University of Antwerp

Scientific Programming

Systems of Linear Equations
Exercise 1

Armin Halilovic - s0122210

November 6, 2015

Contents

1 Problem

We are given a system of linear equations $Ax = y$ that looks like

$$\begin{bmatrix} 3.021 & 2.714 & 6.913 \\ 1.031 & -4.273 & 1.121 \\ 5.084 & -5.832 & 9.155 \end{bmatrix} x = \begin{bmatrix} 12.648 \\ -2.121 \\ 8.407 \end{bmatrix}$$

Such a system can be used to represent three planes and find their intersections. A solution for x would then be a 3D point that is an intersection of the three planes.
We will solve the system using LU decomposition and will discuss the results and their validity. Then, we will change $a_{2,2}$ from -4.273 to -4.275, and observe the difference in the results.

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

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
```

[1] We 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 inequalites which express that the maximum possible relative errors are bounded:

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

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 $\rho\epsilon = 2.22\text{E-}15$

[2] https://en.wikipedia.org/wiki/Condition_number
[3] https://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:

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

We can now say that our result for x will have a maximal error of 1.25E-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 decreassed. We see this when calculating $\rho\kappa(A)\epsilon$:

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

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 <math.h>
#include <stdio.h>
#include <fstream>
#include <iostream>
#include <gsl/gsl_blas.h>
#include <gsl/gsl_linalg.h>
#include <gsl/gsl_pow_int.h>
#include <gsl/gsl_vector_double.h>
#include <gsl/gsl_matrix_double.h>
#include <cmath>
#include "functions.h"

int main (void) {
    static const int size_i = 3, size_j = 3;

    //Input data which will be used to create and solve systems of equations
    double a[][ size_j ] = {{3.021,  2.174,  6.913},
                            {1.031, -4.273, 1.121},
                            {5.084, -5.832, 9.155}};
    double y[] = {12.648,
                  -2.121,
                  8.407};

    //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_j );

    //Put the input data into matrix A and vector Y
    for (int i = 0; i < size_i; i++) {
        for (int j = 0; j < size_j; j++) {
            gsl_matrix_set(A, i, j, a[i][j]);

            if (i == size_j-1) gsl_vector_set(Y, j, y[j]);
        }
    }

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

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

    //Introduce a change into A
    output << "Changed a_1,1 from " << gsl_matrix_get(A, 1, 1) <<" to -4.275" <<
     std::endl;
    std::cout << "Changed a_1,1 from " << gsl_matrix_get(A, 1, 1) <<" to -4.275" <<
     std::endl;
    gsl_matrix_set(A, 1, 1, -4.275);

    //Do the exercise for the changed matrix A
```

```
    solve(A, Y, output);

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

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 <stdio.h>
#include <stddef.h>
#include <stdint.h>
#include <math.h>
#include <iomanip>
#include <iostream>
#include <gsl/gsl_blas.h>
#include <gsl/gsl_linalg.h>
#include <gsl/gsl_pow_int.h>
#include <gsl/gsl_matrix_double.h>
#include <cmath>


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(21) << std::setprecision(15) << 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(21) << std::setprecision(15) << 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(21) << std::setprecision(15) << gsl_vector_get(v, i) << "\n";
        out << std::setw(21) << std::setprecision(15) << 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(21) << std::setprecision(15) << 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(21) << std::setprecision(15) << 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(21) << std::setprecision(15) << gsl_matrix_get(m, i, j);
            out << std::setw(21) << std::setprecision(15) << 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

```
Matrix Input Matrix A:
                3.021                2.174                6.913
                1.031               −4.273                1.121
                5.084               −5.832                9.155

Vector Input Vector Y:
               12.648
               −2.121
                8.407

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

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

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

Calculating condition number by: abs(max(singular values)) / abs(min(singular values)):
   Condition number = 13.7188626226398 / 0.0243650331438208 = 563.055364696643


==============================

Changed a_1,1 from −4.273 to −4.275
Matrix Input Matrix A:
                3.021                2.174                6.913
                1.031               −4.275                1.121
                5.084               −5.832                9.155

Vector Input Vector Y:
               12.648
               −2.121
                8.407

Matrix LU, result of LU decomposition:
                5.084               −5.832                9.155
    0.594217151848938    5.63947442958301    1.47294197482297
     0.20279307631786   −0.548333150105779   0.0720922992877012

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

Vector S, singular values of A, result of doing SV decomposition:
      13.7190169534295
      6.13993401415102
     0.024538406356238

Calculating condition number by: abs(max(singular values)) / abs(min(singular values)):
    Condition number = 13.7190169534295 / 0.024538406356238 = 559.083452864162
