Лабораторная работа №2
Тема «РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХУРАВНЕНИЙ»

Дана система n алгебраических уравнений с n неизвестными:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n = b_2 \\ ..... \\ a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n = b_n \end{cases} \quad (1)$$

Эту систему можно записать в матричном виде: $A \bullet X = B$, где

$$A = \begin{bmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ . & . & . & . \\ a_{n1} & a_{n1} & & a_{nn} \end{bmatrix}; \qquad X = \begin{bmatrix} x_1 \\ x_2 \\ . \\ x_n \end{bmatrix}; \qquad B = \begin{bmatrix} b_1 \\ b_2 \\ . \\ b_n \end{bmatrix}.$$

A - квадратная матрица коэффициентов, X - вектор-столбец неизвестных, B - вектор-столбец свободных членов.

Численные методы решения систем линейных уравнений делятся на прямые и итерационные.

Прямые методы используют конечные соотношения для вычисления неизвестных. Эти методы сравнительно просты и пригодны для широкого класса систем. Недостатки: требуют хранения в памяти ЭВМ сразу всей матрицы A. При больших порядках системы расходуется много места в памяти и накапливается вычислительная погрешность. Кроме того, существенно возрастает время вычисления вектора X. Поэтому прямые методы обычно применяют при небольших порядках системы (n<200). Примеры прямых методов - метод определителей Крамера, метод Гаусса. Первый из них применяется крайне редко, так как с ростом n алгоритм нахождения определителей резко возрастает. Метод Гаусса будет подробно рассмотрен в дальнейшем.

Итерационные методы основаны на последовательных приближениях. Задается некоторое приближенное значение вектора X – начальное приближение. Затем с помощью некоторого алгоритма проводится первый цикл вычислений – итерация, в результате которого получается новое приближение вектора X. Итерации проводятся до получения решения с заданной точностью. Алгоритм решения систем линейных уравнений здесь более сложен, чем у прямых методов. Не всегда выполняется условие сходимости. Однако, в ряде случаев итерационные методы предпочтительнее. Они требуют хранения в памяти ЭВМ не всей матрицы A, а лишь нескольких векторов. Вычислительная погрешность практически не накапливается. Поэтому итерационные методы применимы и для больших порядков системы. Примеры - метод простой итерации и метод Зейделя.

Метод Гаусса

Метод основан на приведении матрицы системы к треугольному виду. Это достигается последовательным исключением неизвестных из уравнений системы. Сначала с помощью первого уравнения исключается x_1 из всех последующих уравнений. Затем с помощью второго уравнения исключается x_2 из последующих и т.д. Этот процесс называется прямым ходом метода Гаусса и продолжается до тех пор, пока в левой части последнего n-го уравнения не останется лишь один член с неизвестным x_n. В результате прямого хода система принимает вид:

$$\begin{cases} x_1 + a'_{12}x_2 + \ldots + a'_{1n}x_n = b'_1 \\ x_2 + \ldots + a'_{2n}x_n = b'_2 \\ \ldots \quad \ldots \\ x_n = b'_n \end{cases} \qquad (2)$$

Обратный ход метода Гаусса состоит в последовательном вычислении искомых неизвестных, начиная с x_n и кончая x_1.

Пример 1: Рассмотрим реализацию метода Гаусса. Дана система уравнений четвертого порядка:

$$\begin{cases} 4x_1 + 2x_2 - 4x_3 + 2x_4 = 12, \\ 2x_1 + 9x_2 - 4x_3 - 4x_4 = 14, \\ 3x_1 + 2x_2 - 8x_3 + 2x_4 = 20, \\ 4x_1 + 2x_2 + 5x_3 + 7x_4 = 10. \end{cases}$$

Решение

Ручной счет:

Запишем систему в виде матрицы, включив коэффициенты уравнений и свободные члены:

A1	4	2	-4	2	12
A2	2	9	-4	-4	14
A3	3	2	-8	2	20
A4	4	2	5	7	10

Буква А в названии строки означает начальное приближение (при дальнейших действиях будут применяться следующие буквы латинского алфавита).

Алгоритм прямого хода метода Гаусса:

1. Нормируем первое уравнение, разделив каждый член на коэффициент a_{11}.

2. Умножаем коэффициенты полученного уравнения на первые коэффициенты остальных уравнений (a_{21}, a_{31}).

3. Полученные при перемножении результаты последовательно вычитаем из соответствующих уравнений.

