REPORT OF PROJECT 2

The following document contains:

• Comments and explanations that I think are necessary for understanding my program.

• The output of my program according to the tasks.

• Program listing.

Alejandra Velasco Sánchez-Villares

Explanations

EXCERCISE 1

//STEP 0 : I prepare all I need.

//STEP 0.1 : I include the files I will need to execute my program.

#include <iostream>

#include <cmath>

//STEP 0.2 : I declare the features of the C++ Standard Library.

using namespace std;

//STEP 1 : I define the "myexp" function that approximates the exponential function for numbers.

double myexp(double x, double tol = 1e-10)

{

double s = 0;

double t = 1;

int i = 1;

//STEP 1.1 : I want to calculate the expression

// 1 + x/1! + (x^2)/2! + (x^3)/3! + (x^4)/4! + (x^5)/5! + ... till (x^n)/n is less than the tolerance.

while (t > tol)

{

//STEP 1.1.1 : I add the terms to the sum one by one.

s += t;

//STEP 1.1.2 : I multiply each time to obtain each term of the sum.

t \*= x/i;

//STEP 1.1.3 : The index increases by one to repeat the process.

i++;

}

return s;

}

//STEP 2 : I implement the compulsory main function.

int main()

{

//STEP 2.1 : I ask for the input.

cout << " Enter a number to evaluate the function " << endl;

//STEP 2.2 : I introduce the type of the input initialising a variable.

double x;

//STEP 2.3 : I tell the program the initialised variable is the input.

cin >> x;

//STEP 2.4 : I call the function.

double myExp;

myExp = myexp(x);

//STEP 2.5 : I give the solution of my exponential function.

cout << " My exponential function is " << myExp << endl;

double Exp;

Exp = exp(x);

//STEP 2.6 : I give the solution of the C++ exponential function.

cout << " The exponential function is " << Exp << endl;

double Error;

//STEP 2.7 : I give the difference between the C++ exponential function and mine.

Error = abs(myExp-Exp);

cout << " The error is " << Error << endl;

return 0;

};

EXCERCISE 2

Header File

//STEP 0 : I include the files I will need to execute my program.

# include <iomanip>

# include <iostream>

# include <cstdlib>

#include <cmath>

#include "r8mat\_expm1.h"

#ifndef PROJECT2\_H

#define PROJECT2\_H

using namespace std;

//STEP 1: I implement the Matrix class for square matrixes.

class Matrix {

private:

double\*\* matrix;

int m;

public:

//STEP 2: I create the constructor with parameters which allocates memory for an mxm matrix.

Matrix(int n) {

if(n >= 0)

m = n;

matrix = (double \*\*) calloc(m, sizeof(double \*\*));

for (int i = 0; i < m; i++) { matrix[i] = (double \*) calloc(m, sizeof(double)); }

}

//STEP 3: I create the copy constructor.

Matrix(const Matrix& A) {

for(int i=0; i<m; i++) {

for(int j=0; j<m; j++) {

matrix[i][j] = A.matrix[i][j];

}

}

}

//STEP 4: I create the destructor.

~Matrix() { };

/// Assignment operator (=).

Matrix& operator=(const Matrix& M) {

if(this != &M) { matrix = M.matrix; }

return \*this;

}

//STEP 5: I create the operation between matrixes for addition.

Matrix& operator+=(const Matrix& M) {

for (int i = 0; i < m; i++) {

for (int j = 0; j < m; j++) {

matrix[i][j] += M.matrix[i][j];

};

};

return \*this;

};

//STEP 6: I create the operator between two matrixes for multiplication.

Matrix& operator\*=(const Matrix& M) {

double total;

for (int i = 0; i < m; i++) {

for (int j = 0; j < m; j++) {

for (int k = 0; k < m; k++) {

total += (matrix[i][k] \* M.matrix[k][j]);

};

matrix[i][j] = total;

total = 0;

};

};

return \*this;

};

//STEP 7: I create the operator between matrix and constant.

Matrix& operator\*=(const double k) {

for (int i = 0; i < m; i++) {

for (int j = 0; j < m; j++) {

matrix[i][j] \*= matrix[i][j]/k;

};

};

return \*this;

};

//STEP 8: I create the norm of a square matrix.

double nMatrix() {

double nor, s=0.0;

for(int i=0; i<m; i++){

for(int j=0; j<m; j++)

s += matrix[i][j]\*matrix[i][j];

};

nor = sqrt(s);

return nor;

};

//STEP 9: I create a function to print out a matrix.

void pMatrix() {

int i,j;

for (i= 0 ; i<m ; i++) {

for(j = 0 ; j<m ; j++)

std::cout << matrix[i][j] << ", ";

std::cout << std::endl;

}

}

//STEP 10: I create a function to fill up a matrix with random values.

void fMatrix() {

for(int i=0; i<m; i++) {

for(int j=0; j<m; j++) {

matrix[i][j] = rand() % 101 - 50;

};

};

};

