recording any type of noisy data, that be audio signals or stock prices, noise is often reduced by *smoothening* the data. One algorithm of smoothing a signal is the n-sample symmetric weighted moving average. Let us say we are given a signal as a vector y with values y_i for $i \in (1, \ldots, N)$. Then, a five-sample symmetric weighted moving average, $\hat{y}$, can be computed by

$$\hat{y}_i = \frac{y_{i-2} + 2 \cdot y_{i-1} + 3 \cdot y_i + 2 \cdot y_{i+1} + y_{i+2}}{9}, \tag{6.8}$$

i.e., as a weighted average of the surrounding datapoints. To compute the first two and last two smoothened signal values, we must make some assumption about the signal outside its bounds: We will assume that y_i is zero for $i < 1$ and $i > N$. The algorithm is illustrated below. The figure shows a noisy measurement of a sine wave function and a five-sample symmetric weighted moving average.



Problem definition

Create a function that takes a signal vector as input and computes the five-sample weighted moving average of the signal.

Solution template

```
function ysmooth = movingAvg(y)
% Insert your code here
```

Input	
`y`	Input signal (vector)
Output	
`ysmooth`	Five-sample moving average smoothing of input signal (vector)

Example

If your signal file consists of a vector

$$y = [0.8, 0.9, 0.7, 0.6, 0.3, 0.4] \tag{6.9}$$

We can solve the smoothing problem by making use of a matrix: We first construct a matrix where each row is

a shifted and scaled version of the signal as illustrated below:

$$\begin{array}{cccc} 1 & \cdot & 0.8 & 0.9 \\ 2 & \cdot & & 0.8 \\ 3 & \cdot & & \\ 2 & \cdot & & \\ 1 & \cdot & & \end{array} \begin{bmatrix} 0.7 & 0.6 & 0.3 & 0.4 & 0 & 0 \\ 0.9 & 0.7 & 0.6 & 0.3 & 0.4 & 0 \\ 0.8 & 0.9 & 0.7 & 0.6 & 0.3 & 0.4 \\ 0 & 0.8 & 0.9 & 0.7 & 0.6 & 0.3 \\ 0 & 0 & 0.8 & 0.9 & 0.7 & 0.6 \end{bmatrix} \begin{array}{cc} & \\ & \\ & \\ 0.4 & \\ 0.3 & 0.4 \end{array} = \begin{bmatrix} 0.7 & 0.6 & 0.3 & 0.4 & 0 & 0 \\ 1.8 & 1.4 & 1.2 & 0.6 & 0.8 & 0 \\ 2.4 & 2.7 & 2.1 & 1.8 & 0.9 & 1.2 \\ 0 & 1.6 & 1.8 & 1.4 & 1.2 & 0.6 \\ 0 & 0 & 0.8 & 0.9 & 0.7 & 0.6 \end{bmatrix}. \quad (6.10)$$

In the first row, y is shifted left twice; in the second row y is shifted left once and multiplied by two; in the third row y is multiplied by three; etc. Summing each column and dividing by 9 yields the final result

$$\hat{y} = [0.54444, 0.7, 0.68889, 0.56667, 0.4, 0.26667]. \quad (6.11)$$

Example test case

Remember to thoroughly test your code before handing it in. You can test your solution on the example above by running the following test code and checking that the output is as expected.

Test code	Expected output
`disp(movingAvg([0.8, 0.9, 0.7, 0.6, 0.3, 0.4]))`	`0.5444 0.7000 0.6889 0.5667 0.4000 0.2667`

Hand in on CodeJudge

In order to validate you can either upload a file containing your implementation of the function or directly copy the code into codejudge.