В результате матрица принимает следующий вид:

B1=A1/4	1	0,5	-1	0,5	3
B2=A2-B1*2	0	8	-2	-5	8
B3=A3-B1*3	0	0,5	-5	0,5	11
B4=A4-B1*4	0	0	9	5	-2

Видно, что члены, содержащие x_1 исключились из всех уравнений, кроме первого. Далее работаем с системой второго порядка (исключаем члены, содержащие x_2 из третьего уравнения). В результате матрица принимает следующий вид:

C1=B1	1	0,5	-1	0,5	3
C2=B2/8	0	1	-0,25	-0,625	1
C3=B3-C2*0,5	0	0	-4,875	0,813	10,5
C4=B4	0	0	9	5	-2

Видно, что члены, содержащие x_2 исключились из всех уравнений, кроме первого и второго. Далее работаем с системой третьего порядка (исключаем члены, содержащие x_3 из третьего уравнения). В результате матрица принимает следующий вид:

D1=B1	1	0,5	-1	0,5	3
D2=B2	0	1	-0,25	-0,625	1
D3=B3/(-4,875)	0	0	1	-0,167	-2,154
D4=B4-D3*9	0	0	0	6,5	17,385

Наконец, нормируем последнее уравнение:

E1=D1	1	0,5	-1	0,5	3
E2=D2	0	1	-0,25	-0,625	1
E3=D3	0	0	1	-0,167	-2,154
E4=D4/(6,5)	0	0	0	6,5	17,385

В результате прямого хода метода Гаусса мы получили следующую систему уравнений, имеющую треугольный вид:

$$\begin{cases} x_1 - 0{,}5x_2 - x_3 + 0{,}5x_4 = 3, \\ x_2 - 0{,}25x_3 - 0{,}625x_4 = 1, \\ x_3 - 0{,}167x_4 = -2{,}154, \\ 6{,}5x_4 = 17{,}385. \end{cases}$$

Обратный ход метода Гаусса существенно проще. Сначала из последнего уравнения вычисляем x_4, затем из третьего – x_3, из второго – x_2, наконец, из первого – x_1. В результате получим:

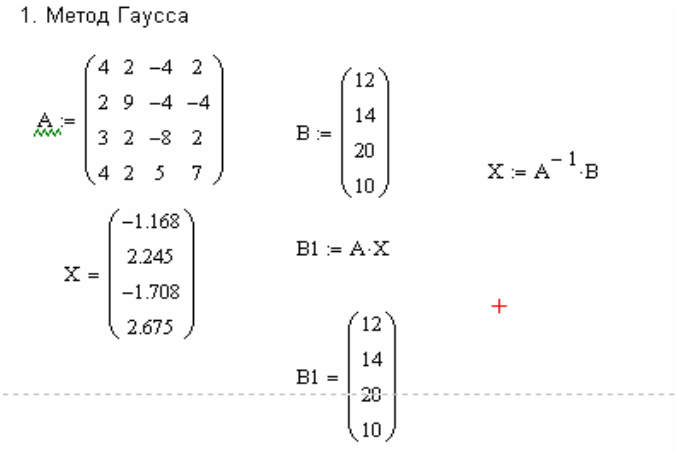
x_4=2,67; x_3 = -1,71; x_2 = 2,24; x_1 = -1,17.

Реализация в Microsoft Excel:

Решение системы методом Гаусса						
4	2	-4	2	12		
2	9	-4	-4	14		
3	2	-8	2	20		
4	2	5	7	10		
1	0,5	-1	0,5	3		
0	8	-2	-5	8		
0	0,5	-5	0,5	11		
0	0	9	5	-2		
1	0,5	-1	0,5	3		
0	1	-0,25	-0,625	1		
0	0	-4,875	0,8125	10,5		
0	0	9	5	-2		
1	0,5	-1	0,5	3		
0	1	-0,25	-0,625	1		
0	0	1	-0,1667	-2,1538		
0	0	0	6,5	17,3846		
1	0,5	-1	0,5	3	**x1=**	**-1,1677**
0	1	-0,25	-0,625	1	**x2=**	**2,24458**
0	0	1	-0,1667	-2,1538	**x3=**	**-1,7081**
0	0	0	1	2,67456	**x4=**	**2,67456**

Приведенный алгоритм является наиболее строгой реализацией метода Гаусса и применяется в стандартных программах ЭВМ. На практике можно использовать более простые действия для приведения системы к треугольному виду (переставлять местами уравнения, проводить между ними любые линейные операции).