//STEP 11: I create an identity matrix.

void iMatrix() {

for(int i=0; i<m; i++) {

for(int j=0; j<m; j++) {

if(i==j){

matrix[i][j] = 1;

}

else {

matrix[i][j] = 0;

};

};

};

};

//STEP 12: I create a zero matrix.

void zMatrix() {

for(int i=0; i<m; i++) {

for(int j=0; j<m; j++) {

matrix[i][j] = 0;

};

};

};

//STEP 13: I create an array from a matrix.

double\* toArray() {

double\* arr = (double\*) calloc(m \* m, sizeof(double));

int counter = 0;

for (int i = 0; i < m; i++) {

for (int j = 0; j < m; j++) {

arr[counter] = matrix[j][i];

counter++;

};

};

return arr;

};

};

#endif

Main File

//STEP 0 : I include the files and projects I will need to execute my program.

# include <cstdlib>

# include <iostream>

# include <iomanip>

# include <cmath>

#include <stdio.h>

#include "project.h"

#include "r8lib.h"

#include "r8mat\_expm1.h"

#include "r8lib.cpp"

#include "r8mat\_expm1.cpp"

using namespace std;

//STEP 1 : I declare myExpMatrix function.

void myExpMatrix(Matrix &A, Matrix &I, Matrix &Z, Matrix &B);

//STEP 2 : I create the compulsory main function.

int main() {

cout << " Give the dimension of the matrix " << endl;

int n;

cin >> n;

//STEP 2.1 : I initialise matrixes and apply on them some functions for a later usage.

Matrix A(n);

Matrix B(n);

Matrix I(n);

Matrix Z(n);

A.fMatrix();

A.pMatrix();

I.iMatrix();

Z.zMatrix();

//STEP 2.2: I call all the functions I want to obtain.

double\* arr = A.toArray();

myExpMatrix(A, I, Z, B);

double \*result\_matrix = r8mat\_expm1(n, arr);

return 0;

};

//STEP 3 : I create my exponential function that iterates to compute the formula given in the task of the project and stops when the requirement we can read in the 19 ways to compute the exponential of a matrix is achieved.

void myExpMatrix(Matrix &A, Matrix &I, Matrix &Z, Matrix &B)

{

I.iMatrix();

Z.zMatrix();

double norma = B.nMatrix();

int i = 1;

int j = 1;

while ((norma / j\*(j+1))\*(1 / (1 - (B.nMatrix()/(i+2))))<i)

{

Z += I;

A \*= 1/i;

norma \*= B.nMatrix();

I \*= A;

j \*= i;

i++;

};

Z.pMatrix();

};

Output

EXCERCISE 1

For x=2

My exponential function is 7.38906

The exponential function is 7.38906

The error is 4.57323e-011

For x=3

My exponential function is 20.0855

The exponential function is 20.0855

The error is 3.20846e-011

For x=10

My exponential function is 22026.5

The exponential function is 22026.5

The error is 5.09317e-011

For x=100

My exponential function is 2.68812e+043

The exponential function is 2.68812e+043

The error is 4.95176e+027

We can conclude that the error increases with the number of iterations.

EXCERCISE 2

For n=2

Matrix

-9, 35,

22, -12,

myExpMatrix

82, 99225,

484, 593045,

expMatrix

17001469.3454, 20316281.7302,

12770234.2304, 15260073.7686,

For n=3

Matrix

-9, 35, 22,

-12, 30, 19,

15, 18, 46,

myExpMatrix

82, 99225, 3.58594e+007,

144, 177301, 6.4075e+007,

225, 275949, 9.97286e+007,

expMatrix

1.3640615435493834e+25, 9.70207232954508e+25, 1.4141585872707988e+26, 1.0797548983132326e+25, 7.679902839542458e+25, 1.1194103879097343e+26, 2.6321590315961664e+25, 1.872158732732927e+26, 2.7288287065907957e+26,

For n=4

Matrix

-9, 35, 22, -12,

30, 19, 15, 18,

46, -28, -1, 17,

1, 11, 13, 37,

myExpMatrix

82, 99225, 2.23648e+007, 6.4956e+009,

900, 1.10286e+006, 2.48579e+008, 7.21969e+010,

2116, 2.59288e+006, 5.84423e+008, 1.69739e+011,

1, 1346, 303503, 8.815e+007,

expMatrix

5.674747166343497e+21, 3.8264472900934335e+21, 4.865024223126337e+21, 5.691570279756893e+21,

1.2530047428055708e+22, 8.448934308902277e+21, 1.0742149825922509e+22, 1.2567193504365524e+22,

2.0349382642632643e+21, 1.3721464197001715e+21, 1.7445753371573229e+21, 2.0409709583796855e+21,

1.1587016545780862e+22, 7.813054355879712e+21, 9.933678907661713e+21, 1.1621366987468485e+22,

We can conclude that the exponential is more accurate the more n increases.

