University of Antwerp

Scientific Programming

Second Session

Exercise 1

Armin Halilovic - s0122210

August 22, 2016

Contents

1 Problem

We have to 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 $n = 10, ..., 15$ using Gauss-elimination with partial pivoting. Then, we calculate and discuss the results and the condition numbers of the coeffient matrices of the linear systems.
This is done using C++ and the GNU Scientific Library. In section 4, we explain how the solutions are reached, using *only* the most important parts from our code. Some basic knowledge of 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 stored in output/output_n_*.txt. The output file for $n = 10$ can be found in appendix B.

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

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

3 Course of action

The systems of linear equations have the form $Ax = y$. For example, if n = 10, we get:

$$A = \begin{bmatrix} (1+i)^0 & (1+i)^1 & ... & (1+i)^9 \\ (1+i)^0 & (1+i)^1 & ... & (1+i)^9 \\ (1+i)^0 & (1+i)^1 & ... & (1+i)^9 \\ (1+i)^0 & (1+i)^1 & ... & (1+i)^9 \\ (1+i)^0 & (1+i)^1 & ... & (1+i)^9 \\ (1+i)^0 & (1+i)^1 & ... & (1+i)^9 \\ (1+i)^0 & (1+i)^1 & ... & (1+i)^9 \\ (1+i)^0 & (1+i)^1 & ... & (1+i)^9 \\ (1+i)^0 & (1+i)^1 & ... & (1+i)^9 \\ (1+i)^0 & (1+i)^1 & ... & (1+i)^9 \end{bmatrix} x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ . \\ . \\ x_{10} \end{bmatrix} y = \begin{bmatrix} \frac{(1+i)^{10}-1}{i} \\ \frac{(1+i)^{10}-1}{i} \\ \frac{(1+i)^{10}-1}{i} \\ . \\ . \\ . \\ \frac{(1+i)^{10}-1}{i} \end{bmatrix}$$

The i's in the systems can be chosen freely, as long as the constraint $1 \leq i \leq n$ holds true. We choose the i's to be the row indices of the systems.
What follows is a brief set of steps we use to solve the matrices.

1. Load the input matrix A and vector Y.
2. Find the LU decomposition of A.

   Gauss-elimination with partial pivoting happens here.
3. Solve the system using the LU decomposition.

   Also, find the residual error vector.[1]
4. Calculate the condition number of the matrix.

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

4 Solutions

4.1 Solution for n=10

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

4.1.1 LU decomposition

To find LU decompositions, we use gsl_linalg_LU_decomp. This function makes use of Gaussian Elimination with partial pivoting to find the LU decompositions.

```
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 copy A to a new matrix LU first, as we need A later on. P contains the permutation matrix, which is not relevant for this exercise. The resulting matrix LU can be found in the output file for n = 10.

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 uses the LU decomposition and permutation matrix to solve the system of equations. The solution for the system is 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 get the output:

```
Vector X, result of solving with LU decomposition:
            1
            1
            1
            1
            1
            1
            1
            1
            1
            1

Vector r, residual of solving by LU decomposition:
   4.3321e-05
  -8.1093e-05
   6.4058e-05
  -2.8163e-05
   7.6338e-06
  -1.3279e-06
   1.4868e-07
   -1.036e-08
   4.0865e-10
   -6.966e-12
```

We notice that the values for the residual vector are very small, this indicates that the resulting vector X is very precise. This does not necessarily mean that the solution to the system is exactly this vector. However, if we check, we see that the formula

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

indeed is true when all x_j are 1.

4.1.3 Condition number of A

To calculate the condition number of A, we 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);
```

We get the following output:

```
Vector S, singular values of A, result of doing SV decomposition:
   2.6054e+09
    1.225e+07
   1.4309e+05
       3220.7
       130.72
       9.8208
       1.2767
      0.13121
    0.0059818
   9.6998e-05

Condition number = 2.6054e+09 / 9.6998e-05 = 2.6861e+13
```

We get a condition number of about 2.7e+13.
Informally, we know that the larger the order of the condition number is, the larger the possible error (in the X vector) can be. For very large condition numbers, a small pertubation in one of the matrix' cells could cause a large change in the results. We test and compare the effect of small perturbations later on.

4.2 Solution for n=11

We get the following results

```
Vector X, result of solving with LU decomposition:
        1.001
      0.99808
       1.0016
```

```
      0.9992
      1.0002
     0.99995
           1
           1
           1
           1
           1

Vector r, residual of solving by LU decomposition:
  -0.0013454
   0.0027319
  -0.0023955
   0.0011972
 -0.00037823
  7.9061e-05
 -1.1093e-05
  1.0335e-06
 -6.1298e-08
  2.0934e-09
  -3.131e-11

Condition number = 6.8216e+10 / 4.1701e-05 = 1.6358e+15
```

We notice that the condition number has increased, which means the possible error that *may* occur also increased.