Реализация в Mathad:

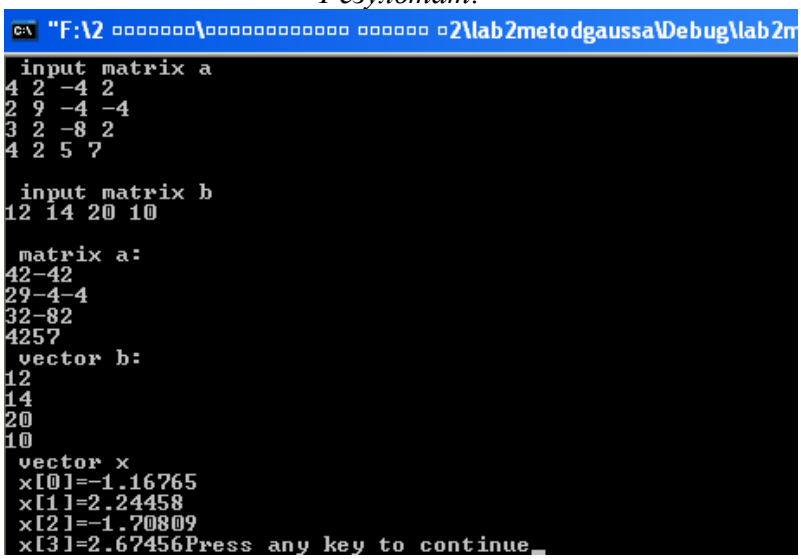


Реализация в Microsoft Visual C++ :

```
#include "stdafx.h"
#include "iostream.h"

int gauss(double a[4][4],double b[4], double x[4]);

int main(int argc, char* argv[])
{
	double a[4][4];double b[4]; double x[4];
	int i,j;
	cout<<"\n input matrix a\n";
	for(i=0;i<4;i++)
		for(j=0;j<4;j++)
			cin>>a[i][j];
		cout<<"\n input matrix b\n";
		for(i=0;i<4;i++)
			cin>>b[i];
		cout<<"\n matrix a:";
		for(i=0;i<4;i++)
		{
			cout<<"\n";
			for(j=0;j<4;j++)
			cout<<""<<a[i][j];
		}
		cout<<"\n vector b:";
		for(i=0;i<4;i++)
			cout<<"\n"<<b[i];
		gauss(a,b,x);
		cout<<"\n vector x";
		for(i=0;i<4;i++)
			cout<<"\n x["<<i<<"]="<<x[i];
		return 0;
}

int gauss(double a[4][4],double b[4], double x[4])
//метод Гаусса
{
	int i,k,m,im;
	long double amm,aim;

	for(m=0;m<=4-2;m++)
	{
		amm=a[m][m];
		im=m;
		for(k=m;k<=4-1;k++)
			a[m][k]=a[m][k]/amm;
			b[m]=b[m]/amm;
			for(i=m+1;i<=4-1;i++)
			{
				aim=a[i][m];
				for(k=m;k<=4-1;k++)
					a[i][k]=a[i][k]-a[m][k]*aim;
```

```
                        b[i]=b[i]-b[m]*aim;
                    }//end i
        }//end m
        x[4-1]=b[4-1]/a[4-1][4-1];
        for(i=4-2;i>=0;i--)
        {
                x[i]=b[i];
                for(k=i+1;k<4;k++)
                        x[i]=x[i]-a[i][k]*x[k];
        }//end i
        return 0;
}//end gauss
```

Результат:



Метод простой итерации

Пусть дана система n алгебраических уравнений с n неизвестными:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases} \qquad (1)$$

Решение систем линейных уравнений с помощью метода простой итерации сводится к следующему алгоритму.

1. Проверка условия сходимости. Для сходимости метода необходимо и достаточно, чтобы в матрице А абсолютные значения всех диагональных элементов были больше суммы модулей всех остальных элементов в соответствующей строке:

$$|a_{ii}| > \sum_{i=1, i\neq j}^{n} |a_{ij}| .$$

Недостатком итерационных методов является это достаточно жесткое условие сходимости, которое выполняется далеко не для всех систем.

2. Если условие сходимости выполнено, то на следующем этапе необходимо задать начальное приближение вектора неизвестных, в качестве которого обычно выбирается нулевой вектор:

$$x_1^{(0)} = x_2^{(0)} = ... = x_n^{(0)} = 0.$$

Заметим, что здесь и в дальнейшем нижний индекс обозначает соответствующую компоненту вектора неизвестных, а верхний индекс – номер итерации (приближения).

