

UNIVERSITY OF ANTWERP

SCIENTIFIC PROGRAMMING

Second Session
Exercise 1

Armin Halilovic - s0122210

August 22, 2016

Contents

1	Problem	2
2	Using the program	3
3	Course of action	5
4	Solutions	5
4.1	Solving the first system of equations	5
4.1.1	LU decomposition	5
4.1.2	Solving the system	6
4.1.3	Condition number of A	6
4.1.4	Maximum possible error	7
4.2	Solving the system after changing $a_{2,2}$	8
5	Conclusion	9
	Appendices	10
A	Code	10
A.1	main.cpp	10
A.2	functions.h	11
A.3	functions.cpp	11
B	Output	15
B.1	output.txt	15

1 Problem

We will solve the system of linear equations

$$\sum_{j=1}^n (1+i)^{j-1} x_j = \frac{(1+i)^n - 1}{i} \quad 1 \leq i \leq n$$

for $n = 10, \dots, 15$ using Gauss-elimination with partial pivoting. Then, we will calculate and discuss the condition number of the coefficient matrix of the linear system.

This will be done using C++ and the GNU Scientific Library. In section 4, the solution is reached, using *only* the most important parts from our code. Some basic knowledge of GSL is assumed.

$$Ax = y$$

$$A = \begin{bmatrix} (1+i)^0 & (1+i)^1 & \dots & (1+i)^9 & 0 & 0 & 0 & 0 & 0 \\ (1+i)^0 & (1+i)^1 & \dots & (1+i)^9 & 0 & 0 & 0 & 0 & 0 \\ (1+i)^0 & (1+i)^1 & \dots & (1+i)^9 & 0 & 0 & 0 & 0 & 0 \\ (1+i)^0 & (1+i)^1 & \dots & (1+i)^9 & 0 & 0 & 0 & 0 & 0 \\ (1+i)^0 & (1+i)^1 & \dots & (1+i)^9 & 0 & 0 & 0 & 0 & 0 \\ (1+i)^0 & (1+i)^1 & \dots & (1+i)^9 & 0 & 0 & 0 & 0 & 0 \\ (1+i)^0 & (1+i)^1 & \dots & (1+i)^9 & 0 & 0 & 0 & 0 & 0 \\ (1+i)^0 & (1+i)^1 & \dots & (1+i)^9 & 0 & 0 & 0 & 0 & 0 \\ (1+i)^0 & (1+i)^1 & \dots & (1+i)^9 & 0 & 0 & 0 & 0 & 0 \\ (1+i)^0 & (1+i)^1 & \dots & (1+i)^9 & 0 & 0 & 0 & 0 & 0 \\ (1+i)^0 & (1+i)^1 & \dots & (1+i)^9 & (1+i)^{10} & 0 & 0 & 0 & 0 \\ (1+i)^0 & (1+i)^1 & \dots & (1+i)^9 & (1+i)^{10} & (1+i)^{11} & 0 & 0 & 0 \\ (1+i)^0 & (1+i)^1 & \dots & (1+i)^9 & (1+i)^{10} & (1+i)^{11} & (1+i)^{12} & 0 & 0 \\ (1+i)^0 & (1+i)^1 & \dots & (1+i)^9 & (1+i)^{10} & (1+i)^{11} & (1+i)^{12} & (1+i)^{13} & 0 \\ (1+i)^0 & (1+i)^1 & \dots & (1+i)^9 & (1+i)^{10} & (1+i)^{11} & (1+i)^{12} & (1+i)^{13} & (1+i)^{14} \end{bmatrix}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ . \\ . \\ x_{15} \end{bmatrix} \quad y = \begin{bmatrix} \frac{(1+i)^{10}-1}{i} \\ \frac{(1+i)^{11}-1}{i} \\ \frac{(1+i)^{12}-1}{i} \\ \frac{(1+i)^{13}-1}{i} \\ \frac{(1+i)^{14}-1}{i} \\ \frac{(1+i)^{15}-1}{i} \\ \frac{(1+i)^{15}-1}{i} \\ \frac{(1+i)^{15}-1}{i} \\ \frac{(1+i)^{15}-1}{i} \\ \frac{(1+i)^{15}-1}{i} \\ \frac{(1+i)^{15}-1}{i} \\ \frac{(1+i)^{15}-1}{i} \\ \frac{(1+i)^{15}-1}{i} \\ \frac{(1+i)^{15}-1}{i} \\ \frac{(1+i)^{15}-1}{i} \end{bmatrix}$$

2 Using the program

All of the C++ code for the program can be found in the main.cpp, functions.cpp and functions.h files, and in appendix A of this document. The output of the program is in output.txt and in appendix B.

