Звіт з предмету Чисельні методи
Пустовіта Дмитра
КА-32

1. Умова Задачі (Варіант 16)

Розв’язати матричне рівняння (систему лінійних алгебраїчних рівнянь)

Ax=b, де $A = \begin{pmatrix} 6.92 & 1.2 & 0.87 & 1.15 & -0.66 \\ 1 & 3.5 & 1.3 & -16.3 & 0.420 \\ 1.07 & -2.46 & 6.1 & 2.1 & 0.883 \\ 1.33 & 0.16 & 2.1 & 5.44 & -10 \\ 1.14 & -1.08 & -0.617 & 5 & 1 \end{pmatrix}$, та $b = \begin{Bmatrix} 2.1 \\ 0.72 \\ 2.15 \\ 10.12 \\ -1.08 \end{Bmatrix}$

Методом Гауса.

2. Математичне розв’язання задачі

Метод Гауса полягає у послідовному виключенні змінних з рівнянь шляхом еквівалентних рядкових перетворень, таким чином , аби звести вихідну систему до зручнішого вигляду.

На етапі прямого ходу система зводиться до трикутного (в такому разі $i = \overline{k+1, n}$, де i – номер рівняння, з якого віднімається k-те) або ж відразу до діагонального, в цьому разі $i = \overline{1, n},\ i \neq k$. В програмі реалізовано обидва методи.

Формули прямого ходу наступні:

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - L_{ik} \cdot a_{ij}^{(k-1)}$$, при цьому $$L_{ik}^{(k-1)} = \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}$$

Для вектора b в свою чергу маємо

$$b_j^{(k)} = b_j^{(k-1)} - L_{ik}^{(k-1)} b_k^{(k-1)}$$.

Тут $k = \overline{1, n-1}$ - номер рівняння яке віднімається від інших.

$j = \overline{k, n}$ - номер стовпця.

Складність прямого ходу складає $\theta = \frac{n(n+1)(n+2)}{3}$

Формули для оберненого ходу такі:

$$x_n = \frac{b_n^{(n)}}{U_{nn}} \qquad x_i = \frac{b_i^{(n)} - \sum_{k=i+1}^{n} U_{ik} x_k}{U_{ii}}$$, де U – отримана трикутна матриця.

Складність оберненого ходу складає $\theta = \frac{n(n+1)}{2}$.

При цьому метод Гауса передбачає що на кожному кроці ведучі елементи (a_{ii}) не рівні нулю. Тому на кожному кроці прийнято шукати далі рівняння для якого модуль і-го коефіцієнта найбільший, та переставляти рівняння.

Після зведення матриці до діагонального вигляду можна обчислити визначник у вигляді:

$detA = (-1)^{z\ mod\ 2} \prod_{i=1}^{n} a_{ii}$, де z – кількість перестановок рівнянь.

При розв'язанні вручну було отримано такі результати:

$$x = \begin{Bmatrix} 2.1 \\ 0.72 \\ 2.15 \\ 10.12 \\ -1.08 \end{Bmatrix}$$

$$detA = \frac{3922828754794413}{500000000000} \approx 7845.65751$$

$$A^{-1} = \begin{matrix} \frac{650619426462400}{3922828754794413} & \frac{-94472715024000}{1307609584931471} & \frac{-23067970072000}{3922828754794413} & \frac{-24084947568500}{1307609584931471} & \frac{-153734967086000}{3922828754794413} \\ \frac{-226072015607450}{3922828754794413} & \frac{393550354560050}{1307609584931471} & \frac{-206804001760000}{3922828754794413} & \frac{29648254406600}{1307609584931471} & \frac{426974588705500}{3922828754794413} \\ \frac{-127890113054000}{3922828754794413} & \frac{127243851058000}{1307609584931471} & \frac{545671993160000}{3922828754794413} & \frac{19577924246000}{1307609584931471} & \frac{-139225369529000}{3922828754794413} \\ \frac{-202169991072500}{3922828754794413} & \frac{106563779014800}{1307609584931471} & \frac{5666408594000}{3922828754794413} & \frac{36243696718000}{1307609584931471} & \frac{814604907085000}{3922828754794413} \\ \frac{-17974055805000}{1307609584931471} & \frac{78423839081000}{1307609584931471} & \frac{37098913597000}{1307609584931471} & \frac{-109661949343000}{1307609584931471} & \frac{133430861550000}{1307609584931471} \end{matrix}$$

