Лабораторная работа №1

«Метод Гаусса»

выполнил Пажитных Иван, 2-й курс, 1-я группа

1) Постановка задачи

Необходимо найти решение системы линейных алгебраических уравнений вида Ax=b, где A- квадратная матрица n-ого порядка, x и b – столбцы размеров n×1.

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

Предполагается, что det A=|A|≠0. Тогда решение системы существует и оно единственно.

Нам необходимо найти:

a) Вектор решений (с точностью 6 значащих цифр)

b) Найти вектор невязки (r=Ax-b)

c) Найти определитель матрицы

d) Найти обратную матрицу

e) Найти число обусловленности матрицы v(A)=||A^(-1)||*||A||

2) 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.

<u>Прямой ход:</u>

Прямой ход метода Гаусса заключается в исключении элементов, расположенных ниже элементов, соответствующих главной диагонали матрицы A. При этом матрица A с помощью элементарных преобразований приводится к нижне треугольной, а расширенная матрица системы – к трапециевидной.

Выбор главного элемента по столбцу заключается в том, чтобы на k-ом шаге переставить строки матрицы так, чтобы наибольший по модулю элемент при x_k попал на главную диагональ, а затем выбрать его в качестве главного элемента .

Далее исключим переменную из всех уравнений, начиная с (k+1)-ого. Для этого вычтем получившуюся после перестановки k-ую строку из остальных строк, домножив её на величину, равную отношению k-ого элемента каждой из этих строк к k-ому элементу первой строки, обнуляя тем самым столбец под ним.

На k-ом шаге:

$c_{ij} = a_{ij} - a_{kj} \cdot \frac{a_{ik}}{a_{kk}}$ *где c_{ij} - новый коэффициент, a_{kk} - главный элемент*

$a_{kk} = \max_{i \geq k}(a_{ik})$

Обратный ход:
Вычисляем ответ из последнего уравнения как $x_n = \frac{b_n}{c_{nn}}$

С помощью полученного значения находим x_{n-1} из предпоследнего уравнения, и так далее, находим x_1 из первого уравнения.

Вычисление определителей и обращения: 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*

Нахождение обратной эквивалентно решению уравнения:

AX=E <=> X=A^(-1)