To compile and run the program, execute the following commands in the build/ directory:

```
cmake ..  
make  
./sys_lineq
```

3 Course of action

What follows is a brief set of steps we will use to solve the exercise:

1. Load the input matrix A and vector Y.
2. Find the LU decomposition of A.
3. Solve the system using the LU decomposition.
Also, find the residual error vector.¹
4. Calculate the condition number of the matrix.
5. Calculate the maximum possible error of the solution.

4 Solutions

4.1 Solving the first system of equations

Initializing all of the necessary structs and loading the input matrix and vector is a trivial task, so we will start at the LU decomposition step.

4.1.1 LU decomposition

We know that $PA = LU$. GSL makes it very easy to find LU decompositions; `gsl_linalg_LU_decomp` will find the LU decomposition by executing Gaussian Elimination with partial pivoting.

```
gsl_matrix_memcpy(LU, A);  
gsl_linalg_LU_decomp(LU, P, &sigNum);
```

`gsl_linalg_LU_decomp` writes the result to its first argument, so we will copy A to a new matrix LU first, as we will need A later on. P will contain the permutation matrix, which is not relevant for this exercise.

This will result in the following output:

Matrix LU, result of LU decomposition:		
5.084	-5.832	9.155
0.594217151848938	5.63947442958301	1.47294197482297
0.20279307631786	-0.547978507128854	0.0715699307609119

¹We cannot find the error vector, since we do not have the exact solution for x in $Ax = y$

4.1.2 Solving the system

The code for this part is very straightforward:

```
gsl_linalg_LU_solve(LU, P, Y, X);  
gsl_linalg_LU_refine(A, LU, P, Y, X, r);
```

`gsl_linalg_LU_solve` will use the LU decomposition and permutation matrix to solve the system of equations. The solution for the system will be stored in `X`. `gsl_linalg_LU_refine` is then used to find the residual vector `r`.

The residual vector is the vector `r` so that

$$r = y - Ax$$

where y is known exactly and Ax is calculated.

We will get the output:

Vector X, result of solving with LU decomposition:

```
-7.57416762700087  
0.0158780881981559  
5.13453514211295
```

Vector r, residual of solving by LU decomposition:

```
4.0146050319364e-14  
4.8599800162611e-15  
-1.95862184650587e-14
```

The vector `X` we have found here can be interpreted as a point in a 3D space, where, for example

$$x = -7.57416762700087$$

$$y = 0.0158780881981559$$

$$z = 5.13453514211295$$

This point could then represent the intersection point of the three planes given in the system of this exercise.

We notice that the values for the residual vector are extremely small, this indicates that the resulting vector `X` is very precise.

4.1.3 Condition number of A

To calculate the condition number of `A`, we will use the following definition:

$$\kappa(A) = \frac{|max(S)|}{|min(S)|}$$

where `S` is the vector of singular values of `A`. In GSL, we can find `S` by using `gsl_linalg_SV_decomp`.

```

gsl_linalg_SV_decomp(U, V, S, work);

double condNumber, minS, maxS;
minS = gsl_vector_get(S, 0); maxS = gsl_vector_get(S, 0);
for (int j = 0; j < size_j; j++) {
    if (gsl_vector_get(S, j) < minS) minS = gsl_vector_get(S, j);
    if (gsl_vector_get(S, j) > maxS) maxS = gsl_vector_get(S, j);
}
condNumber = fabs(maxS) / fabs(minS);

```

After doing this, we can generate the following output:

```

Vector S, singular values of A, result of doing SV decomposition:
    13.7188626226398
     6.13888725147292
     0.0243650331438208

```

```

Condition number = 13.7188626226398 / 0.0243650331438208 = 563.055364696643

```

We get a condition number of about 563. Informally, we know that

As a rule of thumb, if the condition number $\kappa(A) = 10^k$, then you may lose up to k digits of accuracy on top of what would be lost to the numerical method due to loss of precision from arithmetic methods.²

This way, we can quickly estimate that with a condition number of 563 we *may* lose approximately 3 digits of accuracy (in the vector X) on top of the loss caused by machine precision.

