Assignment 3

Scientific Programming: Operations on Matrices

Pedram Pasandide

Due Date: 25 March

1 Introduction

In this programming assignment, you will be tasked with creating a C program that generates two random 2D arrays, named A and B, with dimensions N1 by M1 and N2 by M2, respectively. These arrays will consist of double-precision **random numbers from -10 to 10**. Your program will then implement functions to perform various scientific operations on these matrices, including addition ($A + B$), subtraction ($A - B$), multiplication ($A \times B$), and solving a linear system of equations ($Ax = B$ for x). The format of the input to your program will be as follows:

`Usage: ./math_matrix nrows_A ncols_A nrows_B ncols_B <operation> [print]`

where `nrows_A`(N1), `ncols_A`(M1), `nrows_B`(N2), and `ncols_B`(M2) represent the dimensions of matrices A and B, and `<operation>` denotes the operation to perform (`add`, `subtract`, `multiply`, or `solve`). Additionally, an optional `print` argument can be provided to print the matrices A, B, and the result.

The files you need to write your program:

- `math_matrix.c`,
- `functions.h`, and
- `functions.c`.

\- `math_matrix.c` is the source code where you can write your codes to call the functions:

```c
// CODE: include necessary library(s)

int main(int argc, char *argv[]) {
 // srand(time(NULL));

 // Check if the number of arguments is correct
 if (argc < 6 || argc > 7) {
  printf("Usage: %s nrows_A ncols_A nrows_B ncols_B <operation> [print]\n", argv
     [0]);
```

```c
    return 1;
  }

  // CODE: Generate random matrices A with size N1*M1 and B with size N2*M2

  // CODE: call the functions as needed
  return 0;
}
```

- `functions.h` is where I declared the functions:

```c
#ifndef FUNCTIONS_H
#define FUNCTIONS_H
// DO NOT change anything here. You will not submit this file.
// Function prototypes
void generateMatrix(int rows, int cols, double matrix[rows][cols]);
void printMatrix(int rows, int cols, double matrix[rows][cols]);
void addMatrices(int N1, int M1, double A[N1][M1], int N2, int M2, double B[N2][
    M2], double result[N1][M1]);
void subtractMatrices(int N1, int M1, double A[N1][M1], int N2, int M2, double B[
    N2][M2], double result[N1][M1]);
void multiplyMatrices(int N1, int M1, double A[N1][M1], int N2, int M2, double B[
    N2][M2], double result[N1][M2]);
void solveLinearSystem(int N1, int M1, double A[N1][M1], int N2, int M2, double B
    [N2][M2], double x[N1][M2]);

#endif /* FUNCTIONS_H */
```

Do NOT make any changes in `functions.h` file as you will not submit this file.

- The actual implementation of functions are in `functions.c`. Here you have to write your codes:

```c
// CODE: include necessary library(s)

// Function to generate a random matrix
void generateMatrix(int rows, int cols, double matrix[rows][cols]) {
  // CODE: Generate random numbers between -10 and 10
}

// Function to print a matrix
void printMatrix(int rows, int cols, double matrix[rows][cols]) {
  // CODE: to print the matrix
}
// Function to add two matrices
void addMatrices(<CODE:inputs to the function>) {
  // CODE: check for the condition
}
```

```
// CODE: do the same for subtractMatrices, multiplyMatrices, and
    solveLinearSystem functions
```

Warning! Before we move forward, **please**

- do **NOT** make any changes in `functions.h`, and
- Dynamic Memory Allocation in this assignments is **NOT** allowed.

2 Addition and Subtraction (2 points)

Write the function implementations for addition $(A+B)$, named `addMatrices()`, and subtraction $(A-B)$, named `subtractMatrices()`. The functions should handle matrices of different dimensions appropriately and print an error message if the dimensions are incompatible. For example, using

`./math_matrix 2 2 2 2 add print`,

in the terminal, I have:

```
Matrix A:
-6.183308 -2.436837
 1.713703  9.623225
Matrix B:
-4.080265 -6.198075
-1.492985  5.481758
Result of A + B:
-10.263573 -8.63491
```