Discussion and futher analysis

Try loading the variables `x` and `y` from the data file `smooth.mat` which contains the noisy sine function data used to generate the figure above. You should then be able to reproduce the plot in the figure using the following code:

```
plot(x, sin(x), 'k');
hold on;
plot(x, y, 'g.');
plot(x, movingAvg(y), 'r.-');
```

```
function ysmooth = movingAvg(y)
    S = [-2,-1,0,1,2];
    if (size(y) == 1)
        ysmooth = y / 3;
        return
    else
        for i = 1:3
            if (i ~= 3)
                M(i,:) = i*circshift(y,[0, S(i)]);
                M(5-i+1,:) = (i)*circshift(y, [0,S(6-i)]);
            else
                M(i,:) = i*y;
            end
        end

        [n,m] = size(M);
        M(1,(m-1):m) = 0;
        M(2,m) = 0;
        M(4,1) = 0;
        M(5,1:2) = 0;

        ysmooth = sum(M, 1)./9;
    end
end
```

Assignment 6D Frequency of letters

A simple way to analyze a text document is to examine how often the different letters in the alphabet occur in the text. As you will see in a later assignment, this can be used for example to detect the language in which the text is written. In this assignment we will create a function that takes the filename of a text file as input and computes the frequency of the letters in the text. In order to simplify the problem we will convert the text to lower case and only consider the characters from `a` to `z`.

First, we must read in the text file and convert it to lower case. Then we should count the number of times each of the letters `a` to `z` occur. Here it might be a good idea to create a string or vector with all the letters in the alphabet: Using that, we can make a loop over the letters in order to count how many times they each occur in the text. Finally, based on the occurence counts we can compute the frequency.

Problem definition

Create a function that takes a text file as input and returns a vector of size 26 with the frequency in percent of each character `a`, `b`, ... `z` (not sensitive to case.) The frequency must be computed as the number of occurences divided by the total number of characters from `a` to `z` that occur, multiplied by one hundred. All other letters such as `ø` and `ä` as well as all symbols must be ignored.

Solution template

```
function freq = letterFrequency(filename)
% Insert your code here
```

Input

`filename` A string that is the filename of a plain text file in the current working directory.

Output

`freq` A vector of length 26 containing the frequency of occurrence of each of the 26 letters from a to z.

Example

In the file `small_text.txt` is a small example. Counting the number of occurences of each of the letters yields the result shown below (second row). Dividing each number by the total number of occurences and multiplying with 100 yields the frequency (last row).

Letter	a	b	c	d	e	f	g	h	i	j	k	l	m
Occurrences	105	29	32	59	160	26	25	87	93	1	15	44	14
Frequency (%)	8.10	2.24	2.47	4.55	12.35	2.01	1.93	6.71	7.18	0.08	1.16	3.40	1.08

Letter	n	o	p	q	r	s	t	u	v	w	x	y	z
Occurrences	87	102	19	1	78	70	142	37	12	38	0	20	0
Frequency (%)	6.71	7.87	1.47	0.08	6.02	5.40	10.96	2.85	0.93	2.93	0.00	1.54	0.00

Example test case

Remember to thoroughly test your code before handing it in. You can test your solution on the example above by running the following test code and checking that the output is as expected.

Test code

```
disp(letterFrequency('small_text.txt'))
```

Expected output

```
Columns 1 through 7
    8.1019    2.2377    2.4691    4.5525   12.3457    2.0062    1.9290
Columns 8 through 14
    6.7130    7.1759    0.0772    1.1574    3.3951    1.0802    6.7130
Columns 15 through 21
    7.8704    1.4660    0.0772    6.0185    5.4012   10.9568    2.8549
Columns 22 through 26
    0.9259    2.9321         0    1.5432         0
```

Hand in on CodeJudge

This exercise must be handed on CodeJudge.

