Лабораторная работа № 1
Тема: «Прямые методы решения СЛАУ»

Курец Любовь (2 группа)

Постановка задачи №1

Написать программу, которая решает систему линейных алгебраических уравнений с матрицей A порядка n методом Гаусса с выбором главного элемента по столбцу, а также вычисляет определитель матрицы $\det A$, обратную матрицу A^{-1}. Предусмотреть сообщения, предупреждающие о невозможности решения указанной задачи с заданной матрицей A.

Для проведения вычислительного эксперимента необходимо решить систему размерности $n = 10$. Матрицу A и вектор точного решения x заполнить случайными числами (сгенерировать) с двумя знаками после запятой из диапазона от -10 до 10. Правую часть задать умножением матрицы A на вектор x

В результатах выполнения тестовой задачи необходимо привести следующую информацию:

- Условие: матрица A (построчно), вектор f, точное решение x.
- Полученное решение
- Максимум-норма невязки
- Максимум-норма погрешности

Алгоритм решения

Решение системы линейных алгебраических уравнений будет найдено методом Гаусса с выбором главного элемента по столбцу. Метод Гаусса содержит две совокупности операций, условно названных прямой ход и обратный ход.

Прямой ход:

$a_{ij}^{(0)} = a_{ij}$, $i = 1,2,\dots,n$, $j = 1,2,\dots,n$;

$b_i^{(0)} = b_i$, $i = 1,2,\dots,n$;

$k = 1, 2, \dots, n-1$:

$i = k+1, k+2, \dots, n$:

$c_i = a_{ik}^{(k-1)}$ // *выделяется k-й столбец подматрицы*

Выбор наибольшего по модулю элемента в массиве c,

перестановка соответствующих строк матрицы A,

перенумерация индекса i элементов $a_{ij}^{(k-1)}$, $b_i^{(k-1)}$

// *ведущим элементом становится элемент* c_k

$i = k+1, k+2, \ldots, n$:

$l_i = \frac{c_i}{c_k}$, // *создается временный коэффициент* $l_i = \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}$ *путем деления i-го элемента k-го столбца матрицы на ведущий элемент*

$b_i^{(k)} = b_i^{(k-1)} - l_i b_k^{(k-1)}$,

$j = k+1, k+2, \ldots, n$:

$a_{ij}^{(k)} = a_{ij}^{(k-1)} - l_i a_{kj}^{(k-1)}$.

Обратный ход заключается в получении значений неизвестных из треугольной системы: из последнего уравнения получаем значение неизвестного x_n; подставляя его в предпоследнее уравнение, находим x_{n-1}; подставляя оба этих неизвестных в третье с конца уравнение, находим x_{n-2} и так далее. Вычисления осуществляются по формулам

$$x_n = \frac{b_n^{(n-1)}}{a_{n\,n}^{(n-1)}}, \quad x_i = \frac{1}{a_{ii}^{(i-1)}}\left(b_i^{(i-1)} - \sum_{j=i+1}^{n} a_{ij}^{(i-1)} x_j\right), \; i=n-1, n-2, \ldots, 1.$$

Нахождение обратной эквивалентно решению уравнения: $AX=E \Leftrightarrow X=A^{-1}$

Вычисление определителя

Так как мы сводим матрицу A к треугольному виду при помощи элементарных

преобразований, то:

