БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Лабораторная работа №4

Решение систем линейных алгебраических уравнений итерационными методами (простой итерации, Гаусса, Зейделя, Гаусса-Зейделя)

Выполнил:
Студент 2 курса 5 группы ФПМИ
Карасик Семён

Руководитель:
Радкевич Елена Владимировна

Минск, 2017 г.

Оглавление

Лабораторная работа №4....1
Постановка задачи....3
Общие сведения об итерационных методах....4
Метод простой итерации....4
Метод Якоби....4
Метод Зейделя....4
Метод Гаусса-Зейделя....4
Листинг основных функций программы....5
Входные данные....7
Выходные данные....7

Постановка задачи

Решить систему линейных алгебраических уравнений методами простой итерации, Гаусса, Зейделя, Гаусса-Зейделя.

Общие сведения об итерационных методах

Пусть дана СЛАУ $Ax=f$ и мы смогли представить ее в эквивалентном виде $x=Bx+b$, тогда попробуем найти последовательность $\left(x^{(k)}\right)$ такую, что $\lim_{x\to+\infty} x^{(k)}=x^{(*)}, где\, x^{(*)}-точное\ решение: Ax^{(*)}=f$, последовательность строится по данному соотношению: $x^{(\mathrm{k})}=Bx^{(k)}+b$, в качестве $x^{(0)}$ можно выбрать произвольный метод, но часто выбирается вектор f.

В случае $\lim_{x\to+\infty} x^{(k)}=x^{(*)}$ такой метод называют *сходящимся.* Сходимость метода зависит от матрицы B.

Условие выхода из итерационного процесса $\left|x_i^{(k+1)}-x_i^{(k)}\right|\leq\epsilon, i=\overline{1,n}$

Оценка скорости сходимости итерационного процесса: $k\geq\log_{\|B\|}\frac{\epsilon(1-\|B\|)}{\|b\|}$

Метод простой итерации

Самым простым из методов является метод простой итерации.

$$B=E-\frac{A^T A}{\|A^T A\|}$$

$$b=\frac{A^T f}{\|A^T A\|}$$

Метод Якоби

$$B_{ij}=\begin{cases}-A_{ij}/A_{ii}, i\neq j\\ 0, i=j\end{cases}, i,j=\overline{1,n}$$
$$b_i=f_i/A_{ii}, i=\overline{1,n}$$

Метод Зейделя

Заметим, что в методе простой итерации мы не используем тот факт, что при вычисления следующего уравнения мы можем использовать уже вычисленные "лучшие" компоненты вектора x. Метод Зейделя использует данный факт, тогда:

$$x^{k+1}=Lx^{k+1}+Ux^k+b, где\ L-нижний\ треугольник\ матрицы\ A, U-верхний\ треугольник\ A\ с\ диагональю.$$

Метод Гаусса-Зейделя

Метод Гаусса-Зейделя совмещает в себе преимущества методов Гаусса и Зейделя: применим к матрице метода Гаусса использование уже вычисленных компонент вектора-решений при вычислении следующих.