Next, we will use the condition number and more formally describe the maximum inaccuracy which *may* occur after these calculations.

4.1.4 Maximum possible error

We know the following inequalities which express that the maximum possible relative errors are bounded:

$$\frac{\|y - Ax_*\|}{\|A\| \|x_*\|} \leq \rho\epsilon, \quad \frac{\|x - x_*\|}{\|x_*\|} \leq \rho\kappa(A)\epsilon$$

Here, ϵ is the relative machine precision eps. For a double, this is 2.22E-16.³ If we take the approximately worst case for p, which is around 10, then we get that $\rho\epsilon = 2.22\text{E-}15$

²https://en.wikipedia.org/wiki/Condition_number

³https://en.wikipedia.org/wiki/Machine_epsilon#Values_for_standard_hardware_floating_point_arithmetics

In the second formula, we see a better way to describe the impact that the condition number has on the accuracy of the calculations for x . For $\rho\kappa(A)\epsilon$, we find:

$$\begin{aligned}\rho\kappa(A)\epsilon &\approx 10 * 563.055364696643 * 2.22\text{E-}16 \\ \rho\kappa(A)\epsilon &\approx 1.24998290962654746\text{E-}12 \\ \rho\kappa(A)\epsilon &\approx 1.25\text{E-}12\end{aligned}$$

We can now say that our result for x will have a maximal error of $1.25\text{E-}12$, with respect to the exact solutions for x .

4.2 Solving the system after changing $a_{2,2}$

In this section, we change $a_{2,2}$ from -4.273 to -4.275 , and just repeat all of the steps. The most important parts of the output are shown below.

Vector X, result of solving with LU decomposition:

-7.57509282124123
0.0157630382743384
5.13497563543488

Vector r, residual of solving by LU decomposition:

9.58958990610966e-14
1.12524872037495e-14
-4.64732888184761e-14

Condition number = $13.7190169534295 / 0.024538406356238 = 559.083452864162$

We notice that the condition number has decreased, which means the maximum inaccuracy that *may* occur also decreased. We see this when calculating $\rho\kappa(A)\epsilon$:

$$\begin{aligned}\rho\kappa(A)\epsilon &\approx 10 * 559.083452864162 * 2.22\text{E-}16 \\ \rho\kappa(A)\epsilon &\approx 1,241165265\text{E-}12 \\ \rho\kappa(A)\epsilon &\approx 1.24\text{E-}12\end{aligned}$$

Coincidentally, the residual vector *increased* in this case, which means the calculated solution for the system of equations became less accurate.

This can be interpreted graphically as the point in 3D space moving away from the intersection of the three planes.

Thus, we have found that changing $a_{2,2}$ from -4.273 to -4.275 has decreased the condition number for the matrix, but increased the residual vector. This is perfectly possible, with the condition number we can only make a statement about the maximum *possible* error, we cannot say anything about the actual error without knowing the exact solution.

5 Conclusion

We have solved the given system of equations and found the following vector as a solution.

Vector X, result of solving with LU decomposition:

-7.57509282124123
0.0157630382743384
5.13497563543488

The maximum possible error of this vector X, with respect to the exact solutions for x , is approximately 1.25E-12.

Graphically, this can be interpreted to be the point in 3D space, that is the intersection of three planes.

We have found that changing $a_{2,2}$ from -4.273 to -4.275 decreased the maximum possible error to approximately 1.24E-12. However, we noticed that the actual error of the vector X increased in this case.

Graphically, we can say that the point moved away from the intersection of the planes.