3. Затем организуется циклический вычислительный процесс каждый цикл которого представляет собой одну итерацию. В результате каждой итерации получается новое значение вектора неизвестных. Для организации итерационного процесса запишем систему (1) в приведенном виде. При этом слагаемые, стоящие на главной диагонали нормируются и остаются слева от знака равенства, а остальные переносятся в правую часть. Приведенная система уравнений имеет вид:

$$\begin{cases} x_1^{(k)} = \left[b_1 - \left(a_{12}x_2^{(k-1)} + ... + a_{1n}x_n^{(k-1)}\right)\right] / a_{11} \\ x_2^{(k)} = \left[b_2 - \left(a_{21}x_1^{(k-1)} + ... + a_{2n}x_n^{(k-1)}\right)\right] / a_{22} \\ ..... \\ x_n^{(k)} = \left[b_n - \left(a_{n1}x_1^{(k-1)} + ... + a_{n,n-1}x_{n-1}^{(k-1)}\right)\right] / a_{nn} \end{cases} \qquad (3)$$

4. Итерационный процесс заканчивается, если для каждой i-й компоненты вектора неизвестных будет выполнено условие достижения точности:

$$\left|x_i^{(k)} - x_i^{(k-1)}\right| < \varepsilon$$

где k - номер итерации, ε - заданная точность.

Пример 2: Рассмотрим реализацию метода простой итерации. Дана система уравнений четвертого порядка:

$$\begin{cases} 20 \cdot x_1 + 2 \cdot x_2 - 3 \cdot x_3 - 2 \cdot x_4 = 0 \\ x_1 + 12 \cdot x_2 + 2 \cdot x_3 - 4 \cdot x_4 = 2 \\ -3 \cdot x_1 + 5 \cdot x_2 + 13 \cdot x_3 + x_4 = 4 \\ -2 \cdot x_1 + x_2 - 3 \cdot x_3 + 15 \cdot x_4 = 6 \end{cases}$$

Требуется найти две первых итерации вектора неизвестных с помощью метода простой итерации.

Решение

Ручной счет:

Применим алгоритм метода.

1. Проверка условия сходимости.

$|20|>|2|+|-3|+|-2|$ - да; $|12|>|1|+|2|+|-4|$ - да; $|13|>|-3|+|5|+|1|$ - да; $|15|>|-2|+|1|+|-3|$ - да; сходимость есть.

2. Выбор начального приближения: $x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = x_4^{(0)} = 0$.

3. Запись приведенной системы уравнений:

$$\begin{cases} x_1^{(k)} = \frac{1}{20} \cdot \left[(-1) \cdot x_2^{(k-1)} + 3 \cdot x_3^{(k-1)} + 2 \cdot x_4^{(k-1)}\right], \\ x_2^{(k)} = \frac{2}{12} - \frac{1}{12} \cdot \left[1 \cdot x_1^{(k-1)} + 2 \cdot x_3^{(k-1)} - 4 \cdot x_4^{(k-1)}\right], \\ x_3^{(k)} = \frac{4}{13} + \frac{1}{13} \cdot \left[3 \cdot x_1^{(k-1)} - 5 \cdot x_2^{(k-1)} - 1 \cdot x_4^{(k-1)}\right], \\ x_4^{(k)} = \frac{6}{15} + \frac{1}{15} \cdot \left[2 \cdot x_1^{(k-1)} - 1 \cdot x_2^{(k-1)} + 3 \cdot x_3^{(k-1)}\right] \end{cases}$$

4. Выполним две итерации.

<u>k=1</u>

$$\begin{cases} x_1^{(1)} = \frac{1}{20} \cdot \left[(-1) \cdot 0 + 3 \cdot 0 + 2 \cdot 0\right] = 0, \\ x_2^{(1)} = \frac{2}{12} - \frac{1}{12} \cdot \left[1 \cdot 0 + 2 \cdot 0 - 4 \cdot 0\right] = \frac{2}{12} \approx 0{,}166, \\ x_3^{(1)} = \frac{4}{13} + \frac{1}{13} \cdot \left[3 \cdot 0 - 5 \cdot 0 - 1 \cdot 0\right] = \frac{4}{13} \approx 0{,}308, \\ x_4^{(1)} = \frac{6}{15} + \frac{1}{15} \cdot \left[2 \cdot 0 - 1 \cdot 0 + 3 \cdot 0\right] \approx \frac{6}{15}. \end{cases}$$