6D

```
function freq = letterFrequency(filename)
    t = fileread(filename);
    t = lower(t);
    N = length(t);
    freq = zeros(1,26);
    v = 0;
    for i=1:N
        c = int8(t(i));
        if (c >= 'a' && c <= 'z')
            freq(c - ('a'-1)) = freq(c - ('a'-1)) + 1;
            v = v + 1;
        else
            continue
        end
    end
    freq = (freq*100)./ v;
end
```

Assignment 6E Language detection

Different languages use the different letters with different frequencies and this can be used to determine in which language a text is written. In this exercise you should use the function created in the previous exercise to compute the frequencies of letters in a text. Given this vector of frequencies you can compute the *squared error* between the frequencies in the text and the (average) frequencies in the language.

$$E_\ell = \sum_{i=1}^{26} (F_i^t - F_i^\ell)^2 \tag{6.12}$$

where F_i^t is the frequency of letter i in the text and F_i^ℓ is the frequency of letter i in the language. The language which has the lowest squared error is the one that best matches the text in terms of the letter frequency.

The frequencies of the letters in fifteen different languages are given in the file `letter_frequencies.csv`.[1] A snapshot of a part the file is given below.

Letter	English	French	German	Spanish	Portuguese	Esperanto	Italian	Turkish
a	8.167	7.636	6.516	11.525	14.634	12.117	11.745	12.92
b	1.492	0.901	1.886	2.215	1.043	0.98	0.927	2.844
c	2.782	3.26	2.732	4.019	3.882	0.776	4.501	1.463
d	4.253	3.669	5.076	5.51	4.992	3.044	3.736	5.206
e	12.702	14.715	16.396	12.681	11.57	8.995	11.792	9.912
f	2.228	1.066	1.656	0.692	1.023	1.037	1.153	0.461
g	2.015	0.866	3.009	1.768	1.303	1.171	1.644	1.253
h	6.094	0.737	4.577	0.703	0.781	0.384	0.636	1.212
i	6.966	7.529	6.55	6.247	6.186	10.012	10.143	9.6
j	0.153	0.613	0.268	0.443	0.397	3.501	0.011	0.034

Problem definition

Create a function that takes as input a vector of frequencies of occurrences of letters in a text. The function must read the file `letter_frequencies.csv`, compute the squared error for each language, and return a vector of squared errors for the fifteen languages.

Solution template

```
function E = computeLanguageError(freq)
% Insert your code here
```

Input

`freq` A vector of size 26 containing the frequency of the letters `a`–`z` in a text.

Output

`se` A vector of length 15 containing the squared error between the input vector and each of the 15 languages in the CSV file

Example

Let the vector in the following table be the input vector and the CSV file as the `letter_frequencies.csv` from CampusNet. This should give the following squared error for the first 10 languages:

English	French	German	Spanish	Portuguese	Esperanto	Italian	Turkish	Swedish	Polish
9.04	108.24	99.55	121.02	165.54	164.75	128.56	211.07	89.98	190.64

Example test case

Remember to thoroughly test your code before handing it in. You can test your solution on the example above by running the following test code and checking that the output is as expected.

[1] Source: http://en.wikipedia.org/wiki/Letter_frequency

Test code	Expected output

`disp(computeLanguageError([8.101852, 2.237654, 2.469136, 4.552469, 12.345679, 2.006173, 1.929012, 6.712963, 7.175926, 0.077160, 1.157407, 3.395062, 1.080247, 6.712963, 7.870370, 1.466049, 0.077160, 6.018519, 5.401235, 10.956790, 2.854938, 0.925926, 2.932099, 0.000000, 1.543210, 0.000000]))`	`Columns 1 through 7`
	`9.0393 108.2363 99.5453 121.0195 165.5445 164.7483`
	`128.5608`
	`Columns 8 through 14`
	`211.0725 89.9806 190.6440 93.7989 112.9329 192.2470`
	`173.1080`
	`Column 15`
	`134.5387`

Hand in on CodeJudge

This exercise must be handed on CodeJudge.

Discussion and futher analysis

This method of only using the letter frequencies to determine a language is not very efficient, specially when the text of which you are trying to identify the language of is only a few words.

The method can be extended to use bigrams or trigrams giving a quite robust method of identifying the language of a text that is actually used in practice. Bigrams are two adjacent characters in a word instead of single characters. The two most frequent bigrams in english are ‘th’ and ‘he’.