Appendices

A Code

A.1 main.cpp

```
#include <fstream>
#include <iostream>
#include <gsl/gsl_vector_double.h>
#include <gsl/gsl_matrix_double.h>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_math.h>
#include <gsl/gsl_vector.h>

#include "functions.h"

// formula:  $\sum_{j=1}^n ((1+i)^{(j-1)}) * x_j = ((1+i)^n - 1) / i$ 

double formulaLeft(double i, double j)
{
    return pow(1+i, j-1);
}

double formulaRight(double i, int n)
{
    return (pow(1+i, n) - 1) / i;
}

void fillA (gsl_matrix* A, int i)
{
    for (size_t r = 0; r < A->size1; ++r) {
        for (size_t j = 0; j < A->size2; ++j) {
            gsl_matrix_set(A, r, j, formulaLeft(i, j));
        }
    }
}

void fillY (gsl_vector* Y, int i)
{
    int n = 10;
    for (size_t r = 0; r < Y->size; ++r) {
        gsl_vector_set(Y, r, formulaRight(i, n));
        n++;
    }
}

void fillSystem (gsl_matrix* A, gsl_vector *Y, int n)
{
}

int main (void)
{
    static const int size_i = 6, size_j = 15;

    //Open the file to write output to
    std::ofstream output("output.txt", std::ofstream::out);
```

```

// Initialize matrices A and Y for the equation Ax = Y
gsl_matrix *A = gsl_matrix_alloc( size_i , size_j );
gsl_vector *Y = gsl_vector_alloc( size_i );

//Put the input data into matrix A and vector Y
for (int n = 10; n < 16; n++) {
    //fillSystem(A, Y, n);
}
fillSystem(A, Y, 15);
printMatrix(A, "A");
printVector(Y, "Y");

//Do the exercise for the input data
//solve(A, Y, output);

output << "\n=====\\n" << std::endl;
std::cout << "\n=====\\n" << std::endl;

//Free the memory
output.close();
gsl_matrix_free(A);
gsl_vector_free(Y);

return 0;
}

```

A.2 functions.h

```

#ifndef PROJECT_FUNCTIONS_H
#define PROJECT_FUNCTIONS_H

#include <fstream>
#include <gsl/gsl_vector_double.h>
#include <gsl/gsl_matrix_double.h>

void printVector(const gsl_vector * v, std::string string);
void printVector(const gsl_vector * v, std::string string, std::ostream &);
void printVectorCoutAndFile(const gsl_vector * v, std::string string, std::ostream &);
void printMatrix(const gsl_matrix *m, std::string string);
void printMatrix(const gsl_matrix *m, std::string string, std::ostream &);
void printMatrixCoutAndFile(const gsl_matrix *m, std::string string, std::ostream &);

void solve(gsl_matrix *A, gsl_vector *Y, std::ostream &);

#endif //PROJECT_FUNCTIONS_H

```

A.3 functions.cpp

```

#include "functions.h"

#include <cmath>
#include <iomanip>
#include <iostream>

```

```

#include <gsl/gsl_linalg.h>
#include <gsl/gsl_matrix_double.h>

const int printWidth = 5;
const int printPrecision = 15;

void printVector(const gsl_vector * v, std::string string) {
    std::cout << "Vector " << string << ":\n";
    for (unsigned int i = 0; i < v->size; i++) {
        std::cout << std::setw(printWidth) << std::setprecision(printPrecision) <<
            gsl_vector_get(v, i) << "\n";
    }
    std::cout << "\n";
}

void printVector(const gsl_vector * v, std::string string, std::ostream &out) {
    out << "Vector " << string << ":\n";
    for (unsigned int i = 0; i < v->size; i++) {
        out << std::setw(printWidth) << std::setprecision(printPrecision) <<
            gsl_vector_get(v, i) << "\n";
    }
    out << "\n";
}

void printVectorCoutAndFile(const gsl_vector * v, std::string string, std::ostream &out) {
    std::cout << "Vector " << string << ":\n";
    out << "Vector " << string << ":\n";
    for (unsigned int i = 0; i < v->size; i++) {
        std::cout << std::setw(printWidth) << std::setprecision(printPrecision) <<
            gsl_vector_get(v, i) << "\n";
        out << std::setw(printWidth) << std::setprecision(printPrecision) <<
            gsl_vector_get(v, i) << "\n";
    }
    std::cout << "\n";
    out << "\n";
}

void printMatrix(const gsl_matrix *m, std::string string) {
    std::cout << "Matrix " << string << ":\n";

    for (unsigned int i = 0; i < m->size1; i++) {
        for (unsigned int j = 0; j < m->size2; j++) {
            std::cout << std::setw(printWidth) << std::setprecision(printPrecision) <<
                gsl_matrix_get(m, i, j);
        }
        std::cout << "\n";
    }
    std::cout << "\n";
}

void printMatrix(const gsl_matrix *m, std::string string, std::ostream &out) {
    out << "Matrix " << string << ":\n";
    for (unsigned int i = 0; i < m->size1; i++) {
        for (unsigned int j = 0; j < m->size2; j++) {
            out << std::setw(printWidth) << std::setprecision(printPrecision) <<
                gsl_matrix_get(m, i, j);
        }
        out << "\n";
    }
    out << "\n";
}