4.3 Solution for n=12

We get the following results

```
Vector X, result of solving with LU decomposition:
     0.97184
      1.0587
     0.94665
       1.028
     0.99058
      1.0021
     0.99966
           1
           1
           1
           1
           1

Vector r, residual of solving by LU decomposition:
    0.065446
    -0.13345
     0.11782
   -0.059773
    0.019457
  -0.0042842
  0.00065322
 -6.9152e-05
  4.9923e-06
 -2.3452e-07
    6.462e-09
 -7.9229e-11

Condition number = 1.9698e+12 / 1.1463e-05 = 1.7184e+17
```

The condition number has increased again.

4.4 Solution for n=13

We get the following results

```
Vector X, result of solving with LU decomposition:
       2.6882
        -2.65
       4.4667
     -0.91978
       1.6926
      0.82808
       1.0302
      0.99622
       1.0003
      0.99998
            1
            1
            1

Vector r, residual of solving by LU decomposition:
      -66.275
       142.06
      -133.52
       73.034
      -25.974
       6.3446
      -1.0941
      0.13448
    -0.011716
   0.00070686
  -2.8076e-05
   6.6017e-07
  -6.9591e-09

Condition number = 6.2191e+13 / 6.1062e-05 = 1.0185e+18
```

The condition number has increased again.

4.5 Solution for n=14

We get the following results

```
Vector X, result of solving with LU decomposition:
      0.17084
       2.3299
      0.30066
      0.98717
       1.1907
      0.89236
        1.033
      0.99344
       1.0009
      0.99992
            1
            1
            1
            1

Vector r, residual of solving by LU decomposition:
```

```
     -941.62
        2021
     -1911.7
      1060.2
     -386.08
      97.825
     -17.794
      2.3584
    -0.22827
    0.015965
 -0.00078561
  2.5795e-05
 -5.0724e-07
  4.5171e-09

Condition number = 2.1314e+15 / 0.00017035 = 1.2512e+19
```

The condition number has increased again.

4.6 Solution for n=15

We get the following results

```
Vector X, result of solving with LU decomposition:
     -6721.2
       15434
      -15775
      9552.4
     -3837.3
      1086.8
     -222.48
      35.065
     -2.8712
      1.3269
     0.97978
      1.0009
     0.99997
           1
           1

Vector r, residual of solving by LU decomposition:
      -12716
       28066
      -27411
       15768
     -5991.5
      1596.1
     -308.34
       44.01
     -4.6765
     0.36907
   -0.021347
   0.0008794
   -2.444e-05
  4.1093e-07
 -3.1584e-09

Condition number = 7.8805e+16 / 0.00074503 = 1.0577e+20
```

The condition number has increased again.

5 Discussion

We notice that when n increases, so does the condition number of each system. This makes sense, as larger and larger numbers are added to the system while the rest of the system stays the same. The scale of the system becomes larger each time.
As a result of this, at each increment of n, X seems to move farther away from the vector of all ones. The residual vector becomes worse each time as well.

For each n, we looked at the result of changing the cell (10, 2) of the input matrix to 1.
For n = 10, we found:

```
Vector X, result of solving with LU decomposition:
      1.5211
  -0.0052109
      1.8263
     0.61862
      1.1093
     0.97977
      1.0024
     0.99982
           1
           1
```

For n = 15, we found:

```
Vector X, result of solving with LU decomposition:
      9.2501
  -0.0060889
     -19.703
      28.878
     -17.302
      8.4653
     -1.0681
      1.4059
     0.94242
      1.0059
     0.99956
           1
           1
           1
           1
```

We see that the small perturbation cause a relatively small change in the solution when n = 10, where the condition number is relatively small compared to the other ones. On the other hand, the small perturbation caused a very large change in the solution where n = 15. If we look at the l_2 norms for example, the norm changed from 3.16 to 3.50 when n = 10, and changed from 25287.12 to 41.10 when n = 15.

Appendices

A Code

A.1 main.cpp

```
#include <fstream>
#include <iostream>
#include <cmath>
#include <iomanip>
#include <gsl/gsl_vector_double.h>
#include <gsl/gsl_matrix_double.h>
#include <sstream>

#include "functions.h"

int main (void)
{
    for (int n = 10; n <= 15; n++) {
        // open the file to write output to
        std::stringstream ss; ss << n;
        std::ofstream output("output_n_" + ss.str() + ".txt", std::ofstream::out);

        // initialize matrices A and Y for the equation Ax = Y
        gsl_matrix *A = gsl_matrix_alloc(n, n);
        gsl_vector *Y = gsl_vector_alloc(n);

        // put the input data into matrix A and vector Y
        fillSystem(A, Y, n);

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

        gsl_matrix_set(A, 9, 1, 1);
        std::cout << "Changed input matrix cell (10, 2) to 1.\n\n";
        output << "Changed input matrix cell (10, 2) to 1.\n\n";
        solve(A, Y, output);

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

        std::cout << "\n================================\n" << std::endl;
    }
    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 &);
void fillSystem(gsl_matrix* A, gsl_vector *Y, int n);

#endif //PROJECT_FUNCTIONS_H
```