Результати роботи (прямий хід методу Гауса)

```
iteration #1
6.920000      1.200000      0.870000      1.150000      -0.660000     2.100000
0.000000      3.326590      1.174277      -1.796185     0.515376      0.416532
1.070000      -2.460000     6.100000      2.100000      0.883000      2.150000
1.330000      0.160000      2.100000      5.440000      -10.000000    10.120000
1.140000      -1.080000     -0.617000     5.000000      1.000000      -1.080000
iteration #2
6.920000      1.200000      0.870000      1.150000      -0.660000     2.100000
0.000000      3.326590      1.174277      -1.796185     0.515376      0.416532
0.000000      -2.645549     5.965477      1.922182      0.985052      1.825289
1.330000      0.160000      2.100000      5.440000      -10.000000    10.120000
1.140000      -1.080000     -0.617000     5.000000      1.000000      -1.080000
iteration #3
6.920000      1.200000      0.870000      1.150000      -0.660000     2.100000
```

```
0.000000      3.326590      1.174277      -1.796185     0.515376      0.416532
0.000000      -2.645549     5.965477      1.922182      0.985052      1.825289
0.000000      -0.070636     1.932789      5.218974      -9.873150     9.716387
1.140000      -1.080000     -0.617000     5.000000      1.000000      -1.080000

                                          ⋮
iteration #9
6.920000      0.000000      0.000000      1.765992      -0.936166     1.810211
0.000000      3.326590      1.174277      -1.796185     0.515376      0.416532
0.000000      0.000000      6.899349      0.493723      1.394917      2.156546
0.000000      0.000000      1.957723      5.180834      -9.862207     9.725232
0.000000      0.000000      -0.309303     4.120664      1.306676      -1.265971
iteration #10
6.920000      0.000000      0.000000      1.765992      -0.936166     1.810211
0.000000      3.326590      0.000000      -1.880217     0.277959      0.049485
0.000000      0.000000      6.899349      0.493723      1.394917      2.156546
0.000000      0.000000      1.957723      5.180834      -9.862207     9.725232
0.000000      0.000000      -0.309303     4.120664      1.306676      -1.265971
                                          ⋮
iteration #19
6.920000      0.000000      0.000000      0.000000      0.000000      0.965735
0.000000      3.326590      0.000000      0.000000      0.000000      0.313489
0.000000      0.000000      6.899349      -0.000000     0.000000      3.384261
0.000000      0.000000      0.000000      5.040738      -10.258021    9.113302
0.000000      0.000000      0.000000      0.000000      9.799904      -8.659181
iteration #20
6.920000      0.000000      0.000000      0.000000      0.000000      0.965735
0.000000      3.326590      0.000000      0.000000      0.000000      0.313489
0.000000      0.000000      6.899349      -0.000000     0.000000      3.384261
0.000000      0.000000      0.000000      5.040738      -0.000000     0.049329
0.000000      0.000000      0.000000      0.000000      9.799904      -8.659181
```

(вихідний файл)

det A = 7845.66
The root is:
0.139557
0.0942374
0.490519
0.00978602
-0.883599
Обернена матриця:

```
0.165855      -0.072248     -0.005880     -0.018419     -0.039190
-0.057630     0.300969      -0.052718     0.022674      0.108844
-0.032602     0.097310      0.139102      0.014972      -0.035491
-0.051537     0.081495      0.001444      0.027718      0.207658
-0.013746     0.059975      0.028372      -0.083864     0.102042
```

A*(A)^-1 =

```
1.000000e+000 3.469447e-017 6.938894e-018 -1.387779e-017 0.000000e+000
-1.561251e-017 1.000000e+000 6.765422e-017 -8.326673e-017 -2.081668e-017
4.857226e-017 6.245005e-017 1.000000e+000 6.938894e-017 -2.775558e-017
-2.775558e-017 -1.110223e-016 0.000000e+000 1.000000e+000 -2.220446e-016
0.000000e+000 2.081668e-017 -3.469447e-018 -4.163336e-017 1.000000e+000
```

вектор нев'язки:

```
0.000000e+000
-9.992007e-016
4.440892e-016
3.552714e-015
-8.881784e-016
```

