

Computer Assignment 1

1. The Maclaurin series of the function e^x is given by

$$e^x = \sum_{k=0}^{\infty} \frac{1}{k!} x^k.$$

How many terms of the above series expansion you need to compute the number e to 5 digits of accuracy?

2. The `uint8` class corresponds to integers with values ranging from 0 to 255. They are used for storing and manipulating images in a computer. Using the function `floor`, write a code that transforms `rand(1)` to a proportionally equivalent `uint8` variable.
3. Type `3 < 7 > 5` in the Command Window. Explain in detail why you get this result. Add a logical AND to the above statement so that the output is a logical 1 (true), given that 7 is indeed bigger than both 3 and 5.
4. A logical expression which is always true is called a *tautology*. Using **MATLAB**, show that the logical expression $P \text{ OR } ((\text{NOT } P) \text{ OR } Q)$ is a tautology.
5. Create, in one line of code (without `for` or `while` loops), a matrix of size 100×100 whose elements are 1 above the diagonal, -1 below the diagonal, and 0 along the diagonal. Consider using **MATLAB**'s built-in function `tril` or `triu`.
6. Write a function with header `M = myCheckerBoard(n)`, where `M` is an $4n \times 4n$ matrix with the following form:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Note that the upper-left element should always be 1 and the upper-right element should always be 0. You can visualize your result by running the command `imshow(M)`.

Hint: you might find the function `repmat` useful here.

7. The equation of an ellipse oriented along the x - y axis with semi-axis of size a and b , respectively, is given by

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 = 1.$$

In this exercise you will create a function that outputs an N -points discretisation of an ellipse of semi-axis lengths a and b , that is rotated at an angle `theta` with respect to the x axis.

Write a function with header `E = ellipse(a, b, theta, N)` following the steps here below.

- (i) Using `linspace`, create an array of size $1 \times N$ consisting of evenly-spaced doubles between 0 and 2π .
- (ii) Create an array of size $2 \times N$ where the first/second row is the cosine/sine of the values of the array you created in (i). This will give you an N -points discretisation of a circle
- (iii) The result of (ii) is an N -points discretisation of a circle with radius 1, where the x coordinate is the first row and the y coordinate is the second row. Scale each coordinate by **a** and **b**, respectively, to obtain the discrete version of an ellipse with semi-axis lengths **a** and **b**.
- (iv) The matrix

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

is a rotation matrix – it maps any given vectore $u \in \mathbb{R}^2$ to another vector $v \in \mathbb{R}^2$ such that $\|u\| = \|v\|$ and the angle between the u and v (anticlockwise) is θ . Using this, create an array of size $2 \times N$ whose columns are θ rotations of the columns of the discretised ellipse of (iii).

You can visualise your results using the command `plot(E(1,:), E(2,:)), axis equal`.

8. Repeat question 7, but now using complex numbers. In particular, instead of using arrays of size $2 \times N$ to store the discretised circle and ellipses, you will use arrays of size $1 \times N$, whose elements are complex numbers to convey the same information. Remember that a complex number multiplied by $e^{i\theta}$ rotates the number an angle θ anti-clockwise. You can visualise your results by running the command `plot(E), axis equal`, where E is the output to your function – your function's output should be an array of complex numbers of size $1 \times N$.
9. Write a program that finds all the prime numbers that are smaller than 100.
10. Download the image `Maxwell_color.png` (see Fig. 1) from Blackboard and save it in your file directory. Run the command `Maxwell = imread(['Maxwell_color.png'])` to read the image in MATLAB. The variable `Maxwell` should be a multidimensional array of class `uint8`, of size $n \times m \times 3$ (n and m are the dimensions of the image), where the first two index dimensions indicate the indices of each pixel, and the third index indicate one of each the red, green, and blue (RGB) components of each color pixel.
 - (i) From the 3D array `Maxwell`, define three 2D arrays, one for each colour, and transform the types of these arrays from `uint8` to `double`.
 - (ii) (*scale of greys*) Add the three matrices from (i) in a pondered average with a weight of 0.1 for red, 0.8 for green, and 0.1 for blue. You can visualise your result by running `imshow(uint8(I))`, where I is your `double` matrix resulting from the pondered average.
 - (iii) (*blurring the image*). Given an integer **b** such that $0 < b < m, n$, replace every element of I by the average its neighbouring pixels, in a neighbourhood of (at



Figure 1: The man on this picture is James Clerk Maxwell, a Scottish Mathematical-Physicist best known for formulating the mathematical theory of electromagnetic waves. James C. Maxwell also did extensive investigations on the nature of color perception, including the RGB description of colors.

most) `b` pixels in both directions. Transform the resulting matrix from `double` to `uint8` and visualise using the function `imshow`.

- (iv) (*bonus question*) Write a function that take as an input a colour image and a blurring parameter `b` (the same from (iii)), and outputs a coloured blurred version the image.