If I remove `print` and just use `./math_matrix 2 2 2 2 add`, then nothing will be printed in the terminal. To subtract two matrices I can use `./math_matrix 2 2 2 2 subtract print`, which gives me:

```
Matrix A:
 7.248748 9.631234
-4.148199 3.226711
Matrix B:
-0.603738  7.175811
```

```
-2.427283 -3.926641
Result of A - B:
 7.852486 2.455424
-1.720916 7.153352
```

3 Multiplication (2 points)

Implement the function to multiply matrices A and B ($A \times B$), named `multiplyMatrices()`. Ensure that the function properly handles matrices with compatible dimensions for multiplication and prints an error message for incompatible dimensions. For instance, using

`./math_matrix 4 4 4 4 multiply print`

gives me:

```
Matrix A:
-1.922321  4.222271 -6.839894 -0.949466
-1.864533  4.569345 -1.728854 -2.339224
 9.712834 -7.586918  4.562223 -1.968853
-7.251313  9.226655  6.535630 -4.333119
Matrix B:
 3.497574  1.028684  9.208544  9.200133
-0.081537 -1.952196  7.492141 -1.109341
 4.644813  2.930490 -6.781554  3.704553
-7.805499  0.217065  3.956323  0.272180
Result of A * B:
-31.426705 -30.470500  56.560790 -47.966722
 3.334694  -16.412427  19.534143 -29.264232
 71.148525  37.744769 -6.129593   114.140961
 38.064619 -7.259518  -59.111577 -53.916357
```

4 Solving the Linear System of equations $Ax = B$ (4 points)

Write the code to solve the linear system of equations $Ax = B$ for x. The function is named `solveLinearSystem()`. Implement Gaussian elimination with back substitution to solve the system (or any other method works for you). Ensure that the function correctly handles square matrices A and properly prints the solution vector x. Using

`./math_matrix 3 3 3 1 solve print`,

I get the following results in the terminal:

```
Matrix A:
 0.713271  4.701861  0.669246
 8.058196 -3.968788 -5.423499
 5.505220  4.388402  7.253037
Matrix B:
 8.654515
 2.801487
 6.213846
Result of x=B/A:
 0.715995
 1.846469
-0.803928
```

Before submitting your codes, double check the results to make sure they are correct. You can use MATLAB syntax `A\B`. If you don't have access to MATLAB, you can use Python, like Google Colab), and solve $Ax = B$ with toolboxes we have on Python to solve the linear system of equations. Compare the results with your code, and make sure you code is faults free! You can use the following **Python** code (file `Assignment3.ipynb`), and try with different inputs for A and B:

```python
import numpy as np

A = np.array([[0.713271, 4.701861, 0.669246],
[8.058196, -3.968788, -5.423499],
[5.505220, 4.388402, 7.253037]])

B = np.array([8.654515, 2.801487, 6.213846 ])

x = np.linalg.solve(A, B)

print("Solution vector x:")
print(x)
```

This code will give me the following result, which is the same as the result we got from C code.

```
Solution vector x:
[ 0.71599548  1.84646918 -0.80392747]
```

5 Segmentation fault (4 points)

When I increase the dimension of matrix A and B to 591, like:

`./math_matrix 591 591 591 591 add`,

I get the following message in the terminal:

```
Segmentation fault (core dumped)
```

The same thing will happen for other functions as well. But let's **focus only** on `add` function. In my OS the dimension `591` is creating an issue. In your OS it **might** be different. You can try much larger numbers like 10000. What matters is that at one point the program will crash, most likely with a Segmentation fault.

Use `gdb` or `lldb` to find the exact line of code where this error happens. Mention the code line in your report, and explain **why** we get this error? The answer must be clear with numbers, prints or anything necessary to support your claim.

6 Report and Makefile (3 points)

Your report must be in a LaTeX format (`report.tex`). In your report, describe the algorithm used to solve $Ax = B$ for x, and **include all your codes inside the report**. You can create a section called appendix, then copy and past all your codes there. Produce the PDF file (`report.pdf`) to make sure it is working, and submit both of them. Create a `Makefile` and make sure it will work in any OS.

7 Submission On Avenue to Learn

You can use any resources to write the code. Don't forget to mention the source, and please follow the submission guidelines. No zip files on Avenue. Please avoid copying from each other. If the copied percentage exceeds the class average, you may be required to present your code.

- `math_matrix.c` AND `functions.c` AND `Makefile`
- `report.tex` AND `report.pdf`