$$\det A = (-1)^m a_{11} \cdot a_{22} \cdot \ldots \cdot a_{nn}$$

где m - кол-во перестановок строк/столбцов

a_{ii} - главные элементы метода Гаусса

Листинг программы

```
//Вычисление определителя матрицы
float Det(float** Matrix, int size)
{
	float det = 1;
	for (int i = 0; i < size; i++)
	{
		det = det * Matrix[i][i];

	}
	if (count_numb % 2 != 0)
	{
		det = det * (-1);
	}
	return det;
}
// обратный ход
float* find_roots(float** Matrix, float* Answ, int size)
{
	float* X = new float[size];
	for (int row = size - 1; row >= 0; row--)
```

```
	{
		for (int column = size - 1; column >= row; column--)
		{
			if (row == column)
			{
				try {
					if (Matrix[row][column] == 0)
					{
						throw(-1);
					}
				}
				catch (int s)
				{
					cout << "нет ответа";
					exit(-1);
				}

				X[column] = Answ[row] / Matrix[row][column]; //xn=bn/cnn
			}
			else
			{
				Answ[row] = Answ[row] - X[column] * Matrix[row][column];
			}
		}
	}
	return X;
 }

//прямой ход

	bool temp;
	float div;
	for (int k = 0; k < size; k++) // матрица A с помощью элементарных преобразований
приводится к нижне треугольной
	{
		//выбор главного по столбцу, переставляем строки марицы так,
		//чтобы наиб по мод элемент при xk попал на глав диагональ
		//потом выбираем его как глав элемент
		for (int row = k; row < size; row++)
		{
			if (abs(Matrix[row][k]) > abs(Matrix[k][k]))
			{
				swap(Matrix[row], Matrix[k]);
				swap(Answ[row], Answ[k]);
				count_numb++;
			}
		}
		/*Далее исключим переменную из всех уравнений, начиная с
		(k + 1).Для этого вычтем получившуюся после перестановки k
			строку из остальных строк, домножив её на величину, равную
			отношению  элемента каждой из этих строк к k элементу
			первой строки, обнуляя тем самым столбец под ним.*/
		for (int row = k + 1; row < size; row++)
		{
			div = Matrix[row][k];
			temp = true;
			for (int col = k; col < size; col++)
			{
				if (Matrix[k][k] == 0)
				{
					temp = false;
					continue;
				}
				Matrix[row][col] = Matrix[row][col] - Matrix[k][col] * div /
Matrix[k][k];
```

```
                        }
                        if (temp == true)
                        {
                                Answ[row] = Answ[row] - Answ[k] * div / Matrix[k][k];
                        }
                }
        }

//вычисление обратной матрицы

        for (int i = 0; i < size; i++)
        {
                float* En = Matrix_en[i];
                for (int row = 0; row < size; row++)
                {
                        for (int col = 0; col < size; col++)
                        {
                                Matrix[row][col] = Matrix_temp[row][col];
                        }
                }

                for (int k = 0; k < size; k++)                  //приводим к ниже
треугольной
                {

                        for (int row = k; row < size; row++)
                        {
                                if (abs(Matrix[row][k]) > abs(Matrix[k][k]))
                                {
                                        swap(Matrix[row], Matrix[k]);
                                        swap(En[row], En[k]);
                                }
                        }
                        for (int row = k + 1; row < size; row++)
                        {
                                div = Matrix[row][k];
                                temp = true;
                                for (int col = k; col < size; col++)
                                {
                                        if (Matrix[k][k] == 0)
                                        {
                                                temp = false;
                                                continue;
                                        }
                                        Matrix[row][col] = Matrix[row][col] - Matrix[k][col] *
div / Matrix[k][k];
                                }
                                if (temp == true)
                                {
                                        En[row] = En[row] - En[k] * div / Matrix[k][k];
                                }
                        }
                }
                float* Temp;
                Temp = find_roots(Matrix, En, size);

                for (int j = 0; j < size; j++)
                {
                        Matrix_reverse[j][i] = Temp[j];
                }


        }

        // A*A^(-1)
```

```
	for (int row = 0; row < size; row++)
	{
		for (int col = 0; col < size; col++)
		{
			Matrix[row][col] = 0;
		}
	}

	for (int A_row = 0; A_row < size; A_row++)
	{
		for (int col = 0; col < size; col++)
		{
			for (int row = 0; row < size; row++)
			{
				Matrix[A_row][col] = Matrix[A_row][col] +
Matrix_temp[A_row][row] * Matrix_reverse[row][col];
			}
		}
	}



for (int row = 0; row < size; row++)
	{
		for (int col = 0; col < size; col++)
		{
			x = x + Matrix_temp[row][col] * X[col];
		}
		v.push_back(Answ_temp[row] - x);
		x = 0;
	}

	sort(v.begin(), v.end());
	cout << "Максимум-норма невязки" << endl;
	cout << abs(v[v.size() - 1]) << endl;
	v.clear();

	for (int i = 0; i < size; i++)
	{
		v.push_back(X[i] - Roots[i]);
	}
	sort(v.begin(), v.end());
	cout << "Максимум-норма погрешности" << endl;
	cout << abs(v[v.size() - 1]);
```