<u>k=2</u>

$$\begin{cases} x_1^{(2)} = \frac{1}{20} \cdot \left[(-1) \cdot 0{,}166 + 3 \cdot 0{,}308 + 2 \cdot 0{,}4\right] \approx 0{,}0695, \\ x_2^{(1)} = \frac{2}{12} - \frac{1}{12} \cdot \left[1 \cdot 0 + 2 \cdot 0{,}308 - 4 \cdot 0{,}4\right] \approx 0{,}25, \\ x_3^{(1)} = \frac{4}{13} + \frac{1}{13} \cdot \left[3 \cdot 0 - 5 \cdot 0{,}166 - 1 \cdot 0{,}4\right] \approx 0{,}21, \\ x_4^{(1)} = \frac{6}{15} + \frac{1}{15} \cdot \left[2 \cdot 0 - 1 \cdot 0{,}166 + 3 \cdot 0{,}308\right] \approx 0{,}45. \end{cases}$$

Заметим, что точное решение $x_1^* = 0{,}1; x_2^* = 0{,}3; x_3^* = 0{,}2; x_4^* = 0{,}5$ в данном методе никогда не будет достигнуто, однако с каждой последующей итерацией вектор неизвестных все ближе приближается к точному решению.

Реализация в Microsoft Excel:

Решение системы методом простой итерации					
k	x1	x2	x3	x4	Погрешность
0	0	0	0	0	
1	0,0000	0,1667	0,3077	0,4000	0,3077
2	0,0695	0,2487	0,2128	0,4504	0,0949
3	0,0521	0,2755	0,1934	0,4352	0,0268
4	0,0450	0,2752	0,1803	0,4273	0,0132
5	0,0422	0,2753	0,1794	0,4237	0,0027
6	0,0417	0,2745	0,1790	0,4232	0,0008
7	0,0417	0,2744	0,1792	0,4231	0,0002
8	0,0417	0,2743	0,1792	0,4231	0,0001
9	0,0418	0,2744	0,1793	0,4231	0,0000
10	0,0418	0,2744	0,1793	0,4231	0,0000

Реализация в Mathad:

$$a := \begin{pmatrix} 20 & 2 & -3 & -2 \\ 1 & 12 & 2 & -4 \\ -3 & 5 & 13 & 1 \\ -2 & 1 & -3 & 15 \end{pmatrix} \qquad b := \begin{pmatrix} 0 \\ 2 \\ 4 \\ 6 \end{pmatrix}$$

ORIGIN := 1 $\qquad k := 1..10$

$x1_1 := 0 \qquad x2_1 := 0 \qquad x3_1 := 0 \qquad x4_1 :=$

$$\begin{pmatrix} x1_{k+1} \\ x2_{k+1} \\ x3_{k+1} \\ x4_{k+1} \end{pmatrix} := \begin{bmatrix} \frac{b_1 - \left(a_{1,2} x2_k + a_{1,3} x3_k + a_{1,4} x4_k\right)}{a_{1,1}} \\ \frac{b_2 - \left(a_{2,1} x1_k + a_{2,3} x3_k + a_{2,4} x4_k\right)}{a_{2,2}} \\ \frac{b_3 - \left(a_{3,1} x1_k + a_{3,2} x2_k + a_{3,4} x4_k\right)}{a_{3,3}} \\ \frac{b_4 - \left(a_{4,1} x1_k + a_{4,2} x2_k + a_{4,3} x3_k\right)}{a_{4,4}} \end{bmatrix}$$

