Университет ИТМО.

Лабораторная работа №1.
Решение системы линейных алгебраических уравнений СЛАУ.

Работу выполнил студент группы 1125
Халанский Дмитрий Владимирович,
принял преподаватель Университета ИТМО
Шипилов П. А..

2014

3 вариант, метод Гаусса-Зейделя

1. Описание метода

Метод Гаусса-Зейделя заключается в разбиении матрицы коэффициентов на две части — нижнюю, включая диагональ, и верхнюю — и последующем итерационном процессе по формуле $(L+D)\vec{x}^{(k+1)} = -U\vec{x}^{(k)} + \vec{b}$ после выбора приближения $\vec{x}^{(0)}$ (в приведённой реализации приближение представляет из себя $(0, 0, \cdots, 0)$).

$$x_i^{(k+1)} = -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{(k+1)} - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_j^{(k)} + \frac{b_i}{a_{ii}}$$

2. Листинг вычисляющих функций

```c
#include <stdbool.h>
#include <unistd.h>
#include <stdlib.h>
#include <float.h>
#include <math.h>

typedef double matr_elem;
struct matrix {
        unsigned short n;
        unsigned short m;
        matr_elem **elems;
};

bool matrix_is_dominant (struct matrix *matr)
{
        bool found_strict = false;
        for (unsigned short i = 0; i < matr->m; ++i) {
                matr_elem sum;
                for (unsigned short j = 0; j < matr->n; ++j)
                        sum += abs(matr->elems[i][j]) * (j != i ? 1 : -1);
                if (sum > DBL_EPSILON)
                        return false;
                if ((sum > -DBL_EPSILON) && (sum < DBL_EPSILON))
                        found_strict = true;
        }
        return found_strict;
}

bool consistent_sle (struct matrix *a)
{
        if (!a)
                return NULL;

```

```
34          for (int i = 0; i < a->m; ++i)
35                  for (int j = 0; j < a->n; ++j)
36                          if ((i != j) &&
37                                  (abs(a->elems[i][j]) >= abs(a->elems[i][i]))
    )
38                                          return false;
39          return true;
40  }
41
42  struct matrix *comp_sle (struct matrix *a, struct matrix *b, double eps,
43                           unsigned int M)
44  {
45          if ((b->n != 1) || (a->n != a-> m) || (a->m != b->m))
46                  return NULL;
47
48          matr_elem *x = (matr_elem*)calloc(a->n, sizeof(matr_elem));
49          double m;
50          unsigned int k = 0;
51          do {
52                  ++k;
53                  m = 0;
54                  for (unsigned short i = 0; i < a->n; ++i) {
55                          matr_elem p = x[i];
56                          double s = 0;
57                          for (unsigned short j = 0; j < a->n; ++j)
58                                  if (i != j)
59                                          s += a->elems[i][j] * x[j];
60                          x[i] = (b->elems[i][0] - s) / a->elems[i][i];
61                          m += (x[i] - p) * (x[i] - p);
62                  }
63          } while ((m > eps*eps) && (k < M));
64
65          struct matrix *matr = build_matrix (1, a->n, x);
66          free(x);
67          if (k >= M)
68                  return NULL;
69          return matr;
70  }
```

3. Блок-схема численного метода

4. Пример работы программы и её результат

```
Please input the number of equations: 2
Input the coefficients and constant terms of each equationseparated by
spaces.
16 3 11
7 -11 13
Matrix is non-dominant
SLE is consistent
Please input the E of the calculation: 0.0000001
(( 0.812183     )
 ( -0.664975    ))
```

5. Выводы

Использование метода Гаусса-Зейделя не оправдано, так как решение сходится лишь для матриц с диагональным преобладанием или положительно определённых, прирост скорости исполнения существенный лишь при весьма точной начальной оценке, а избежать использования типов с плавающей точкой и императивной парадигмы программирования не представляется возможным.