Результаты:

```
Размер матрицы: 10
A|f
     8.77      4.03      2.22     -2.57    -10.08      1.23     -5.43      -5.7     -3.07      4.73|   194.564
    -6.62     10.51    -10.47      3.08    -10.88      0.03     -2.03     -3.76      3.69     -7.69|  -155.501
    -2.07     -2.14      8.28      2.79      8.13      5.93      8.47       4.3     -2.71     -0.26|  -24.0248
    -0.42      7.41     -6.26     -1.76      0.24     -5.57      3.98      0.56     10.72     -7.27|  -259.593
    -0.03      3.03     -9.81      4.51      1.43     -5.53     -0.28     -2.79     -0.79     -8.98|   -95.086
    -1.91      -0.1      2.78     -3.04    -10.72     -0.63     -5.32      4.79     -6.88     -7.15|   185.564
    -4.63      2.11     -6.65      3.56     -6.58     -4.92     -8.47     -5.19      4.07     -1.83|  -19.2924
     2.71      4.44       2.3      0.73    -10.69      7.52     -3.01      2.63      3.17     -5.68|   50.7694
    -7.36      7.62     -7.16     -3.35      4.89      0.64     -5.47     -1.85      0.38      8.72|  -142.883
    -6.59     10.22     -1.35      3.45     -1.27     -2.25     -7.81     -5.88      2.43     -5.06|  -48.6672
X:      5.76      -6.78       9.44      -0.18      -5.38      -0.24      -8.88       -3.1      -10.8      -0.64
Det: -1.15419e+10

Решения:
   5.75999      -6.78       9.44  -0.180007      -5.38  -0.239998      -8.88   -3.10001      -10.8  -0.640002
A^(-1)
   0.0743092   -0.0773018    0.0603395    0.0426877    0.0819343  -3.6925e-05    0.0558375     0.051742    0.0302563   -0.0489806
    0.139723   -0.0611835     0.225001     0.113845    0.0698117    0.0630769     0.148316    0.0116093    0.0995284    -0.059712
   0.0337789   -0.0288141      0.06095    0.0320645   -0.0427611    0.0235012    0.0353581   -0.0222678   -0.0242805    0.0392113
    0.182627    -0.123934     0.405246     0.117716     0.140556    0.0626692     0.399155    0.0468489     0.112177    -0.172517
  -0.0598485   -0.0273891   -0.0841349   -0.0342332    0.0201283   -0.0373002   -0.0992532    0.0179888  -0.00265785    0.0672951
   -0.141801    0.0923171    -0.239785    -0.144754   -0.0542042   -0.0996668    -0.250516    0.0611003   -0.0609552     0.101443
   0.0308683    0.0757099    0.0504129    0.025173   -0.0286022  -0.00424007   -0.0147861    -0.06613   -0.0278963   -0.0367058
     0.15039    -0.166919     0.365663     0.173232    0.128449     0.135591      0.35503    0.076799     0.173147    -0.209198
  -0.0436164   -0.0107127   -0.0513377    0.0221658   -0.0504619   -0.0458314   0.00738535    0.0351155   -0.0240595    0.0170653
    0.135658   -0.0619217     0.236261    0.0894961    0.0312363    0.0445577     0.239671  -0.00805069      0.10265    -0.136577
A*A^(-1)
           1 -2.98023e-08 -1.19209e-07  2.98023e-08 -7.45058e-08  4.47035e-08            0 -8.56817e-08 -2.98023e-08 -5.96046e-08
 2.38419e-07            1  3.57628e-07 -1.19209e-07   1.3411e-07  2.98023e-08            0  2.23517e-08 -1.78814e-07  1.19209e-07
  8.9407e-08  9.49949e-08            1   5.7742e-08  1.08033e-07  1.17347e-07  2.19792e-07 -5.30854e-08 -2.79397e-08  -2.5332e-07
 1.78814e-07 -1.19209e-07  5.96046e-07            1  2.98023e-08  2.98023e-08 -1.19209e-07  2.04891e-07  1.78814e-07  5.96046e-08
 3.57628e-07            0  4.76837e-07  5.96046e-08            1  1.19209e-07  2.38419e-07  1.49012e-07  1.19209e-07 -2.38419e-07
-5.96046e-08 -1.19209e-07  2.38419e-07  5.96046e-08  4.47035e-08            1  1.19209e-07            0  1.78814e-07  1.19209e-07
 2.08616e-07 -4.47035e-08   8.9407e-08  2.08616e-07  1.11759e-07 -3.72529e-08            1  6.51926e-08  1.49012e-07 -1.19209e-07
 3.57628e-07  5.96046e-08  3.57628e-07  1.19209e-07   1.3411e-07  1.19209e-07  1.19209e-07            1            0 -1.78814e-07
 1.19209e-07  1.19209e-07  4.76837e-07 -1.19209e-07  5.96046e-08  2.98023e-08 -2.38419e-07  7.45058e-09            1  1.19209e-07
 3.57628e-07  2.68221e-07  1.07288e-06            0            0 -2.23517e-07            0 -4.84288e-08 -2.98023e-07            1
Максимум-норма невязки
3.8147e-05
Максимум-норма погрешности
2.22027e-06
```

Вывод:

Метод Гаусса является универсальным прямым методом решения систем линейных алгебраических уравнений. Общая идеология прямых методов решения СЛАУ состоит в преобразовании системы к некоторому более простому виду и в последующем решении получившейся более простой системы.