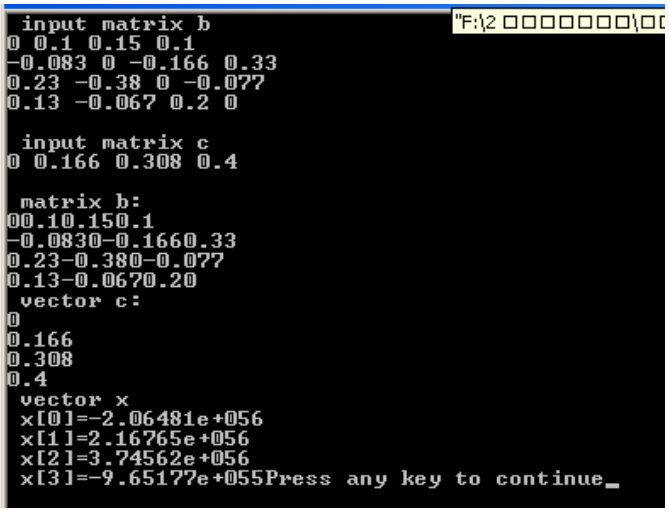
$k =$	$x1_k =$	$x2_k =$	$x3_k =$	$x4_k =$
1	0	0	0	0
2	0	0.167	0.308	0.4
3	0.069	0.249	0.213	0.45
4	0.052	0.276	0.193	0.435
5	0.045	0.275	0.18	0.427
6	0.042	0.275	0.179	0.424
7	0.042	0.274	0.179	0.423
8	0.042	0.274	0.179	0.423
9	0.042	0.274	0.179	0.423
10	0.042	0.274	0.179	0.423

Реализация в Microsoft Visual C++ :

```
#include "stdafx.h"
#include "iostream.h"
#include "math.h"
int iter(double b[4][4],double c[4], double x[4], double eps, int k_max);

int main(int argc, char* argv[])
{
        double b[4][4];
        double c[4],x[4],eps;
        int i,j,k_max;
        cout<<"input eps=";
        cin>>eps;
        cout<<"input k_max=";
        cin>>k_max;
        cout<<"\n input matrix b\n";
        for(i=0;i<4;i++)
                for(j=0;j<4;j++)
```

```
                cin>>b[i][j];
        cout<<"\n input matrix c\n";
        for(i=0;i<4;i++)
                cin>>c[i];
        cout<<"\n matrix b:";
        for(i=0;i<4;i++)
        {
                cout<<"\n";
                for(j=0;j<4;j++)
                cout<<""<<b[i][j];
        }
        cout<<"\n vector c:";
        for(i=0;i<4;i++)
                cout<<"\n"<<c[i];
        iter(b,c,x,eps,k_max);
        cout<<"\n vector x";
        for(i=0;i<4;i++)
                cout<<"\n x["<<i<<"]="<<x[i];
        return 0;
}//end main
int iter(double b[4][4],double c[4], double x[4], double eps, int k_max)
{
        int i,j,m;
        double x1[4],xerr;
        for(m=0;m<=k_max;m++)
        {
                for(i=0;i<4;i++)
                {
                        x1[i]=c[i];
                        for(j=0;j<4;j++)
                                x1[i]=x1[i]+b[i][j]*x[j];
                }
                xerr=0;
                for(i=0;i<4;i++)
                        xerr=xerr+(x1[i]-x[i])*(x1[i]-x[i]);
                xerr=sqrt(xerr);
                for(i=0;i<4;i++)
                        x[i]=x1[i];
                if(xerr<eps) break;
        }//end m
        return 0;
}//end iter
```

Результат



Метод Зейделя

Отличие метода Зейделя от метода простой итерации заключается в том, что при вычислении очередного приближения вектора неизвестных используются уже уточненные значения на этом же шаге итерации. Это обеспечивает более быструю сходимость метода Зейделя. Алгоритм метода Зейделя весьма похож на алгоритм предыдущего метода. Первые два пункта (проверка условия сходимости и выбор начального приближения), а также четвертый пункт (проверка достижения заданной точности) остаются без изменения.

Отличается здесь только третий пункт алгоритма. При вычислении x_1 используется информация об остальных неизвестных, найденных на предыдущей итерации. При вычислении x_2 используется значение x_1, найденное на текще1 итерации и значения остальных переменных, найденные на предыдущей итерации и т.д. Наконец, при вычислении последней компоненты вектора неизвестных x_n используется информация об остальных компонентах, найденных на текущей итерации. Приведенная система уравнений имеет вид:

$$\begin{cases} x_1^{(k)} = \left[b_1 - \left(a_{12}x_2^{(k-1)} + \ldots + a_{1n}x_n^{(k-1)}\right)\right] / a_{11} \\ x_2^{(k)} = \left[b_2 - \left(a_{21}x_1^{(k)} + \ldots + a_{2n}x_n^{(k-1)}\right)\right] / a_{22} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n^{(k)} = \left[b_n - \left(a_{n1}x_1^{(k)} + \ldots + a_{n,n-1}x_{n-1}^{(k)}\right)\right] / a_{nn} \end{cases} \qquad (4)$$

