EXERCISE 1: INTRODUCTION

Computational Physics

Author
George Farah
Email: george.farah@rwth-aachen.de
Matricle No: 421409

Contents

1 Introduction

In this first exercise sheet, two exercises are presented. The first exercise is a basic exercise creating a 6×6 matrix with random numbers between -5 and 5, then the largest elements and their positions are found. Afterward, a row and a column consisting of the largest value at each column and row are created, and then these two are multiplied. At the end another matrix with the same characteristics as the first matrix is created and both are multiplied.

In the second exercise, the Chebyshev polynomial is discussed. First, a program is written that gives the polynomials of degree N and smaller for input x between -1 and 1. Afterward, the polynomials are plotted for degree 4 and smaller.

Important side note: As an exception this exercise is submitted by one person only, as I am still looking for a partner, hopefully by the next exercise I will already have one, as I am a data science student and I don't know many people from the physics department. But if I don't find anyone I would still like to proceed with this course on my own.

2 Simulation Model and Method

For both exercises programming language Python has been used with its libraries numpy and matplotlib for plots.

2.1 Exercise One

For exercise 1, random values from uniform distribution have been used, using the numpy function: numpy.random.uniform

For the second and third, and fourth parts the numpy function numpy.max has been used to find the maximum values and numpy where to find the index, and numpy.dot to multiply the vectors and matrices.

2.2 Exrcise Two

The Chebyshev polynomial for polynomial of degree $n > 1is$ defined as the following:

$$T_n(x) = \begin{cases} 1, & \text{if } n = 0 \\ x, & \text{if } n = 1 \\ 2xT_{n-1}(x) - T_{n-2}(x), & \text{otherwise} \end{cases} \tag{1}$$

A recursive function according to this definition is defined in Python to find the polynomial of degree n and for x between -1 and 1. On top of this function, another function is defined to find all polynomials of degree N and smaller, for input x of any length. It returns a matrix of size length$(x) \times N + 1$.

3 Simulation results

Exercise 1.1

A random matrix of size 6×6 for values between -5 and 5 with the random seed of 1409:

$$A = \begin{bmatrix} -4.39541752 & -4.35184249 & 0.04270395 & -1.37646311 & 1.92959141 & -3.43260207 \\ 2.92636905 & -2.90388447 & -3.60797792 & 0.78220062 & 4.48872299 & -1.1624837 \\ -3.63926417 & 2.35372732 & -4.60114811 & -3.03037669 & 0.39013925 & 3.05750356 \\ 1.41951188 & -3.58183405 & -1.55717345 & 3.72633397 & 2.35235009 & 0.24529594 \\ -2.26545385 & 1.0382752 & 3.93943639 & -4.49688802 & 1.71553135 & 0.88228205 \\ 1.08651554 & 4.18963106 & -4.05065504 & 4.1564247 & 2.41653967 & 2.16088762 \end{bmatrix}$$

Exercise 1.2

Index of the largest value in the matrix A is: (1, 4), and its value is: 4.48872 With 0 as the first index

Exercise 1.3

The column of the largest values in each row:

$$\begin{pmatrix} 1.92959141 \\ 4.48872299 \\ 3.05750356 \\ 3.72633397 \\ 3.93943639 \\ 4.18963106 \end{pmatrix}$$

The row of the largest values in each column:

$$[2.92636905 \quad 4.18963106 \quad 3.93943639 \quad 4.1564247 \quad 4.48872299 \quad 3.05750356]$$

We could either multiply the column with the row and have a dot product, in this case the output is: 82.4787

