Міністерство освіти і науки України
Національний університет “Львівська політехніка”
Інститут комп’ютерних наук та інформаційних технологій
Кафедра програмного забезпечення

Звіт
Про виконання лабораторної роботи №6

На тему:

«Розв'язування перевизначених систем лінійних алгебраїчних рівнянь»
з дисципліни «Чисельні методи»

Лектор:
доцент каф. ПЗ
Мельник Н. Б.
Виконав:
ст. гр. ПЗ-11
Морозов О. Р.
Прийняла:
доцент каф. ПЗ
Мельник Н. Б.

« __ » _____ 2022 р.

∑ = _____ _____ .

Львів – 2022

Тема: Розв'язування перевизначених систем лінійних алгебраїчних рівнянь.
Мета: ознайомлення на практиці з методами розв’язування перевизначених систем лінійних алгебраїчних рівнянь.

Теоретичні відомості

Метод найменших квадратів для розв’язування СЛАР

Вводимо нові матриці $N = A^T \times A$; і $C = A^T \times B$, тоді початкову систему можна переписати як $N \times X = C$, далі розв'язуємо за допомогою методу квадратного кореня.

Метод квадратного кореня

Для данного методу вводимо трикутну матрицю L, таку, що $N = L \times L^T$, тоді можна записати рівняння як: $L \times L^T \times X = C$; Тоді легко, ввівши новий вектор Y, $LY = C$, і з нього випливає, що $L^T \times X = Y$. Так як матриці L і L^T - трикутні, то знаходження X та Y не буде складною задачею.

Формула L

$$\{L_{ij} = \sqrt{N_{ij} - \sum_{k=1}^{k<i} L_{ik}^2},\ i = j\ L_{ij} = 0,\ i < j\ L_{ij} = \frac{N_{ij} - \sum_{k=1}^{k<j} L_{kj} * L_{ki}}{L_{jj}},\ i > j$$

Індивідуальне завдання

Розв’язати перевизначену систему лінійних алгебраїчних рівнянь методом найменших квадратів. Отриману відповідну нормальну систему розв’язати методом квадратного кореня

x + 3y - 2z = 5
3x + 4y - 5z = 6
-2x - 5y + 3x = -13
0x + y - 2z = 4
-2x - 3y + 2z = 10

Хід роботи

Код програми:

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

#define n_ 5
#define m_ 3

void printMatrix(double** A, int n, int m);
void printMatrix(double** A, double* B, int n, int m);

void transposingmatrix(double** A, double** At, int n, int m);
```

```
double det(double** A);

int main() {

	double baseA[5][3] = {{1, 3, -2}, {3, 4, -5}, {-2, -5, 3}, {0, 1,
-2}, {-2, -3, 2}};
	double baseB[5] = { 5, 6, -13, 4, 10 };

	double** A = new double*[n_];
	double* B = new double[n_];
	for (int i = 0; i < n_; i++) {
		B[i] = baseB[i];
		A[i] = new double[m_];
		for (int j = 0; j < m_; j++) {
			A[i][j] = baseA[i][j];
		}
	}
	std::cout << "\t[ Matrix A and B ]" << std::endl;
	printMatrix(A, B, n_, m_);

	//------Transposing Matrix A------------//

	double** At = new double*[m_];

	for (int i = 0; i < m_; i++) {
		At[i] = new double[n_];
	}
	transposingmatrix(A, At, n_, m_);

	//---------------End-------------------//

	std::cout << "\t[ Matrix At ]" << std::endl;
	printMatrix(At, m_, n_);

	//------------MATRIX N-----------------//


	double** N = new double*[m_];
	double* C = new double[m_];
	for (int i = 0; i < m_; i++)
		N[i] = new double[m_];