A.3 functions.cpp

```
#include "functions.h"

#include <cmath>
#include <iomanip>
#include <iostream>
#include <gsl/gsl_linalg.h>

const int PRINT_WIDTH = 11;
const int PRINT_PRECISION = 3;

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(PRINT_WIDTH) << std::setprecision(PRINT_PRECISION) << 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(PRINT_WIDTH) << std::setprecision(PRINT_PRECISION) << 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(PRINT_WIDTH) << std::setprecision(PRINT_PRECISION) << gsl_vector_get(v, i) <<
    "\n";
        out << std::setw(PRINT_WIDTH) << std::setprecision(PRINT_PRECISION) << 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(PRINT_WIDTH) << std::setprecision(PRINT_PRECISION) << 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(PRINT_WIDTH) << std::setprecision(PRINT_PRECISION) << 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(PRINT_WIDTH) << std::setprecision(PRINT_PRECISION) << gsl_matrix_get(m, i, j);
            out << std::setw(PRINT_WIDTH) << std::setprecision(PRINT_PRECISION) << 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 = A->size1;
    size_t 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";
```

```cpp
    // 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);
}


// 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);
}

// formula: sum_{j=1}^{n} ((1+i)^(j-1)) * x_j  =  ((1+i)^n - 1) / i
double formulaRight(double i, int n)
{
    return (pow(1+i, n) - 1) / i;
}


double checkValuesInFormula(double i, int n)
{
    double leftSide = 0, rightSide = 0;
    //i *= drand48()*100;

    for (int j = 1; j <= n; j++) {
        leftSide += formulaLeft(i, j);
    }
    rightSide = formulaRight(i, n);

    std::cout << "i = "          << std::setw(5) << std::setprecision(3) << i << ", n = " << n
              << ", left side:  " << std::setw(10) << std::setprecision(5) << leftSide
              << ", right side: " << std::setw(10) << std::setprecision(5) << rightSide << std::endl;
}


void fillSystem(gsl_matrix* A, gsl_vector *Y, int n)
{
    for (size_t row = 0; row < n; ++row) {
        for (size_t col = 0; col < n; ++col) {
            gsl_matrix_set(A, row, col, formulaLeft(row+1, col+1));
        }
        gsl_vector_set(Y, row, formulaRight(row+1, n));
        //checkValuesInFormula(row+1, n);
    }
}
```

B Output

B.1 output_n_10.txt

```
Matrix Input Matrix A:
         1          2          4          8         16         32         64        128        256        512
         1          3          9         27         81        243        729   2.19e+03   6.56e+03   1.97e+04
         1          4         16         64        256   1.02e+03    4.1e+03   1.64e+04   6.55e+04   2.62e+05
         1          5         25        125        625   3.12e+03   1.56e+04   7.81e+04   3.91e+05   1.95e+06
         1          6         36        216    1.3e+03   7.78e+03   4.67e+04    2.8e+05   1.68e+06   1.01e+07
         1          7         49        343    2.4e+03   1.68e+04   1.18e+05   8.24e+05   5.76e+06   4.04e+07
         1          8         64        512    4.1e+03   3.28e+04   2.62e+05    2.1e+06   1.68e+07   1.34e+08
         1          9         81        729   6.56e+03    5.9e+04   5.31e+05   4.78e+06    4.3e+07   3.87e+08
         1         10        100      1e+03      1e+04      1e+05      1e+06      1e+07      1e+08      1e+09
         1         11        121   1.33e+03   1.46e+04   1.61e+05   1.77e+06   1.95e+07   2.14e+08   2.36e+09

Vector Input Vector Y:
  1.02e+03
  2.95e+04
   3.5e+05
```

2.44e+06
1.21e+07
4.71e+07
1.53e+08
4.36e+08
1.11e+09
2.59e+09

Matrix LU, result of LU decomposition:

1	2	4	8	16	32	64	128	256	512
1	9	117	1.32e+03	1.46e+04	1.61e+05	1.77e+06	1.95e+07	2.14e+08	2.36e+09
1	0.444	−20	−380	−5.22e+03	−6.38e+04	−7.41e+05	−8.38e+06	−9.36e+07	−1.04e+09
1	0.778	0.7	−42	−1.18e+03	−2.15e+04	−3.28e+05	−4.51e+06	−5.82e+07	−7.2e+08
1	0.111	0.4	−0.571	−144	−4.46e+03	−8.73e+04	−1.39e+06	−1.96e+07	−2.58e+08
1	0.889	0.4	0.762	0.222	−224	−9.18e+03	−2.28e+05	−4.43e+06	−7.48e+07
1	0.222	0.7	−0.667	0.972	0.625	840	3.78e+04	1e+06	2.06e+07
1	0.667	0.9	0.857	−0.25	−0.804	−0.429	−1.44e+03	−7.63e+04	−2.33e+06
1	0.333	0.9	−0.429	0.5	0.643	0.857	−0.5	−2.16e+03	−1.25e+05
1	0.556	1	0.476	−0.278	−0.714	−0.571	1	−0.667	2.88e+03

Vector X, result of solving with LU decomposition:

1
1
1
1
1
1
1
1
1
1

Vector r, residual of solving by LU decomposition:

4.33e−05
−8.11e−05
6.41e−05
−2.82e−05
7.63e−06
−1.33e−06
1.49e−07
−1.04e−08
4.09e−10
−6.97e−12

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

2.61e+09
1.23e+07
1.43e+05
3.22e+03
131
9.82
1.28
0.131
0.00598
9.7e−05

Calculating condition number by: abs(max(singular values)) / abs(min(singular values)):

Condition number = 2.61e+09 / 9.7e−05 = 2.69e+13

Changed input matrix cell (10, 2) to 1.

Matrix Input Matrix A:

1	2	4	8	16	32	64	128	256	512
1	3	9	27	81	243	729	2.19e+03	6.56e+03	1.97e+04
1	4	16	64	256	1.02e+03	4.1e+03	1.64e+04	6.55e+04	2.62e+05
1	5	25	125	625	3.12e+03	1.56e+04	7.81e+04	3.91e+05	1.95e+06
1	6	36	216	1.3e+03	7.78e+03	4.67e+04	2.8e+05	1.68e+06	1.01e+07
1	7	49	343	2.4e+03	1.68e+04	1.18e+05	8.24e+05	5.76e+06	4.04e+07
1	8	64	512	4.1e+03	3.28e+04	2.62e+05	2.1e+06	1.68e+07	1.34e+08
1	9	81	729	6.56e+03	5.9e+04	5.31e+05	4.78e+06	4.3e+07	3.87e+08
1	10	100	1e+03	1e+04	1e+05	1e+06	1e+07	1e+08	1e+09
1	1	121	1.33e+03	1.46e+04	1.61e+05	1.77e+06	1.95e+07	2.14e+08	2.36e+09

Vector Input Vector Y:

1.02e+03
2.95e+04
3.5e+05
2.44e+06
1.21e+07
4.71e+07
1.53e+08
4.36e+08
1.11e+09
2.59e+09

Matrix LU, result of LU decomposition:

1	2	4	8	16	32	64	128	256	512
1	8	96	992	9.98e+03	1e+05	1e+06	1e+07	1e+08	1e+09
1	−0.125	129	1.45e+03	1.59e+04	1.74e+05	1.9e+06	2.07e+07	2.27e+08	2.48e+09
1	0.625	−0.116	−117	−2.01e+03	−2.55e+04	−2.87e+05	−3.02e+06	−3.04e+07	−2.96e+08
1	0.25	−0.093	0.492	208	4.69e+03	7.15e+04	9.28e+05	1.11e+07	1.27e+08
1	0.875	−0.0543	0.587	−0.725	−664	−2.06e+04	−4.01e+05	−6.3e+06	−8.74e+07
1	0.375	−0.116	0.743	0.977	−0.248	−722	−2.62e+04	−5.76e+05	−9.96e+06
1	0.75	−0.093	0.903	−0.564	0.613	−0.6	1.25e+03	5.52e+04	1.43e+06
1	0.125	−0.0543	0.227	0.643	0.147	−0.934	−0.986	6.28e+03	2.98e+05
1	0.5	−0.124	0.93	0.598	−0.313	0.933	−0.513	0.206	3.74e+03

Vector X, result of solving with LU decomposition:

1.52
−0.00521
1.83
0.619
1.11
0.98
1
1
1
1

Vector r, residual of solving by LU decomposition:

2.79e−06
6.68e−07
−6.14e−06
5.61e−06
−2.43e−06
6.06e−07
−9.11e−08
8.18e−09
−4.04e−10
8.46e−12

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

2.61e+09
1.23e+07
1.43e+05
3.22e+03
131
9.87
1.21
0.196
0.0392
0.002

Calculating condition number by: abs(max(singular values)) / abs(min(singular values)):

Condition number = 2.61e+09 / 0.002 = 1.3e+12
