БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Лабораторная работа
по вычислительным методам алгебры на тему:

Решение систем линейных алгебраических уравнений с помощью меода простой итерации и метода Зейделя

Выполнил:
Архангельский И.А.

Проверил:
Кондратюк А.П.

Минск, 2012

Входные и выходные данные.

Входные данные

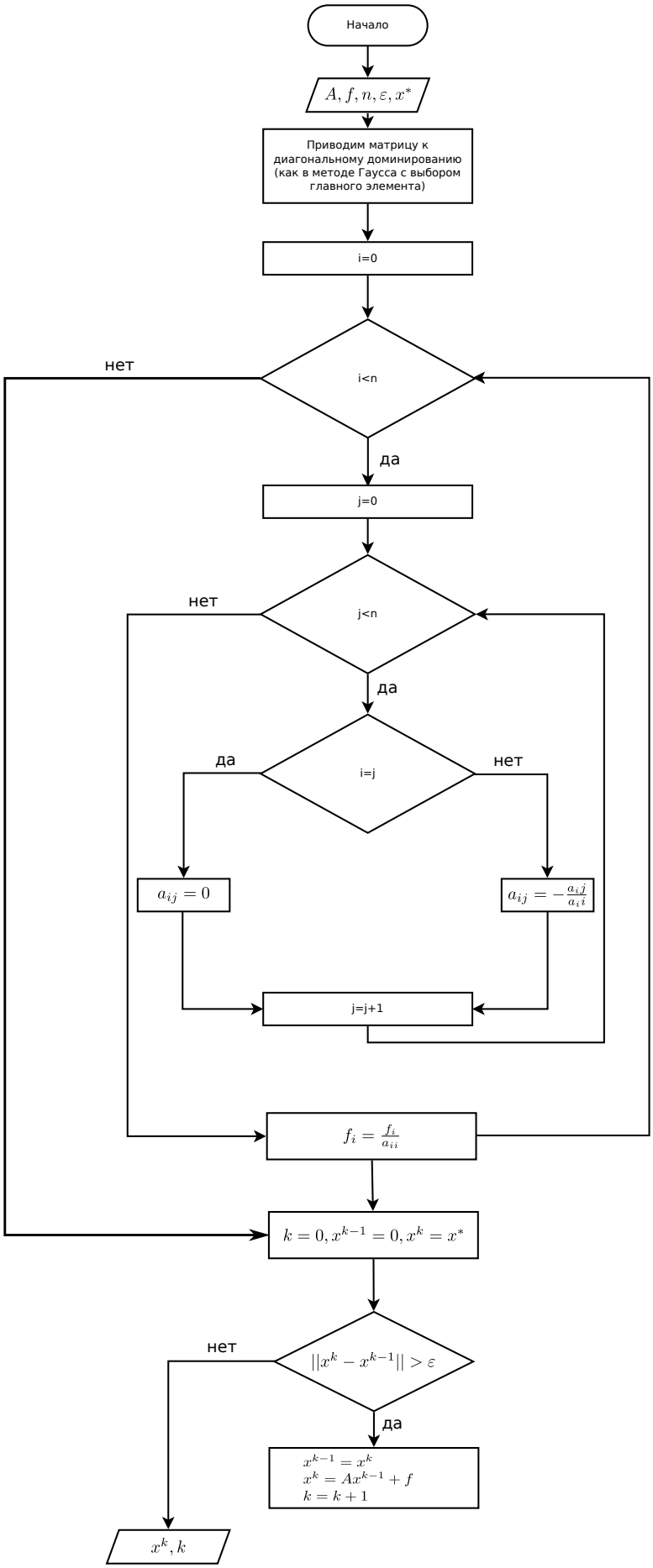
Входной файл содержит матрицу $(A|f)$, где A - квадратная матрица коэффициентов СЛУ, f - вектор-столбец свободных членов. ε - погрешность. x^0 - начальное приближение.