Постановка задачи №1

Написать программу, которая решает систему линейных алгебраических уравнений

Ax=f для трехдиагональной матрицы A порядка n методом прогонки. Предусмотреть сообщения, предупреждающие о невозможности решения указанной задачи с заданной матрицей A .

Для проведения вычислительного эксперимента необходимо сгенерировать случайную трехдиагональную матрицу A с диагональным преобладанием размерности n =10 и вектор точного решения x . Правую часть задать умножением матрицы A на вектор x: f= Ax . Затем необходимо решить полученную систему с помощью вашей программы и занести в отчет результаты.

В результатах выполнения тестовой задачи необходимо привести следующую информацию:

- Условие (матрицу A (построчно), вектор f , точное решение x).
- Полученное решение.
- Максимум-норма невязки.
- Максимум-норма погрешности.

Алгоритм метода прогонки

$$
\begin{aligned}
c_0 y_0 - b_0 y_1 &= f_0,\\
-a_1 y_0 + c_1 y_1 - b_1 y_2 &= f_1,\\
&\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\\
-a_i y_{i-1} + c_i y_i - b_i y_{i+1} &= f_i,\\
&\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\\
-a_N y_{N-1} + c_N y_N &= f_N.
\end{aligned}
$$

• прямая прогонка – вычисление прогоночных коэффициентов по формулам

$$\alpha_1 = \frac{b_0}{c_0},\ \beta_1 = \frac{f_0}{c_0},$$

$$\alpha_{i+1} = \frac{b_i}{c_i - a_i\alpha_i},\ \beta_{i+1} = \frac{f_i + a_i\beta_i}{c_i - a_i\alpha_i},\ i = 1, 2, \dots, N-1,$$

$$\beta_{N+1} = \frac{f_N + a_N\beta_N}{c_N - a_N\alpha_N};$$

• обратная прогонка – вычисление решения по формулам

$y_N=\beta_{N+1}$, $y_i=\alpha_{i+1}y_{i+1}+\beta_{i+1}$, $i = N-1, \dots, 1, 0$.

