УТВЕРЖДАЮ

заведующий кафедрой Вычислительной математики

и прикладных информационных технологий

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Стрыгин В.В.

*подпись*

\_\_.\_\_.20\_\_

Направление подготовки / специальность 010503 Мат. обеспечение и адм. инф. систем

Дисциплина вычислительная математика

Вид контроля отчет по лабораторной работе

***Промежуточный контроль – экзамен, зачет; текущий контроль с указанием формы***

Фамилия, имя, отчество студентов\_\_ \_\_\_Самороковский Александр Владимирович\_  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Мачнев Максим Эдуардович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Курс, группа 4 курс, 9 группа

Контрольно-измерительный материал № 2.

Решение систем линейных уравнений с разреженной матрицей специального вида

Преподаватель Корзунина В.В.

***подпись расшифровка подписи***

Дата \_\_\_.\_\_\_.2013

# 1) Постановка задачи

Систему уравнений задают шесть векторов: *a, b, c, f, p, q ∈ Rn*:

*a, b, c* – векторы для элементов матрицы А, расположенных на нижней побочной

кодиагонали, на побочной диагонали и на верхней побочной диагонали;

*p, q*– векторы для элементов *k*-й и *k+1*-й столбцов матрицы *А*;

*f*– вектор правой части системы уравнений.

Символическое изображение такой системы:

# 2) Метод решения

Преобразование матрицы

**Шаг 1.**

Идем от первой до *k*-й строки, делаем на главной диагонали 1, с ее помощью обнуляем элементы нижней поддиагонали и элементы векторов *p, q:*

for (int i = 0; i <= k - 1; i++)

{

r = 1 / b[i];

b[i] = 1;

c[i] = r \* c[i];

p[i] = r \* p[i];

q[i] = r \* q[i];

f[i] = r \* f[i];

\_f[i] = r \* \_f[i];

r = a[i];

a[i] = 0;

b[i + 1] = b[i + 1] - r \* c[i];

p[i + 1] = p[i + 1] - r \* p[i];

q[i + 1] = q[i + 1] - r \* q[i];

f[i + 1] = f[i + 1] - r \* f[i];

\_f[i + 1] = \_f[i + 1] - r \* \_f[i];

c[k] = q[k];

c[k - 1] = p[k - 1];

}

**Шаг 2.**

Идем от последней до *k+1*-й строки, делаем на главной диагонали 1, с ее помощью обнуляем элементы верхней кодиагонали и элементы векторов *p, q:*

for (int i = n - 1; i >= k + 2; i--)

{

r = 1 / b[i];

b[i] = 1;

a[i - 1] = r \* a[i - 1];

p[i] = r \* p[i];

q[i] = r \* q[i];

f[i] = r \* f[i];

\_f[i] = r \* \_f[i];

r = c[i - 1];

c[i - 1] = 0;

b[i - 1] = b[i - 1] - r \* a[i - 1];

p[i - 1] = p[i - 1] - r \* p[i];

q[i - 1] = q[i - 1] - r \* q[i];

f[i - 1] = f[i - 1] - r \* f[i];

\_f[i - 1] = \_f[i - 1] - r \* \_f[i];

a[k + 1] = q[k + 2];

a[k] = p[k + 1];

}

**Шаг 3.**

, , , и отсюда определяем и :

r = 1 / b[k + 1];

b[k + 1] = 1;

f[k + 1] = r \* f[k + 1];

\_f[k + 1] = r \* \_f[k + 1];

p[k + 1] = r \* p[k + 1];

q[k + 1] = b[k + 1];

a[k] = p[k + 1];

r = c[k];

c[k] = 0;

q[k] = c[k];

b[k] = b[k] - r \* a[k];

p[k] = b[k];

f[k] = f[k] - r \* f[k + 1];

\_f[k] = \_f[k] - r \* \_f[k + 1];

r = 1 / b[k];

b[k] = 1;

f[k] = r \* f[k];

\_f[k] = r \* \_f[k];

**Шаг 4.**

Считаем остальные иксы от *k* и до 0:

\_x[k] = \_f[k];

x[k] = f[k];

\_x[k - 1] = \_f[k - 1] - q[k - 1] \* \_x[k + 1] - p[k - 1] \* \_x[k];

x[k - 1] = f[k - 1] - q[k - 1] \* x[k + 1] - p[k - 1] \* x[k];

for (int i = k - 2; i >= 0; i--)

{

x[i] = f[i] - q[i] \* x[k + 1] - p[i] \* x[k] - c[i] \* x[i + 1];

\_x[i] = \_f[i] - q[i] \* \_x[k + 1] - p[i] \* \_x[k] - c[i] \* \_x[i + 1];

}

**Шаг 5.**

Считаем остальные иксы от *k+1* и до *n-1*:

\_x[k + 1] = \_f[k + 1] - \_x[k] \* a[k];

x[k + 1] = f[k + 1] - x[k] \* a[k];

\_x[k + 2] = \_f[k + 2] - q[k + 2] \* \_x[k + 1] - p[k + 2] \* \_x[k];

x[k + 2] = f[k + 2] - q[k + 2] \* x[k + 1] - p[k + 2] \* x[k];

for (int i = k + 3; i <= n - 1; i++)

{

\_x[i] = \_f[i] - q[i] \* \_x[k + 1] - p[i] \* \_x[k] - a[i - 1] \* \_x[i - 1];

x[i] = f[i] - q[i] \* x[k + 1] - p[i] \* x[k] - a[i - 1] \* x[i - 1];

}

# 3) Описание подпрограмм.

static public double[] gauss(int n, int k, double[] a, double[] b, double[] c, double[] p, double[] q, double[] f, out double accur) — нахождение вектора решений;

static public double[] makeF(int n, int k, double[] a, double[] b, double[] c, double[] p, double[] q, double[] x) — вычисление вектора F.

static public double[] makeX(int n, int range) — генерация вектора X.

static public void makeMatrix(int n, int k, out double[] a, out double[] b, out double[] c, out double[] p, out double[] q, int range) – генерация матрицы.

# 4) Тестирование.