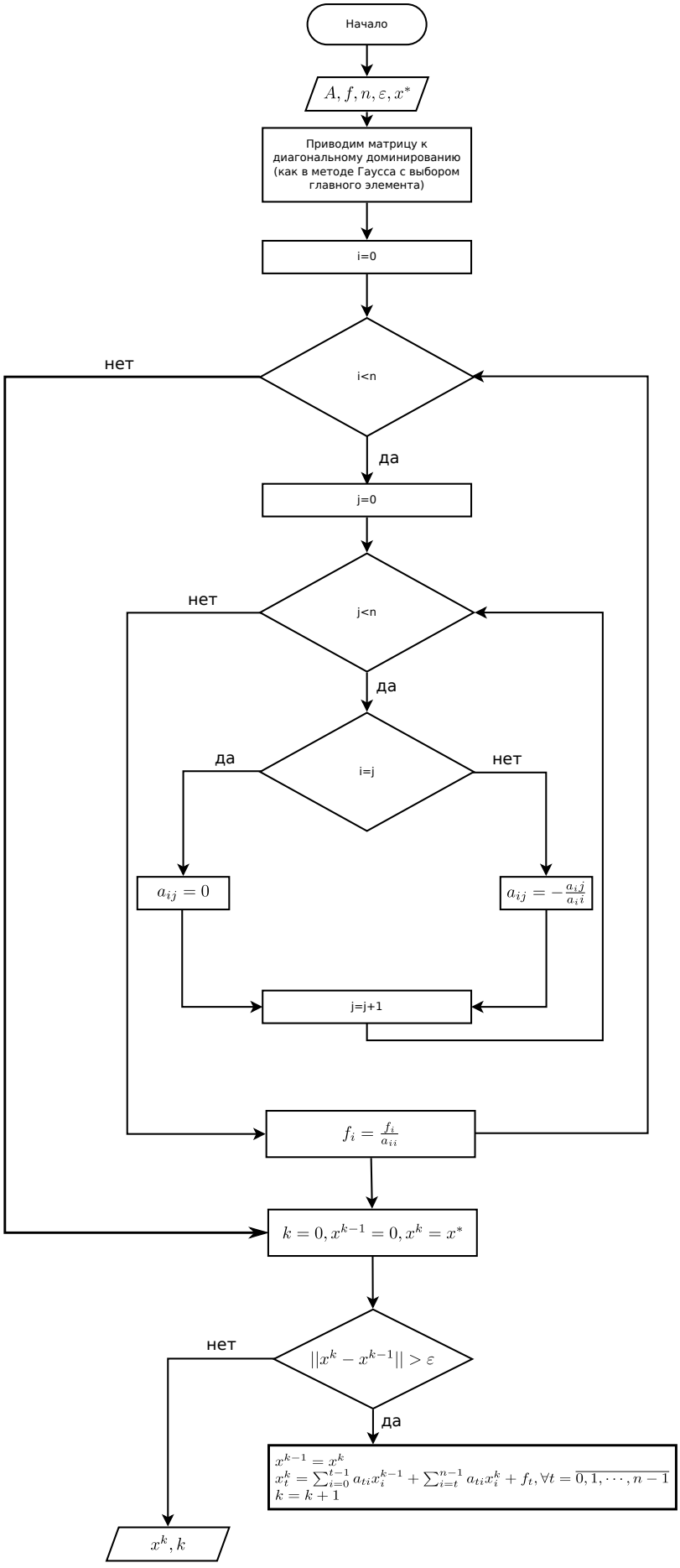
Выходные данные

Выходной файл содержит решения СЛУ полученные двумя методами, и количество выполненных итераций для каждого метода соответсвенно.

Блок-схема

Метод простой итерации




Начало
$A, f, n, \varepsilon, x^*$
Приводим матрицу к диагональному доминированию (как в методе Гаусса с выбором главного элемента)
i=0
i<n
нет
да
j=0
j<n
нет
да
i=j
да
нет
$a_{ij} = 0$
$a_{ij} = -\frac{a_{ij}}{a_{ii}}$
j=j+1
$f_i = \frac{f_i}{a_{ii}}$
$k = 0, x^{k-1} = 0, x^k = x^*$
$||x^k - x^{k-1}|| > \varepsilon$
нет
да
$x^{k-1} = x^k$
$x_t^k = \sum_{i=0}^{t-1} a_{ti} x_i^{k-1} + \sum_{i=t}^{n-1} a_{ti} x_i^k + f_t, \forall t = \overline{0, 1, \cdots, n-1}$
$k = k + 1$
x^k, k