6E

```
function E = computeLanguageError(freq)
    tab = readtable('letter_frequencies.csv');
    M = tab{1:end, 2:end};
    C = M - diag(freq)*ones(size(M));
    C = C.^2;
    E = sum(C,1);
end
```

Assignment 7G The NATO alphabet

The NATO alphabet is useful for spelling words over a noisy telephone connection.

A Alpha	B Bravo	C Charlie	D Delta	E Echo	F Foxtrot	G Golf	H Hotel	I India
J Juliet	K Kilo	L Lima	M Mike	N November	O Oscar	P Papa	Q Quebec	R Romeo
S Sierra	T Tango	U Uniform	V Victor	W Whiskey	X Xray	Y Yankee	Z Zulu	

Problem definition

Create a function that spells out a word (written in plain text) in the NATO alphabet. The function should accept input in upper, lower, or mixed case, and should output the NATO "code words" exactly as written above, separated by dashes. You can assume that the input string contains only letters from a-z (A-Z).

Solution template

```
function natoText = textToNato(plainText)
% Insert your code here
```

Input `plainText`	Input word in plain text (string)
Output `natoText`	Output in dash-separated NATO alphabet (string)

Example

For example, given the input `hello`, the function should return the string `Hotel-Echo-Lima-Lima-Oscar`.

Example test case

Remember to thoroughly test your code before handing it in. You can test your solution on the example above by running the following test code and checking that the output is as expected.

Test code	Expected output
`disp(textToNato('hello'))`	`Hotel-Echo-Lima-Lima-Oscar`

Hand in on CodeJudge

In order to validate you can either upload a file containing your implementation of the function or directly copy the code into codejudge.

Discussion and futher analysis

How could you create a function that does the reverse, i.e., converts a word written in the NATO alphabet (as output by your function) back into plain text? If you are up for the task, create such a function and test thoroughly that it works correctly by converting words first to the NATO alphabet and then back and checking that the result is identical to the original input.

7G

```
function natoText = textToNato(plainText)
    t = lower(plainText);

    L = length(t);
    s = [];

    for i = 1:L
       switch t(i)
           case 'a'
                s = strcat(s,'Alpha-');
               continue
           case 'b'
                s = strcat(s,'Bravo-');
               continue
           case 'c'
                s = strcat(s,'Charlie-');
               continue
           case 'd'
                s = strcat(s,'Delta-');
               continue
           case 'e'
                s = strcat(s,'Echo-');
               continue
           case 'f'
                s = strcat(s,'Foxtrot-');
               continue
           case 'g'
                s = strcat(s,'Golf-');
               continue
           case 'h'
                s = strcat(s,'Hotel-');
               continue
           case 'i'
                s = strcat(s,'India-');
               continue
           case 'j'
                s = strcat(s,'Juliet-');
               continue
           case 'k'
                s = strcat(s,'Kilo-');
               continue
           case 'l'
                s = strcat(s,'Lima-');
               continue
           case 'm'
                s = strcat(s,'Mike-');
               continue
           case 'n'
                s = strcat(s,'November-');
               continue
           case 'o'
                s = strcat(s,'Oscar-');
               continue
           case 'p'
                s = strcat(s,'Papa-');
```

```
            continue
        case 'q'
            s = strcat(s,'Quebec-');
            continue
        case 'r'
            s = strcat(s,'Romeo-');
            continue
        case 's'
            s = strcat(s,'Sierra-');
            continue
        case 't'
            s = strcat(s,'Tango-');
            continue
        case 'u'
            s = strcat(s,'Uniform-');
            continue
        case 'v'
            s = strcat(s,'Victor-');
            continue
        case 'w'
            s = strcat(s,'Whiskey-');
            continue
        case 'x'
            s = strcat(s,'Xray-');
            continue
        case 'y'
            s = strcat(s,'Yankee-');
            continue
        case 'z'
            s = strcat(s,'Zulu-');
            continue
        otherwise
            continue
    end
  end
  natoText = s(1:length(s)-1);
end
```