Листинг основных функций программы

```
bool eq(double x, double y) {
	return abs(x - y) <= EPS;
}

bool eq(const Vector & x, const Vector & y)
{
	if (x.size() != y.size())
		throw invalid_argument("Vectors
have different size");
	for (int i = 0; i < x.size(); i++)
		if (!eq(x[i], y[i]))
			return false;
	return true;
}

double computeNorm(const Vector & v)
{
	double norm = 0;
	for (int i = 0; i < v.size(); i++)
		norm = max(norm, abs(v[i]));
	return norm;
}

double computeNorm(const Matrix & mat)
{
	double norm = 0;
	for (int i = 0; i < getN(mat); i++)
	{
		double sum = 0;
		for (int j = 0; j < getM(mat); j+
+)
			sum += abs(mat[i][j]);
		norm = max(norm, sum);
	}

	return norm;
}

double computeCovergenceSpeed(const Matrix & B,
const Vector & b)
{
	double normB = computeNorm(B), normb =
computeNorm(b);
	double numerator = log(EPS*(1 - normB) /
normb);
	double denumerator = log(normB);
	return trunc(numerator / denumerator) +
1;
}

Vector computeError(const Matrix & A, const
Vector & f, const Vector & x)
{
	Vector error(x.size(), 0), f1 = A * x;
	for (int i = 0; i < error.size(); i++)
		error[i] = abs(f1[i] - f[i]);
	return error;
}

void computeMatrixAndVectorForSimpleIteration(
	const Matrix & A, const Vector & f,
	Matrix & B, Vector & b)
{
	Matrix symmetrical = transpose(A) * A;
	double norm = computeNorm(symmetrical);
	B = loadIdentity(getN(A)) - symmetrical /
norm;
	b = transpose(A) * f / norm;
}

Vector simpleIterationMethod(const Matrix & A,
const Vector & f, MethodSummary & methodSummary)
{
	Matrix B;
	Vector b;

computeMatrixAndVectorForSimpleIteration(A, f,
B, b);
	Vector x = b, oldX;

	int stepCnt = 0;
	do
	{
		oldX = x;
		x = B * oldX + b;
		stepCnt++;
	} while (!eq(x, oldX));

	methodSummary.B = B;
	methodSummary.b = b;
	methodSummary.stepCnt= stepCnt;
	methodSummary.x = x;
	methodSummary.error = computeError(A, f,
methodSummary.x);
	methodSummary.theoreticalStepCnt =
computeCovergenceSpeed(B, b);
	return x;
}


Vector JacobiMethod(const Matrix & A, const
Vector & f, MethodSummary & methodSummary)
{
	const int n = getN(A);
	Matrix B = A;
	Vector b = f;
	for (int i = 0; i < b.size(); i++)
		b[i] /= A[i][i];
	for (int i = 0; i < n; i++)
	{
		for (int j = 0; j < n; j++)
			if (i == j)
				B[i][j] = 0;
			else
				B[i][j] = -B[i]
[j] / A[i][i];
	}

	int stepCnt = 0;
	Vector x = f, oldX;
	do
	{
		oldX = x;
		x = B * oldX + b;
		stepCnt++;
	} while (!eq(x, oldX));

	methodSummary.stepCnt = stepCnt;
	methodSummary.B = B;
	methodSummary.b = b;
	methodSummary.x = x;
```

```
	methodSummary.error = computeError(A, f, 
methodSummary.x);
	methodSummary.theoreticalStepCnt = 
computeCovergenceSpeed(B, b);
	return x;
}

Vector SeidelMethod(const Matrix & A, const 
Vector & f, MethodSummary & methodSummary)
{
	const int n = getN(A);
	Matrix B;
	Vector b;
	
computeMatrixAndVectorForSimpleIteration(A, f, 
B, b);
	Vector x = b, oldX;
	int stepCnt = 0;
	do
	{
		oldX = x;
		for (int i = 0; i < n; i++)
		{
			double sum = 0;
			for (int j = 0; j < i; j+
+)
				sum += x[j] * B[i]
[j];
			for (int j = i; j < n; j+
+)
				sum += oldX[j] * 
B[i][j];
			x[i] = sum + b[i];
		}
		stepCnt++;
	} while (!eq(x, oldX));

	methodSummary.B = Matrix();
	methodSummary.b = f;
	methodSummary.error = computeError(A, f, 
x);
	methodSummary.stepCnt = stepCnt;
	methodSummary.theoreticalStepCnt = 0;
	methodSummary.x = x;
	return x;
}
```

```
Vector GaussSeidelMethod(const Matrix & A, const 
Vector & f, MethodSummary & methodSummary)
{
	const int n = getN(A);
	Vector b = f;
	for (int i = 0; i < n; i++)
		b[i] /= A[i][i];
	Matrix B = loadMatrix(n, n);
	for (int i = 0; i < n; i++)
		for (int j = 0; j < n; j++)
			if (i == j)
				B[i][j] = 0;
			else
				B[i][j] = -A[i]
[j] / A[i][i];

	Vector x = b, oldX;
	int stepCnt = 0;
	do
	{
		oldX = x;
		for (int i = 0; i < n; i++)
		{
			x[i] = 0;
			for (int j = 0; j < i; j+
+)
				x[i] += B[i][j] * 
x[j];
			for (int j = i; j < n; j+
+)
				x[i] += B[i][j] * 
oldX[j];
			x[i] += b[i];
		}
		stepCnt++;
	} while (!eq(x, oldX));

	methodSummary.B = B;
	methodSummary.b = b;
	methodSummary.error = computeError(A, f, 
x);
	methodSummary.stepCnt = stepCnt;
	methodSummary.theoreticalStepCnt = 0;
	methodSummary.x = x;
	return x;
}
```

Входные данные

input.txt:
```
5
-0.0305 -0.0914 -0.0741 -0.0122
0.0000 0.0000 0.0122 0.0000
0.4974 0.3284 0.5887 0.5887 0.4263
1.5875 -1.7590 1.4139 1.7702 -2.0767
```

Выходные данные

```
output.txt
Method of simple iteration
B:
  0.5980  0.0186  0.0932 -0.1049 -0.1354
  0.0186  0.8115  0.0938  0.0338  0.0154
  0.0932  0.0938  0.3714  0.0816 -0.1028
 -0.1049  0.0338  0.0816  0.4222 -0.0127
 -0.1354  0.0154 -0.1028 -0.0127  0.6675
b:
  1.3994 -1.0357  0.3046  2.1522 -1.0124
x:
  5.0006 -3.9559  2.0054  3.0003 -5.9988
error:
9.9560e-05  1.4503e-04  1.0864e-04  7.3447e-06  1.6532e-04
iteration count: 46
theoretical iteration count: 499

Jacobi method
B:
  0.0000 -0.0000  0.2612 -0.1838 -0.3062
  0.0929  0.0000 -0.0000  0.1885 -0.0618
 -0.0173  0.1553  0.0000 -0.0190 -0.0603
 -0.0518 -0.0000  0.1259  0.0000 -0.0000
 -0.0476  0.0715 -0.3453  0.0286  0.0000
b:
  3.1916 -5.3563  2.4017  3.0070 -4.8715
x:
  5.0010 -3.9554  2.0057  3.0003 -5.9993
error:
3.3820e-06  7.4916e-07  2.1202e-06  4.4436e-09  2.1324e-06
iteration count: 9
theoretical iteration count: 43

Seidel method
b:
  1.5875 -1.7590  1.4139  1.7702 -2.0767
x:
  5.0007 -3.9555  2.0055  3.0003 -5.9990
error:
7.8004e-05  1.3120e-05  6.2919e-05  1.6944e-05  1.1173e-04
iteration count: 41

Gauss-Seidel method
```

```
B:
  0.0000 -0.0000  0.2612 -0.1838 -0.3062
  0.0929  0.0000 -0.0000  0.1885 -0.0618
 -0.0173  0.1553  0.0000 -0.0190 -0.0603
 -0.0518 -0.0000  0.1259  0.0000 -0.0000
 -0.0476  0.0715 -0.3453  0.0286  0.0000
b:
  3.1916 -5.3563  2.4017  3.0070 -4.8715
x:
  5.0010 -3.9554  2.0057  3.0003 -5.9993
error:
1.9200e-06  6.4513e-08  2.2497e-07  0.0000e+00  0.0000e+00
iteration count: 5
```