```

```

void printMatrixCoutAndFile(const gsl_matrix *m, std::string string, std::ostream &out) {
    std::cout << "Matrix " << string << ":\n";
    out << "Matrix " << string << ":\n";
    for (unsigned int i = 0; i < m->size1; i++) {
        for (unsigned int j = 0; j < m->size2; j++) {
            std::cout << std::setw(printWidth) << std::setprecision(printPrecision) <<
            gsl_matrix_get(m, i, j);
            out << std::setw(printWidth) << std::setprecision(printPrecision) <<
            gsl_matrix_get(m, i, j);
        }
        std::cout << "\n";
        out << "\n";
    }
    std::cout << "\n";
    out << "\n";
}

void solve(gsl_matrix *A, gsl_vector *Y, std::ostream &out) {
    size_t size_i, size_j;
    size_i = A->size1; size_j = A->size2;

    // Initialize all the necessary matrices and vectors
    gsl_matrix *LU = gsl_matrix_alloc(size_i, size_j),
        *U = gsl_matrix_alloc(size_i, size_j),
        *V = gsl_matrix_alloc(size_j, size_j);

    gsl_vector *X = gsl_vector_alloc(size_j),
        *r = gsl_vector_alloc(size_j),
        *S = gsl_vector_alloc(size_j),
        *work = gsl_vector_alloc(size_j);

    //gsl_permutation and pInt are necessary to do the LU decomposition
    gsl_permutation *P = gsl_permutation_alloc(size_i);
    int sigNum;

    //Copy the contents in matrix A to matrices LU and U
    //We will work with LU and U so A won't be overwritten
    gsl_matrix_memcpy(LU, A);
    gsl_matrix_memcpy(U, A);

    //Do the LU decomposition
    //The algorithm used in the decomposition is Gaussian Elimination with partial pivoting
    gsl_linalg_LU_decomp(LU, P, &sigNum);

    //Solve the system of equations, put the result in X and the residual vector in r
    gsl_linalg_LU_solve(LU, P, Y, X);
    gsl_linalg_LU_refine(A, LU, P, Y, X, r);

    //Do a singular value decomposition, we will use values of S to calculate the condition
    //number of A
    gsl_linalg_SV_decomp(U, V, S, work);

    //The condition number we will use is max(S) / min(S)
    double condNumber, minS, maxS;
    minS = gsl_vector_get(S, 0); maxS = gsl_vector_get(S, 0);
    for (int j = 0; j < size_j; j++) {
        if (gsl_vector_get(S, j) < minS) minS = gsl_vector_get(S, j);
        if (gsl_vector_get(S, j) > maxS) maxS = gsl_vector_get(S, j);
    }
    condNumber = fabs(maxS) / fabs(minS);

    //Write out all of the results

```

```

printMatrixCoutAndFile(A, "Input Matrix A", out);
printVectorCoutAndFile(Y, "Input Vector Y", out);
printMatrixCoutAndFile(LU, "LU, result of LU decomposition", out);
printVectorCoutAndFile(X, "X, result of solving with LU decomposition", out);
printVectorCoutAndFile(r, "r, residual of solving by LU decomposition", out);
printVectorCoutAndFile(S, "S, singular values of A, result of doing SV decomposition",
    out);

std::cout << "Calculating condition number by: abs(max(singular values)) /
    abs(min(singular values)):\n\t";
std::cout << "Condition number = " << fabs(maxS) << " / " << fabs(minS) << " = "
    << condNumber << "\n";
out << "Calculating condition number by: abs(max(singular values)) / abs(min(singular
    values)):\n\t";
out << "Condition number = " << fabs(maxS) << " / " << fabs(minS) << " = " <<
    condNumber << "\n\n";

//Free the memory
gsl_matrix_free (LU); gsl_matrix_free (U); gsl_matrix_free (V);
gsl_vector_free (X); gsl_vector_free (r); gsl_vector_free (S); gsl_vector_free (work);
gsl_permutation_free(P);
}

```

B Output

B.1 output.txt