	for (int i = 0; i < m_; i++) {
		double sumb = 0;
		for (int j = 0; j < m_; j++) {
			double suma = 0;
			sumb = 0;
```

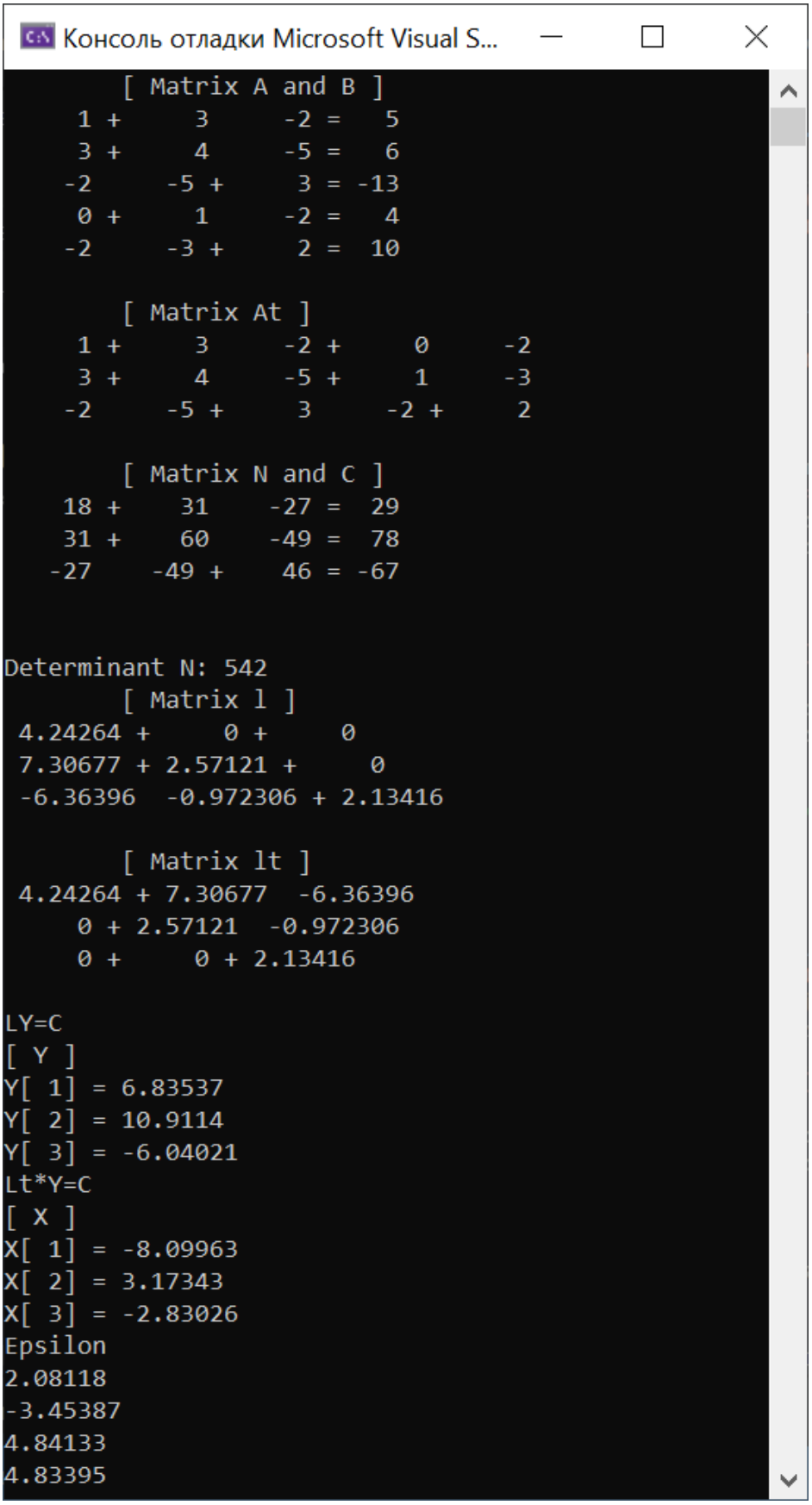
```
            for (int k = 0; k < n_; k++) {
                  suma += At[i][k] * A[k][j];
                  sumb += At[i][k] * B[k];
            }
            N[i][j] = suma;
      }
      C[i] = sumb;
}
//---------------End-------------------//

std::cout << "\t[ Matrix N and C ]" << std::endl;
printMatrix(N, C, m_, m_);
std::cout << "\nDeterminant N: " << det(N) << std::endl;

for (int i = 0; i < m_; i++) {
      delete[] At[i];
}
delete[] At;

//-------Method Square roots-----------//

double** l = new double* [m_];
double** lt = new double* [m_];
for (int i = 0; i < m_; i++) {
      l[i] = new double[m_];
      lt[i] = new double[m_];
      for (int j = 0; j < m_; j++) {
            l[i][j] = 0;
            lt[i][j] = 0;
      }
}


for (int i = 0; i < m_; i++) {
      double tmp = 0;

      for (int k = 0; k < i; k++)
            tmp += lt[k][i] * lt[k][i];

      lt[i][i] = sqrt(N[i][i] - tmp);

      for (int j = i; j < m_; j++) {
            tmp = 0;

            for (int k = 0; k < i; k++)
                  tmp += lt[k][i] * lt[k][j];
```

```
                lt[i][j] = (N[i][j] - tmp) / lt[i][i];
            }
        }

        transposingmatrix(lt, l, m_, m_);

        std::cout << "\t[ Matrix l ]" << std::endl;
        printMatrix(l, m_, m_);
        std::cout << "\t[ Matrix lt ]" << std::endl;
        printMatrix(lt, m_, m_);