Пример 3: Рассмотрим реализацию метода Зейделя. Дана система уравнений четвертого порядка:

$$\begin{cases} 20\cdot x_1 + 2\cdot x_2 - 3\cdot x_3 - 2\cdot x_4 = 0 \\ x_1 + 12\cdot x_2 + 2\cdot x_3 - 4\cdot x_4 = 2 \\ -3\cdot x_1 + 5\cdot x_2 + 13\cdot x_3 + x_4 = 4 \\ -2\cdot x_1 + x_2 - 3\cdot x_3 + 15\cdot x_4 = 6 \end{cases}$$

Требуется найти две первых итерации вектора неизвестных с помощью метода Зейделя.

Решение

Ручной счет:

Применим алгоритм метода.

1. Проверка условия сходимости.
   $|20|>|2|+|-3|+|-2|$ - да; $|12|>|1|+|2|+|-4|$ - да; $|13|>|-3|+|5|+|1|$ - да; $|15|>|-2|+|1|+|-3|$ - да; сходимость есть.
2. Выбор начального приближения: $x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = x_4^{(0)} = 0$.
3. Запись приведенной системы уравнений:

$$\begin{cases} x_1^{(k)} = \frac{1}{20}\cdot\left[(-1)\cdot x_2^{(k-1)} + 3\cdot x_3^{(k-1)} + 2\cdot x_4^{(k-1)}\right], \\ x_2^{(k)} = \frac{2}{12} - \frac{1}{12}\cdot\left[1\cdot x_1^{(k)} + 2\cdot x_3^{(k-1)} - 4\cdot x_4^{(k-1)}\right], \\ x_3^{(k)} = \frac{4}{13} + \frac{1}{13}\cdot\left[3\cdot x_1^{(k)} - 5\cdot x_2^{(k)} - 1\cdot x_4^{(k-1)}\right], \\ x_4^{(k)} = \frac{6}{15} + \frac{1}{15}\cdot\left[2\cdot x_1^{(k)} - 1\cdot x_2^{(k)} + 3\cdot x_3^{(k)}\right] \end{cases}$$

4. Выполним две итерации.

<u>k=1</u>

$$\begin{cases} x_1^{(1)} = \frac{1}{20}\cdot\left[(-1)\cdot 0 + 3\cdot 0 + 2\cdot 0\right] = 0, \\ x_2^{(1)} = \frac{2}{12} - \frac{1}{12}\cdot\left[1\cdot 0 + 2\cdot 0 - 4\cdot 0\right] = \frac{2}{12} \approx 0{,}166, \\ x_3^{(1)} = \frac{4}{13} + \frac{1}{13}\cdot\left[3\cdot 0 - 5\cdot 0{,}166 - 1\cdot 0\right] \approx 0{,}244 \\ x_4^{(1)} = \frac{6}{15} + \frac{1}{15}\cdot\left[2\cdot 0 - 1\cdot 0{,}166 + 3\cdot 0{,}244\right] \approx 0{,}438. \end{cases}$$

<u>k=2</u>

$$\begin{cases} x_1^{(2)} = \frac{1}{20}\cdot\left[(-1)\cdot 0{,}166 + 3\cdot 0{,}244 + 2\cdot 0{,}438\right] \approx 0{,}064, \\ x_2^{(1)} = \frac{2}{12} - \frac{1}{12}\cdot\left[1\cdot 0{,}064 + 2\cdot 0{,}244 - 4\cdot 0{,}438\right] \approx 0{,}267, \\ x_3^{(1)} = \frac{4}{13} + \frac{1}{13}\cdot\left[3\cdot 0{,}064 - 5\cdot 0{,}267 - 1\cdot 0{,}438\right] \approx 0{,}186, \\ x_4^{(1)} = \frac{6}{15} + \frac{1}{15}\cdot\left[2\cdot 0{,}064 - 1\cdot 0{,}267 + 3\cdot 0{,}186\right] \approx 0{,}428. \end{cases}$$

Результаты свидетельствуют о более быстрой сходимости метода Зейделя по сравнению с методом простой итерации.