4.Код програми

```
// Gauss.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include "fstream"
#include "iostream"
#define n 5
#define m 5
#define eps 0.00001
#define zn 6
typedef double Mat[n][n];
typedef double vec[n];
typedef double Sys[n][m];
using namespace std;

void error(int);
void fileMat(ofstream &);
void inMat(Mat*);
void inVec(vec*);
void inSys(Sys*);
void randMatr(Mat*, vec* );
void outMat(ofstream &, Mat* , vec* );
 int maxLine(Mat*, int);
void prGaus(Mat*, vec*, int, int);
void outVec(ofstream &, vec*);
void obGaus(Mat*, vec*, vec*);
void diag(Mat*, vec*, vec*, int);
void GausMeth(Mat*, vec*, vec*, int);
void outSys(ofstream &, Sys *);
void GausSys(Mat* , Sys* );
void outProd(ofstream &, Mat*, Sys*);
void product(Mat*, vec*, vec*);
void check(Mat*, Sys*, vec*, vec*);

int _tmain(int argc, _TCHAR* argv[])
{
	Mat A,M;
	Sys B;
	vec b,x;
	//randMatr(&A,&b);
	inMat(&A);
	inVec(&b);
	GausMeth(&A, &b, &x,0);

	inMat(&M);
	inSys(&B);
	GausSys(&M,&B);

	ofstream fout("out.txt",ios_base::app);
	fileMat(fout);
	fout << "Обернена матриця: " << endl;
	outSys(fout, &B);
	fout.close();

	inMat(&M);
	inVec(&b);
	check(&M,&B,&b,&x);
	system("Pause");
	return 0;
}

void error(int k){
	ofstream stm("output.txt", ios_base::app);
	switch (k)
	{
	case 1: stm << "Матриця Вироджена";
		break;
	case 2: "";
		break;
	}
	exit(k);
	stm.close();
};

void fileMat(ofstream &fout){
	fout.width(zn + 6);
	fout.precision(zn);
	fout.setf(ios::right);
	fout.setf(ios::fixed);
}

void inMat(Mat* A){
	ifstream stm("inMat.txt");
	int i, j;
	for (i = 0; i < n; i++)
		for (j = 0; j < n; j++)
			stm >> (*A)[i][j];
	stm.close();
};

void inVec(vec* b){
	ifstream stm("inVec.txt");
	int i;
	for (i = 0; i < n; i++)
		stm >> (*b)[i];
	stm.close();
};

void inSys(Sys* A){
	ifstream stm("inSys.txt");
	int i, j;
	for (i = 0; i < n; i++)
		for (j = 0; j < m; j++)
			stm >> (*A)[i][j];
	stm.close();
};

void randMatr(Mat*  A, vec* b){
	int i, j;
	for (i = 0; i < n; i++){
		for (j = 0; j < n; j++)
			(*A)[i][j] = 1 + rand() % 10;
		(*b)[i] = rand() % 10;
	}
};

void outMat(ofstream &stm, Mat* A, vec* b){
```

```
        int i, j;
        for (i = 0; i < n; i++){
                for (j = 0; j < n; j++)
                        stm << (*A)[i][j] << "\t";
                stm << (*b)[i] << endl;
        }
};

void outVec(ofstream &stm, vec* x){
        int i;
        for (i = 0; i < n; i++)
                stm << (*x)[i] << endl;
}

int maxLine(Mat* A,int i){
        double max = fabs((*A)[i][i]);
        int j, ind=i;
        for (j=ind+1; j < n; j++)
                if ( fabs((*A)[j][i]) > max){
                max = fabs((*A)[j][i]);
                ind = j;
        }
        if (max < eps) error(1);
        return ind;
};

void prGaus(Mat*  A, vec* b, int f2, int znack){
        ofstream stm("priamyj_hid.txt");
        fileMat(stm);
        int k, i, j, p2 = 0, t = 0, ind;
        double L,hold;
        for (k = 0; k < n; k++){
                if (ind=maxLine(A, k) != k){
                                znack++;
                                for (j = 0; j < n;
j++){
                                        hold =
(*A)[k][j];
                                        (*A)[k][j] =
(*A)[ind][j];
                                        (*A)[ind][j] =
hold;
                                }
                                hold = (*b)[i];
                                (*b)[k] = (*b)[ind];
                                (*b)[ind] = hold;
                        }
                if (!f2) p2 = k + 1;
                for (i = p2; i < n; i++){
                        if (f2 && (i == k)) ++i;
                        if (i == n) break;
                        if (f2 && (i == k)) ++i;
                        if (i == n) break;

                        L = (*A)[i][k] / (*A)[k][k];
                        for (j = 0; j < n; j++)
                                (*A)[i][j] =
(*A)[i][j] - L*(*A)[k][j];
                        (*b)[i] = (*b)[i] - L*(*b)[k];