        std::cout << "LY=C" << std::endl;
        double* Y = new double[m_];
        for (int i = 0; i < m_; i++)
            Y[i] = 0;

        for (int i = 0; i < m_; i++) {
            double sum = 0;
            for (int j = 0; j < i; j++) {
                sum += Y[j] * l[i][j];
            }
            Y[i] = (C[i] - sum)/l[i][i];
        }
        std::cout << "[ Y ]" << std::endl;
        for (int i = 0; i < m_; i++)
            std::cout << "Y[" << std::setw(2) << i + 1 << "] = " <<
std::setw(7) << std::setprecision(-4) << Y[i] << std::endl;

        std::cout << "Lt*Y=C" << std::endl;
        double* X = new double[m_];
        for (int i = 0; i < m_; i++)
            X[i] = 0;

        for (int i = 2; i >= 0; i--) {
            double sum = 0;
            for (int j = i + 1; j < 3; j++) {
                sum += X[j] * l[j][i];
            }
            X[i] = (Y[i] - sum) / l[i][i];
        }
        std::cout << "[ X ]" << std::endl;
        for (int i = 0; i < m_; i++)
            std::cout << "X[" << std::setw(2) << i + 1 << "] = " <<
std::setw(7) << std::setprecision(-4) << X[i] << std::endl;

        std::cout << "Epsilon" << std::endl;
```

```
	for (int i = 0; i < 5; i++) {
		double sum = 0;
		for (int j = 0; j < 3; j++) {
			sum += A[i][j] * X[j];
		}
		std::cout << sum - B[i] << std::endl;
	}
	//--------Clear Memory-------------//

	for (int i = 0; i < m_; i++) {
		delete[] N[i];
	}
	delete[] N;
	delete[] C;

	//-----------END---------------//
}

void transposingmatrix(double** A, double** At, int n, int m) {
	for (int i = 0; i < n; i++) {
		for (int j = 0; j < m; j++) {
			At[j][i] = A[i][j];
		}
	}
}

double det(double** A) {
		return (A[0][0] * A[1][1] * A[2][2] + A[0][1] * A[2][0] *
A[1][2] + A[1][0] * A[2][1] * A[0][2] - (A[0][2] * A[1][1] * A[2][0] +
A[0][1] * A[1][0] * A[2][2] + A[0][0] * A[1][2] * A[2][1]));
}

void printMatrix(double** A, double* B, int n, int m) {
	for (int i = 0; i < n; i++) {
		for (int j = 0; j <= m; j++) {

			char znak = '\0';

			if (j == m) {
				znak = '=';
				std::cout << znak << " ";
				std::cout << std::setw(3) << B[i];
			}
			else {
				if (j > 0) {
					((A[i][j]) >= 0) ? znak = '+' : znak = '\0';
				}
```

```
                        std::cout << znak << " ";
                        std::cout << std::setw(5) << A[i][j] << " ";
                  }
            }
            std::cout << "\n";
      }
      std::cout << std::endl;
}

void printMatrix(double** A, int n, int m) {
            for (int i = 0; i < n; i++) {
                  for (int j = 0; j < m; j++) {
                        char znak = '\0';
                        if (j > 0) {
                              ((A[i][j]) >= 0) ? znak = '+' : znak = '\0';
                        }
                        std::cout << znak << " ";
                        std::cout << std::setw(5) << A[i][j] << " ";
                  }
                  std::cout << "\n";
            }
            std::cout << std::endl;
}
```

Результат:


Консоль отладки Microsoft Visual S...
[Matrix A and B]
1 + 3 -2 = 5
3 + 4 -5 = 6
-2 -5 + 3 = -13
0 + 1 -2 = 4
-2 -3 + 2 = 10
[Matrix At]
1 + 3 -2 + 0 -2
3 + 4 -5 + 1 -3
-2 -5 + 3 -2 + 2
[Matrix N and C]
18 + 31 -27 = 29
31 + 60 -49 = 78
-27 -49 + 46 = -67
Determinant N: 542
[Matrix l]
4.24264 + 0 + 0
7.30677 + 2.57121 + 0
-6.36396 -0.972306 + 2.13416
[Matrix lt]
4.24264 + 7.30677 -6.36396
0 + 2.57121 -0.972306
0 + 0 + 2.13416
LY=C
[Y]
Y[1] = 6.83537
Y[2] = 10.9114
Y[3] = -6.04021
Lt*Y=C
[X]
X[1] = -8.09963
X[2] = 3.17343
X[3] = -2.83026
Epsilon
2.08118
-3.45387
4.84133
4.83395

Висновок

Виконуючи цю лабораторну роботу я вивчив та ознайомився на практиці з методами перевизначення систем лінійних рівнянь, та ознайомився з таким методом рішення СЛАР, як метод квадратного кореня.