Or we could multiply the row with the column and have outer product with woudl result in a 6×6 Matrix:

$$\begin{pmatrix} 5.64669658 & 13.13566005 & 8.94738378 & 10.9046284 & 11.52824473 & 12.26040668 \\ 8.08427609 & 18.80609327 & 12.80981186 & 15.61196453 & 16.50478505 & 17.55300843 \\ 7.6015026 & 17.68303869 & 12.04484076 & 14.67965562 & 15.51915905 & 16.50478505 \\ 8.02020139 & 18.65703912 & 12.7082833 & 15.48822654 & 16.37397071 & 17.41388603 \\ 8.66140132 & 20.1486341 & 13.72428651 & 16.72648096 & 17.68303869 & 18.80609327 \\ 5.89973259 & 13.72428651 & 9.34832799 & 11.39327935 & 12.04484076 & 12.80981186 \end{pmatrix}$$

Exercise 1.4

Random matrix B of size 6×6 and values between -5 and 5:

$$B = \begin{bmatrix} -3.63135564 & -3.69084833 & 2.06599055 & 3.34215449 & -0.51629922 & -4.5583217 \\ -2.27964906 & 2.37179272 & -1.92599113 & 3.27748086 & -0.46016084 & -3.66277583 \\ -0.36708228 & 0.68782433 & 4.14412495 & 1.62451805 & 0.78488277 & -0.51051754 \\ -4.42291923 & 4.58757551 & 0.01718295 & 3.4272739 & -0.99105507 & -0.21524978 \\ -2.25036106 & 2.93456462 & -2.1102394 & 1.93448223 & 1.85088142 & -2.77331833 \\ 1.43654297 & -3.39197136 & 1.64869643 & 4.22281582 & -2.86020059 & -0.81334938 \end{bmatrix}$$

$$C = AB = \begin{bmatrix} 22.6809494 & 16.921694 & -10.2771808 & -44.3638736 & 19.0589418 & 33.6905664 \\ -17.9132399 & 0.534112138 & -14.6886271 & 0.856942526 & 7.85135882 & -12.5326386 \\ 26.4561825 & -7.27841382 & -26.9541388 & -8.64327415 & -7.83520754 & 7.40021709 \\ -17.8403501 & 8.36029226 & -1.1174165 & 8.83276523 & -0.347548381 & -0.0815917322 \\ 21.7099119 & -5.05449084 & 7.40256514 & -6.13657155 & 8.89227932 & 0.00518280207 \\ -32.7268988 & 21.9703684 & -24.0762919 & 38.8273655 & -11.49525 & -27.5844962 \end{bmatrix}$$

$$D = BA = \begin{bmatrix} -1.2221843 & -22.5253823 & -24.11392544 & -12.27954737 & -4.18946932 & -14.99142074 \\ -4.86617688 & -15.69325287 & 57.12109271 & -6.92608464 & -17.18816538 & -10.66293564 \\ 21.3777584 & -2.79432319 & 17.95571637 & 18.27715066 & -41.17695082 & 4.8542682 \\ 9.06608498 & -1.21176366 & -4.20655398 & 32.49755561 & -16.49310139 & -7.9295951 \\ 2.86842127 & -21.17582061 & -43.37868116 & 29.07277185 & 19.83696111 & -13.12418508 \\ -16.5921555 & 6.10227311 & -3.32605279 & 0.25212858 & -4.19303635 & 1.23647005 \end{bmatrix}$$

Exercise 2.1

This exercise has been explained in (2.2), and the code is included in (5).

The cheby function takes as an input a vector x of values between -1 and 1 of ay length (we have chosen 1000 here for next exercise) and polynomial degrees N. It returns a matrix of size length$(x) \times N + 1$, with all the polynomials from degree N to 0.