                        stm << "iteration #" << ++t <<
endl;
                        outMat(stm, A, b);
                }
        }
        stm.close();
};

void obGaus(Mat*  A, vec* b, vec* x){
        (*x)[n - 1] = (*b)[n - 1] / (*A)[n - 1][n -
1];
        int i, k;
        double U;
        for (i = n - 2; i >= 0; i--){
                U = 0;
                for (k = i + 1; k < n; k++)
                        U = U + (*A)[i][k] * (*x)[k];
                (*x)[i] = ((*b)[i] - U) / (*A)[i][i];
        }
        ofstream fout("out.txt");
        fout << "The root is:" << endl;
        outVec(fout, x);
        fout.close();
};

void diag(Mat*  A, vec* b, vec* x,int znack){
        int i;
        double D = 1;
        for (i = 0; i < n; i++){
                (*x)[i] = (*b)[i] / (*A)[i][i];
                D = D*(*A)[i][i];
        }
        D = D*pow(-1, znack % 2);
        ofstream fout("out.txt");
        fout << "det A = " << D << endl;
        fout << "The root is:" << endl;
        outVec(fout, x);
        fout.close();
}

void GausMeth(Mat*  A, vec* b, vec* x, int fl2){
        int z=0;
        prGaus(A, b, !fl2, z);
        if (fl2) obGaus(A, b, x);
        else diag(A, b, x, z);
};

void outSys(ofstream &stm, Sys *B){
        int i, j;
        for (i = 0; i < n; i++){
                for (j = 0; j < m; j++)
                        stm << (*B)[i][j] << " \t";
                stm << endl;
        }
};

void GausSys(Mat* A, Sys* B){
        int k, i, j, t = 0;
        double L;
        ofstream fout("SystemSol.txt");
        fileMat(fout);

        for (k = 0; k < n; k++){
                for (i = 0; i < n; i++){
                        if (i == k) ++i;
                        if (i == n) break;
                        L = (*A)[i][k] / (*A)[k][k];
                        for (j = 0; j < n; j++)
                                (*A)[i][j] =
(*A)[i][j] - L*(*A)[k][j];
                        for (j = 0; j < m; j++)
                                (*B)[i][j] =
(*B)[i][j] - L*(*B)[k][j];
                        fout << "iteration #" << ++t
<< endl;
                        outSys(fout, B);
                }
        }
        for (i = 0; i < n; i++)
                for (j = 0; j < m; j++){
                (*B)[i][j] = (*B)[i][j] / (*A)[i][i];
                }
        fout.close();
};
```

```
void outProd(ofstream &fout, Mat*A, Sys* B){
        int i, k, j;
        double S = 0;
        for (i = 0; i < n; i++){
                for (j = 0; j < m; j++){
                        for (k = 0; k < n; k++)
                                S = S + (*A)[i][k] *
(*B)[k][j];
                        fout << S << "\t";
                        S = 0;
                }
                fout << endl;
        }
}

void product(Mat*  A, vec* b, vec* Ax){
        int i, j;
        double S = 0;
        for (i = 0; i < n; i++){
                for (j = 0; j < n; j++)
                        S = S + (*A)[i][j] * (*b)[j];
                (*Ax)[i] = S;
                S = 0;
        }
}

void check(Mat* A, Sys* B, vec* b, vec* x){
        ofstream fout("out.txt", ios_base::app);
        fout.setf(ios::scientific);
        fout << "A*(A)^-1 = " << endl;
        outProd(fout, A, B);
        fout << "вектор нев'язки: " << endl;
        vec Ax;
        product(A, x, &Ax);
        for (int i = 0; i < n; i++)
                Ax[i] = Ax[i] - (*b)[i];
        outVec(fout, &Ax);
}
```

3. **Висновок**

Метод Гауса має велике число модифікацій, тому для різних цілей зручно використовувати певну модифікацію. В даній програмі реалізований «класичний» метод Гауса з вибором ведучого елементу для розв’язання рівняння, при цьому бічним результатом роботи методу стало обчислення визначника (з урахуванням числа перестановок рядків), також було обчислено обернену матрицю, добуток $A \times A^{-1}$, та вектор нев’язки $Ax - b$. Було обчислено похибки порядку 10^{-15}, що цілком задовольняє надану точність.