Листинг программы

```
// распределяем по векторам
```

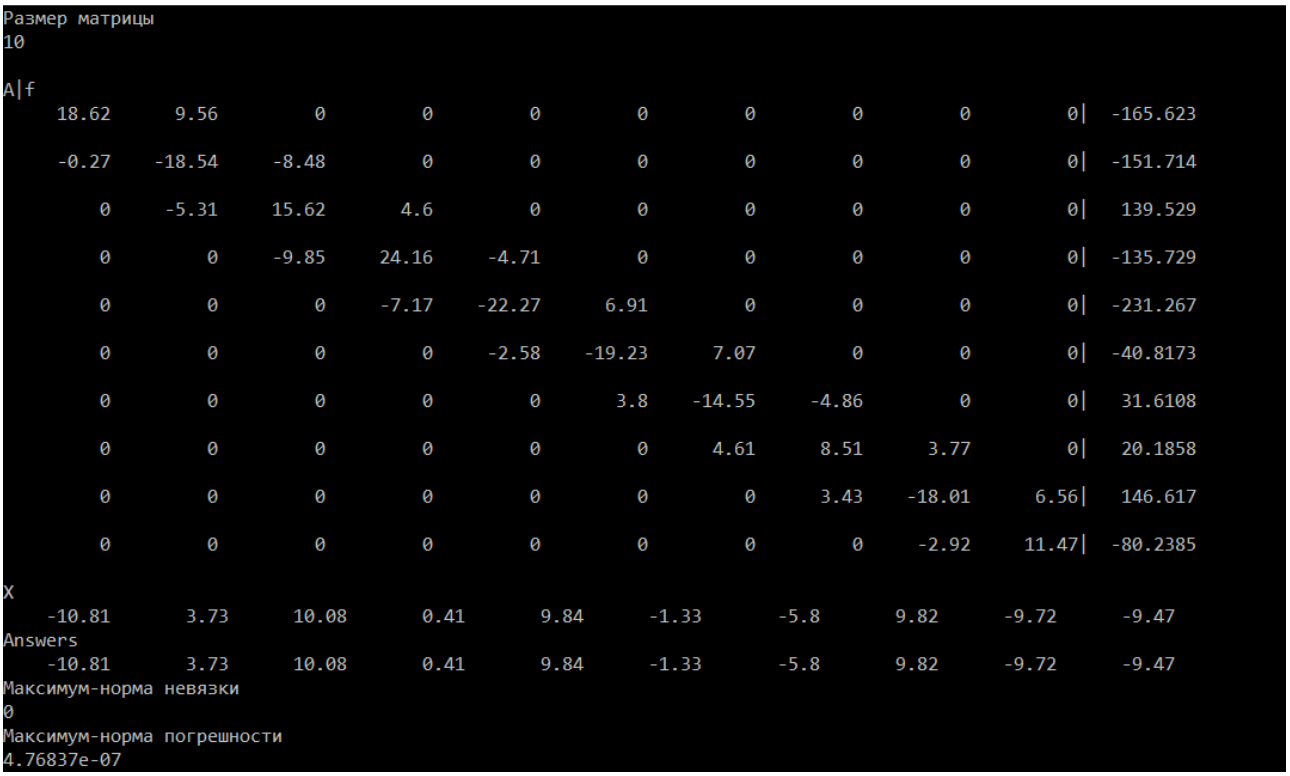
```
	for (int row = 0; row < size; row++)
	{
		for (int col = 0; col < size; col++)
		{
			if (row == col)
			{
				c[row] = Matrix[row][col];
			}
			if (row + 1 == col)
			{
				b[row] = -Matrix[row][col];
			}
			if (row - 1 == col)
			{
				a[row] = -Matrix[row][col];
			}
		}
	}
	//проверка на устойчивость и корректность

	try {
		if (abs(c[0]) == 0 || abs(c[size - 1]) == 0 || abs(b[0]) == 0 || abs(a[size
- 1]) == 0 || abs(c[0]) < abs(b[0]) || abs(c[size - 1]) < abs(a[size - 1]))
			throw(-1);
	}
	catch (int s)
	{
		cout << "нет ответа";
		exit(-1);
	}

	for (int i = 1; i < size - 1; i++)
	{
		try {
			if (abs(a[i]) == 0 || abs(b[i]) == 0 || abs(c[i]) < abs(a[i]) +
abs(b[i]))
				throw(-1);
```

```
	}
	catch (int s)
	{
		cout << "нет ответа";
		exit(-1);
	}
}

bool ind = false;
for (int i = 1; i < size; i++)
{
	if (abs(c[i]) > abs(a[i]) + abs(b[i]))
		ind = true;
}
if (abs(c[0]) > abs(b[0]) || abs(c[size - 1]) > abs(b[size - 1]))
{
	ind = true;
}

try {
	if (ind == false)
	{
		throw(-1);
	}
}
catch (int s)
{
	cout << "нет ответа";
	exit(-1);
}



bool temp;
float div;
float mul;
//вычисление по формулам
```

```
	//прямой ход
	alf[1] = b[0] / c[0];
	bet[1] = f[0] / c[0];
	for (int i = 1; i < size - 1; i++)
	{
		alf[i + 1] = b[i] / (c[i] - alf[i] * a[i]);;
		bet[i + 1] = (f[i] + a[i] * bet[i]) / (c[i] - alf[i] * a[i]);
	}
	bet[size] = (f[size - 1] + a[size - 1] * bet[size - 1]) / (c[size - 1] - alf[size
- 1] * a[size - 1]);



	float* X;


	//обратный ход
	X = roots(alf, bet, size);                    //находим корни
	cout << "Answers" << endl;
	for (int i = 0; i < size; i++)
	{
		cout << setw(10) << X[i] << " ";
	}
	for (int row = 0; row < size; row++)
	{
		for (int col = 0; col < size; col++)
		{
			x = x + Matrix[row][col] * X[col];
		}
		v.push_back(f[row] - x);
		x = 0;
	}


	sort(v.begin(), v.end());
	cout << "Максимум-норма невязки" << endl;
	cout << abs(v[v.size() - 1]) << endl;
	v.clear();


	for (int i = 0; i < size; i++)
```

```
{
        v.push_back(X[i] - y[i]);
}
sort(v.begin(), v.end());
cout << "Максимум-норма погрешности" << endl;
cout << abs(v[v.size() - 1]);
```

Результаты:



Вывод:

Метод прогонки есть метод исключения Гаусса без выбора главного элемента, примененный к системе уравнений с трехдиагональной матрицей. И учет специфики задачи позволяет построить алгоритмы с меньшими, по сравнению с универсальными, вычислительными затратами.