Реализация в Microsoft Excel:

Решение системы методом Зейделя					
k	x1	x2	x3	x4	Погрешность
0	0	0	0	0	
1	0,000	0,167	0,244	0,438	0,244
2	0,064	0,267	0,186	0,428	0,100
3	0,044	0,275	0,179	0,423	0,020
4	0,042	0,274	0,179	0,423	0,002
5	0,042	0,274	0,179	0,423	0,000
6	0,042	0,274	0,179	0,423	0,000
7	0,042	0,274	0,179	0,423	0,000
8	0,042	0,274	0,179	0,423	0,000
9	0,042	0,274	0,179	0,423	0,000
10	0,042	0,274	0,179	0,423	0,000

Реализация в Mathad:

$$a := \begin{pmatrix} 20 & 2 & -3 & -2 \\ 1 & 12 & 2 & -4 \\ -3 & 5 & 13 & 1 \\ -2 & 1 & -3 & 15 \end{pmatrix} \qquad b := \begin{pmatrix} 0 \\ 2 \\ 4 \\ 6 \end{pmatrix}$$

$$\text{ORIGIN} := 1 \qquad k := 1..10$$

$$x1_1 := 0 \qquad x2_1 := 0 \qquad x3_1 := 0 \qquad x4_1 := 0$$

$$\begin{pmatrix} x1_{k+1} \\ x2_{k+1} \\ x3_{k+1} \\ x4_{k+1} \end{pmatrix} := \begin{bmatrix} \dfrac{b_1 - \left(a_{1,2}\,x2_k + a_{1,3}\,x3_k + a_{1,4}\,x4_k\right)}{a_{1,1}} \\ \dfrac{b_2 - \left(a_{2,1}\,x1_{k+1} + a_{2,3}\,x3_k + a_{2,4}\,x4_k\right)}{a_{2,2}} \\ \dfrac{b_3 - \left(a_{3,1}\,x1_{k+1} + a_{3,2}\,x2_{k+1} + a_{3,4}\,x4_k\right)}{a_{3,3}} \\ \dfrac{b_4 - \left(a_{4,1}\,x1_{k+1} + a_{4,2}\,x2_{k+1} + a_{4,3}\,x3_{k+1}\right)}{a_{4,4}} \end{bmatrix}$$

k =	$x1_k$ =	$x2_k$ =	$x3_k$ =	$x4_k$ =
1	0	0	0	0
2	0	0.167	0.244	0.45
3	0.065	0.27	0.193	0.435
4	0.045	0.275	0.18	0.427
5	0.042	0.275	0.179	0.424
6	0.042	0.274	0.179	0.423
7	0.042	0.274	0.179	0.423
8	0.042	0.274	0.179	0.423
9	0.042	0.274	0.179	0.423
10	0.042	0.274	0.179	0.423

Реализация в Microsoft Visual C++ :

```
#include "stdafx.h"
#include "iostream.h"
#include "math.h"

int zeidel(double a[4][4],double b[4], double x[4], double eps, int k_max);
int main(int argc, char* argv[])
{
	double a[4][4];
	double  b[4],x[4],eps;
	int i,j,k_max;
	cout<<"input eps=";
	cin>>eps;
	cout<<"input k_max=";
	cin>>k_max;
	cout<<"\n input matrix a\n";
	for(i=0;i<4;i++)
		for(j=0;j<4;j++)
			cin>>a[i][j];
		cout<<"\n input matrix b\n";
		for(i=0;i<4;i++)
			cin>>b[i];
		cout<<"\n matrix a:";
		for(i=0;i<4;i++)
		{
			cout<<"\n";
			for(j=0;j<4;j++)
			cout<<""<<a[i][j];
		}
		cout<<"\n vector b:";
		for(i=0;i<4;i++)
			cout<<"\n"<<b[i];
		zeidel(a,b,x,eps,k_max);
		for(i=0;i<4;i++)
```

```
                cout<<"\n x["<<i<<"]="<<x[i];
        return 0;
}//end main
int zeidel(double a[4][4],double b[4], double x[4], double eps, int k_max)
{
        int i,j,m;
        double x1[4],xerr,s;
        for(m=0;m<=k_max;m++)
        {
                for(i=0;i<4;i++)
                {
                        s=b[i];
                        for(j=0;j<4;j++)
                                if(j!=i)
                                        s=s-a[i][j]*x1[j];
                                x1[i]=s/a[i][i];
                }//end i
                xerr=0;
                for(i=0;i<4;i++)
                        xerr=xerr+(x1[i]-x[i])*(x1[i]-x[i]);
                xerr=sqrt(xerr);
                for(i=0;i<4;i++)
                        x[i]=x1[i];
                if(xerr<eps) break;
        }//end m
        return 0;
}//end iter
```

Результат