Program Listing

EXCERCISE 1

#include <iostream>

#include <cmath>

using namespace std;

double myexp(double x, double tol = 1e-10)

{

double s = 0;

double t = 1;

int i = 1;

while (t > tol)

{

s += t;

t \*= x/i;

i++;

};

return s;

};

int main()

{

cout << " Enter a number to evaluate the function " << endl;

double x;

cin >> x;

double myExp;

myExp = myexp(x);

cout << " My exponential function is " << myExp << endl;

double Exp;

Exp = exp(x);

cout << " The exponential function is " << Exp << endl;

double Error;

Error = abs(myExp-Exp);

cout << " The error is " << Error << endl;

return 0;

};

EXCERCISE 2

Header File

# include <iomanip>

# include <iostream>

# include <cstdlib>

#include <cmath>

#include "r8mat\_expm1.h"

#ifndef PROJECT2\_H

#define PROJECT2\_H

using namespace std;

class Matrix {

private:

double\*\* matrix;

int m;

public:

Matrix(int n) {

if(n >= 0)

m = n;

matrix = (double \*\*) calloc(m, sizeof(double \*\*));

for (int i = 0; i < m; i++) { matrix[i] = (double \*) calloc(m, sizeof(double)); }

};

Matrix(const Matrix& A) {

for(int i=0; i<m; i++) {

for(int j=0; j<m; j++) {

matrix[i][j] = A.matrix[i][j];

};

};

};

~Matrix() { };

Matrix& operator=(const Matrix& M) {

if(this != &M) { matrix = M.matrix; }

return \*this;

};

Matrix& operator+=(const Matrix& M) {

for (int i = 0; i < m; i++) {

for (int j = 0; j < m; j++) {

matrix[i][j] += M.matrix[i][j];

};

};

return \*this;

};

Matrix& operator\*=(const Matrix& M) {

double total;

for (int i = 0; i < m; i++) {

for (int j = 0; j < m; j++) {

for (int k = 0; k < m; k++) {

total += (matrix[i][k] \* M.matrix[k][j]);

};

matrix[i][j] = total;

total = 0;

};

};

return \*this;

};

Matrix& operator\*=(const double k) {

for (int i = 0; i < m; i++) {

for (int j = 0; j < m; j++) {

matrix[i][j] \*= matrix[i][j]/k;

};

};

return \*this;

};

double nMatrix() {

double L, sum=0.0;

for(int i=0; i<m; i++){

for(int j=0; j<m; j++)

sum += matrix[i][j]\*matrix[i][j];

};

L = sqrt(sum);

return L;

};

void pMatrix() {

int i,j;

for (i= 0 ; i<m ; i++) {

for(j = 0 ; j<m ; j++)

std::cout << matrix[i][j] << ", ";

std::cout << std::endl;

}

}

void fMatrix() {

for(int i=0; i<m; i++) {

for(int j=0; j<m; j++) {

matrix[i][j] = rand() % 101 - 50;

};

};

};

void iMatrix() {

for(int i=0; i<m; i++) {

for(int j=0; j<m; j++) {

if(i==j){

matrix[i][j] = 1;

}

else {

matrix[i][j] = 0;

};

};

};

};

void zMatrix() {

for(int i=0; i<m; i++) {

for(int j=0; j<m; j++) {

matrix[i][j] = 0;

};

};

};

double\* toArray() {

double\* arr = (double\*) calloc(m \* m, sizeof(double));

int counter = 0;

for (int i = 0; i < m; i++) {

for (int j = 0; j < m; j++) {

arr[counter] = matrix[j][i];

counter++;

};

};

return arr;

};

};

#endif

Main File

# include <cstdlib>

# include <iostream>

# include <iomanip>

# include <cmath>

#include <stdio.h>

#include "project.h"

#include "r8lib.h"

#include "r8mat\_expm1.h"

#include "r8lib.cpp"

#include "r8mat\_expm1.cpp"

using namespace std;

void myExpMatrix(Matrix &A, Matrix &I, Matrix &Z, Matrix &B);

int main() {

cout << " Give the dimension of the matrix " << endl;

int n;

cin >> n;

Matrix A(n);

Matrix B(n);

Matrix I(n);

Matrix Z(n);

A.fMatrix();

B = A;

A.pMatrix();

I.iMatrix();

Z.zMatrix();

myExpMatrix(A, I, Z, B);

double\* arr = A.toArray();

double \*result\_matrix = r8mat\_expm1(n, arr);

cout << result\_matrix <<endl;

return 0;

};

void myExpMatrix(Matrix &A, Matrix &I, Matrix &Z, Matrix &B)

{

I.iMatrix();

Z.zMatrix();

double norma = B.nMatrix();

int i = 1;

int j = 1;

while ((norma / j\*(j+1))\*(1 / (1 - (B.nMatrix()/(i+2))))<i)

{

Z += I;

A \*= 1/i;

norma \*= B.nMatrix();

I \*= A;

j \*= i;

i++;

};

Z.pMatrix();

};