Exercise 2.2

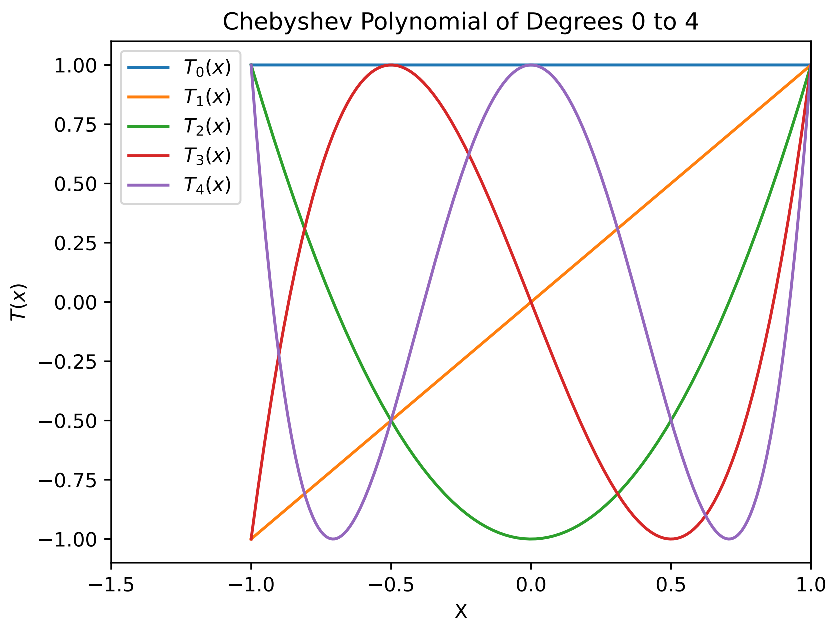

Figure 1: Comparing the e Chebyshev polynomial for degrees from 0 to 4.

4 First Appendix

Here is the code for the first exercise complete in python:

```python
import numpy as np
"""
exercise 1:

"""
## 1:
matrikel="421409"
np.random.seed(1409)
A = np.random.uniform(low=-5.0, high=5.0, size=(6,6))
print(A)

## 2:
max_val = np.max(A)
max_idx = np.where(A == max_val)
print(f"Index of the largest value in the matrix A is: {max_idx[0][0],max_idx[1][0]}, and its value is:{max_val}")

# 3:
larget_in_col_row = np.max(A,axis=0)
larget_in_row_col = np.max(A,axis=1)
print(larget_in_col_row)
print(larget_in_row_col)

print(f"Multiplying the column and the row as dot product is: {np.dot(larget_in_row_col,larget_in_col_row)}")
print(f"Multiplying the row and the column as outer product is: {np.outer(larget_in_col_row,larget_in_row_col)}")

# 4:
B = np.random.uniform(low=-5.0, high=5.0, size=(6,6))
C = np.dot(A,B)
D = np.dot(B,A)
```

5 Second Appendix

Here is the code for the second exercise complete in python:

```
import numpy as np
import matplotlib.pyplot as plt
"""
exercise 2:
"""
# 1:
def cheby(x,N):
    ## define the recursive chebyshev function
    def chebyshev_poly(x, n):
        if n == 0:
            return 1
        elif n == 1:
            return x
        else:
            return 2 * x * chebyshev_poly(x, n - 1) - chebyshev_poly(x, n - 2)
    ## define a matrix
    Matrix = np.zeros((len(x),(N+1)))
    ## fill the matrix with chebyshev polynomials for all x values
    ## for all degrees of N and smaller to zero
    for ns in range(N+1):
        for i, xs in enumerate(x):
            Matrix[i,ns] = chebyshev_poly(xs, ns)
    return Matrix

## create x of length 1000
X = np.linspace(-1,1,1000)
## define the orders of polynomials
N = 5
cheby_matrix = cheby(X,N)
###########################
## 2:
T0 = cheby_matrix[:,0]
T1 = cheby_matrix[:,1]
T2 = cheby_matrix[:,2]
T3 = cheby_matrix[:,3]
T4 = cheby_matrix[:,4]

plt.plot(X,T0, label="$T_0(x)$")
plt.ylabel("$T(x)$")
plt.xlabel('X')
```

```

plt.plot(X,T1,label="$T_1(x)$")
plt.plot(X,T2,  label="$T_2(x)$")
plt.plot(X,T3,  label="$T_3(x)$")
plt.plot(X,T4,label="$T_4(x)$")
plt.xlim(-1.5,1)
plt.legend()
plt.savefig("polynomials.pdf", bbox_inches="tight")
plt.show()
```