Реализация

iterationzeidel.h

```
#ifndef ITERATIONZEIDEL_H
#define ITERATIONZEIDEL_H

#include <stdio.h>
#include <cmath>

struct DivByZeroException{};

class IterationZeidel
{

    double** matrix;
    int size;
    double* x0;
    double epsilon;
    int* permutation;
    struct Position
    {
        unsigned int n;
        unsigned int m;
    };
    void permutate (double* &x);
    Position findDominant (int k);
    void swapRows (int x, int y);
    void swapColumns (int x, int y);
    double normMatrix (double** matr, int sz);
    double normVector (double* vector, int sz);
    void print();
public:
    IterationZeidel(char* filename);
    double* solveIteration();
    double* solveZeidel ();
    void makeIterable ();
    ~IterationZeidel();
};

#endif // ITERATIONZEIDEL_H
```

iterationzeidel.cpp

```
#include "iterationzeidel.h"

IterationZeidel::IterationZeidel(char *filename)
{
    FILE* fin = fopen (filename,"r");
    fscanf (fin,"%d\n",&size);
    matrix = new double* [size];
    for (int i=0;i<size;i++)
    {
        matrix[i]= new double [size+1];
        for (int j=0;j<size;j++)
        {
            fscanf(fin,"%lf",&matrix[i][j]);
        }
        fscanf (fin,"%lf\n",&matrix[i][size]);
        matrix[i][size]*=-1;
    }
    x0 = new double [size];

    for (int i=0;i<size;i++)
    {
        fscanf (fin,"%lf",&x0[i]);
    }
    fscanf (fin,"%lf",&epsilon);
    permutation = new int [size];
    for (int i=0;i<size;i++)
    {
        permutation[i]=i;
    }
    fclose (fin);
}

IterationZeidel::~IterationZeidel()
{
```

```
35      for (int i=0;i<size;i++)
36      {
37          delete [] matrix[i];
38      }
39      delete [] matrix;
40      delete [] x0;
41      delete [] permutation;
42  }
43
44  void IterationZeidel::makeIterable()
45  {
46      for (int i=0;i<size;i++)
47      {
48
49          Position dominantPos = findDominant(i);
50          double dominant = matrix[dominantPos.m][dominantPos.n];
51          if (fabs(dominant)<pow(10,-20)) throw DivByZeroException ();
52
53
54          if (dominantPos.m != dominantPos.n)
55          {
56              if (dominantPos.m==i)
57              {
58                  swapColumns(i,dominantPos.n);
59              }
60              if (dominantPos.n==i)
61              {
62                  swapRows(i,dominantPos.m);
63              }
64          }
65
66          for (int j=0;j<size;j++)
67          {
68              if (i==j)
69              {
70                  matrix[i][j] = 0;
71              }
72              else
73              {
74                  matrix[i][j]/=-dominant;
75              }
76          }
77          matrix[i][size]/=dominant;
78
79      }
80  }
81
82  IterationZeidel::Position IterationZeidel::findDominant(int k)
83  {
84      Position pos;
85      pos.m = k;
86      pos.n = k;
87      for (int i=k;k<size;k++)
88      {
89          if (fabs(matrix[i][k])>fabs(matrix[pos.m][pos.n]))
90          {
91              pos.m = i;
92              pos.n = k;
93          }
94          if (fabs(matrix[k][i])>fabs(matrix[pos.m][pos.n]))
95          {
96              pos.m = k;
97              pos.n = i;
98          }
99      }
100     return pos;
101 }
102
103
104 double IterationZeidel::normVector(double *vector, int sz)
105 {
106     double norm = 0;
107     for (int i=0;i<sz;i++)
108     {
109         norm += (vector[i]*vector[i]);
110     }
111     return pow(norm,0.5);
112 }
113
```

```
114 void IterationZeidel::swapRows(int x, int y)
115 {
116
117     for (int i=0;i<size+1;i++)
118     {
119         double tmp = matrix[x][i];
120         matrix [x][i] = matrix[y][i];
121         matrix [y][i] = tmp;
122     }
123
124 }
125
126 void IterationZeidel::swapColumns(int x, int y)
127 {
128     for (int i=0;i<size;i++)
129     {
130         double tmp = matrix[i][x];
131         matrix [i][x] = matrix[i][y];
132         matrix [i][y] = tmp;
133     }
134     int tmp = permutation[x];
135     permutation [x] = permutation[y];
136     permutation [y] = tmp;
137 }
138
139 double* IterationZeidel::solveIteration()
140 {
141
142     double* xK = new double [size];
143     for (int i=0;i<size;i++)
144     {
145         xK[i]=x0[i];
146     }
147
148     for (int i=0;i<size;i++)
149     {
150         double tmp = xK[i];
151         xK[i]=xK[permutation[i]];
152         xK[permutation[i]]=tmp;
153     }
154     double* xKnxt = new double [size];
155     double* xDiff = new double [size];
156     int k=0;
157     while (normVector(xDiff,size)>epsilon || k==0 )
158     {
159         k++;
160         for (int i=0;i<size;i++)
161         {
162             xKnxt[i]=matrix[i][size];
163             for (int j=0;j<size;j++)
164             {
165                 xKnxt[i]+=xK[j]*matrix[i][j];
166             }
167         }
168         for (int i=0;i<size;i++)
169         {
170             xDiff[i]=xK[i]-xKnxt[i];
171             xK[i]=xKnxt[i];
172         }
173
174
175     }
176     printf ("SIMPLE ITERATION:\n");
177     printf ("K = %d\n",k);
178     for (int i=0;i<size;i++)
179     {
180         double tmp = xKnxt[i];
181         xKnxt[i]=xKnxt[permutation[i]];
182         xKnxt[permutation[i]]=tmp;
183     }
184     for (int i=0;i<size;i++)
185     {
186         printf ("%5.3f ",xKnxt[i]);
187     }
188     printf ("\n");
189
190     delete [] xKnxt;
191     delete [] xK;
192     delete [] xDiff;
```

```
193     return NULL;
194 }
195
196 double* IterationZeidel::solveZeidel()
197 {
198     double* xK = new double [size];
199     for (int i=0;i<size;i++)
200     {
201         xK[i]=x0[i];
202     }
203
204     for (int i=0;i<size;i++)
205     {
206         double tmp = xK[i];
207         xK[i]=xK[permutation[i]];
208         xK[permutation[i]]=tmp;
209     }
210     double* xKnxt = new double [size];
211     double* xDiff = new double [size];
212     int k=0;
213     while (normVector(xDiff,size)>epsilon || k==0)
214     {
215         k++;
216         for (int i=0;i<size;i++)
217         {
218             xKnxt[i]=matrix[i][size];
219             for (int j=0;j<size;j++)
220             {
221                 if (j<i)
222                 {
223                     xKnxt[i]+=xKnxt[j]*matrix[i][j];
224                 }
225                 else
226                 {
227                     xKnxt[i]+=xK[j]*matrix[i][j];
228                 }
229             }
230         }
231         for (int i=0;i<size;i++)
232         {
233             xDiff[i]=xK[i]-xKnxt[i];
234             xK[i]=xKnxt[i];
235         }
236
237
238     }
239     printf ("ZEIDEL:\n");
240     printf ("K = %d\n",k);
241     for (int i=0;i<size;i++)
242     {
243         double tmp = xKnxt[i];
244         xKnxt[i]=xKnxt[permutation[i]];
245         xKnxt[permutation[i]]=tmp;
246     }
247     for (int i=0;i<size;i++)
248     {
249         printf ("%5.3f ",xKnxt[i]);
250     }
251     printf ("\n");
252
253     delete [] xKnxt;
254     delete [] xK;
255     delete [] xDiff;
256     return NULL;
257 }
```