3) Листинг программы

```
int gauss (vector < vector<double> > a, vector < vector<double> > &e, vector<double> &ans, double &det)
{
   const double EPS=0.000000000001;                    //эквивалент нуля
   const double INF=INT_MAX;                            //эквивалент бесконечности
   int n = (int) a.size();
   int m = (int) a[0].size() - 1;
   int ins=0;

   vector<int> where (m, -1);                          //где будет ответ

   for (int col=0, row=0; col<m && row<n; col++)
   {
      int sel = row;

      for (int i=row; i<n; i++)                        //выбирает главный по столбцу
         if (fabs (a[i][col]) > fabs (a[sel][col]))
            sel = i;

      if (fabs (a[sel][col]) < EPS)                    //проверка на равенство нулю
```

```
            continue;

        for (int i=col; i<=m; i++)                                //перестановка строк
        {
            swap (a[sel][i], a[row][i]);
            swap (e[sel][i], e[row][i]);
        }

        ins++;                                            //инкрементируем кол-во перестановок
        where[col] = row;                                         //сохраняем изменеия индексов

        for (int i=0; i<n; i++)                                                  //прямой ход
            if (i != row)
            {
                double c = a[i][col] / a[row][col];
                for (int j=0; j<=m; j++)
                {
                    a[i][j] -= a[row][j] * c;                                    //исключение переменных
                    e[i][j] -= e[row][j] * c;
                }
            }
        row++;
    }

    for (int col=0, row=0; col<m; col++)                                         //обратный ход
    {
        for (int i=0; i<n; i++)
            if (i != row)
            {
                double c = a[i][col]/a[row][col];
                //a[i][col] -= a[row][col] * c;
                e[i][col] -= e[row][col] * c;
            }
        row++;
    }

    ans.assign (m, 0);
    for (int i=0; i<m; i++)                                              //получаем ответ
        if (where[i] != -1)
            ans[i] = a[where[i]][m] / a[where[i]][i];

    for (int row=0; row<n; row++)                                                //находим обратную
        for (int col=0; col<m; col++)
           e[row][col]/=a[where[row]][row];

    det=1;                                                      //вычисляем определитель
    for (int i=0; i<n; i++)
        det*=a[i][i];
    if (!ins%2)
        det*=(-1);

    for (int i=0; i<m; ++i)
        if (where[i] == -1)
            return INF;
    return 1;
}


for (int i=0; i<m; i++)                                        //находим вектор невязки
    {
```

```
        for (int j=0;j<m;j++)
          r[i]+=ans[j]*a[i][j];
        r[i]=fabs(fabs(r[i])-fabs(a[i][m]));
      }


      for (int i=0; i<n; i++)                                    //находим матрицу невязки
        for (int j=0; j<n; j++)
          for (int k=0; k<n; k++)
            R[i][j]+=a[i][k]*e[k][j];

      for (int i=0; i<n; i++)
        for (int j=0; j<n; j++)
          if (i==j)
            R[i][j]=fabs(R[i][j]);

      for (int i=0; i<n; i++)
        R[i][i]=fabs(R[i][i]-1);

      for (int i=0;i<m;i++)
        cout<<r[i]<<" ";
      cout<<endl;
    }

    double normA=0,normE=0;
    for (int i=0;i<n;i++)                                 //находим нормы и число обусловленности
    {
      double sum=0;
      for (int j=0; j<m; j++)
        sum+=a[i][j];
      normA=max(normA,sum);
    }
    for (int i=0;i<n;i++)
    {
      double sum=0;
      for (int j=0; j<m; j++)
        sum+=e[i][j];
      normE=max(normE,sum);
    }
  nu = normA*normE;
```

4) Результат и его анализ

Расширенная матрица коэффициентов:

0.6897	-0.0908	0.0182	0.0363	0.1271	4.2108
0.0944	1.0799	0.0000	-0.0726	0.0726	4.6174
0.0545	0.0000	0.8676	-0.2541	0.1452	-5.8770

-0.1089 0.2287 0.0000 0.8531 -0.0363 2.7842
0.4538 0.0000 0.1634 0.0182 1.0164 0.2178

Вектор решений:
7.00099 3.99995 -6.00023 2.99988 -2.0006

Определитель матрицы:
0.518224

Вектор невязки:
8.88178e-16 8.88178e-16 0 4.44089e-16 8.88178e-16

Обратная матрица:
1.55914 0.141257 0.00640836 -0.0479818 -0.207688
-0.07573 0.901731 0.011681 0.0845834 -0.0535873
0.0765532 -0.0669149 1.18136 0.34636 -0.161189
0.189041 -0.225759 -0.0105083 1.14107 0.0347398
-0.711814 -0.0482681 -0.192592 -0.0546915 1.10188

Матрица R=AA^(-1)-E:
1.11022e-16 1.21431e-17 -3.46945e-18 3.46945e-18 0
-6.93889e-18 0 -1.73472e-18 -2.60209e-18 0
-1.38778e-17 1.21431e-17 0 -6.07153e-17 -2.77556e-17
-3.46945e-18 -3.03577e-18 -1.73472e-18 0 0
1.11022e-16 0 2.77556e-17 -6.93889e-18 0

Число обусловленности:
2.39699

Вывод:
Число обусловленности небольшое (<10^2), из этого следует, что система хорошо обусловлена. Векторы невязки очень близки к нулю (порядок 10^(-16)), это говорит о том, что ответ достаточно точен.

[6.5923072 3.69949644 -5.81607733 3.27025309 0.15572746]