main.cpp

```
1  #include <cstdlib>
2  #include <iostream>
3  #include "iterationzeidel.h"
4
5  using namespace std;
6
7  int main(int argc, char *argv[])
8  {
9      for (int i=1;i<argc;i++)
10     {
11         printf ("RUNNING ON TEST: %s\n",argv[i]);
```

```
12          IterationZeidel iter =  IterationZeidel(argv[i]);
13          double * res = NULL;
14          try
15          {
16              iter.makeIterable();
17          }
18          catch (DivByZeroException e)
19          {
20              printf ("ERR::DIVIZION BY ZERO\n");
21              continue;
22          }
23          iter.solveIteration();
24          iter.solveZeidel();
25          printf ("\n");
26          delete res;
27      }
28      return 0;
29
30  }
```

Тестовые данные

test01.in

```
3
100  30  -70  60
15  -50  -5   -40
6    2    20   28
0  0  0
0.0001
```

test01.out

```
RUNNING ON TEST: tests/test01.in
SIMPLE ITERATION:
K = 17
-1.000 -1.000 -1.000
ZEIDEL:
K = 9
-1.000 -1.000 -1.000
```

test02.in

```
3
4  1  -2  8
1  -5  1  -10
3  1  -5  10
1  1  1
0.00000001
```

test02.out

```
RUNNING ON TEST: tests/test02.in
SIMPLE ITERATION:
K = 35
-1.000 -2.000 1.000
ZEIDEL:
K = 13
-1.000 -2.000 1.000
```

test03.in

```
4
12.5      1.4       1.4       0.7       -6.8
1.2       10        1.5       2         5.6
1.1       2.2       11.2      1.3       10.3
0.7       7         3.1       15        -5.2
1         1         1         1
0.00004
```

test03.out

```
RUNNING ON TEST: tests/test03.in
SIMPLE ITERATION:
K = 17
0.679 -0.663 -0.951 0.821
ZEIDEL:
K = 6
0.679 -0.663 -0.951 0.821
```