|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования  Российской Федерации | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования | | |
| «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
|  | | |
| Практическое задание № 1 | | |
| по дисциплине «Программирование Вычислений» | | |
|  | | |
| **Решение СЛАУ** | | |
|  | | |
|  | Факультет: | ПМИ |
| Группа: | ПМ-71 |
| Вариант | 6 |
| Студенты: | Камынин А.С. |
|  |  |
|  |  |
| Преподаватели: | Задорожный А. Г. |
|  | Патрушев И. И. |
|  | | |
| Новосибирск | | |
| 2019 | | |

**Цель Работы:**

Разработать программу решения СЛАУ прямым методом с хранением матрицы в профильном или ленточном формате. Исследовать накопление погрешности и ее зависимость от числа обусловленности. Сравнить реализованный метод по точности получаемого решения и количеству действий с методом Гаусса.

**Метод решения:**

Построенное разложение по варианту: (\*),  - нижняя треугольная c единицами на диагонали, - верхняя треугольная матрица. Матрица хранится в ленточном формате.

Формулы для вычисления L, U\*:

Матрица U:

Матрица L:

**Текст Программы:**

class Matrix

{

public:

Matrix \*M = this;

form \*\*al;

form \*\*au;

form \*di;

int n;

int m;

int p;

void Decompose()

{

for (int i = 0; i < n; i++)

{

form2 sumL;

form2 sumU;

for (int j = i-p, jl=0; j<=i; j++, jl++)

{

if (j < 0) continue;

sumL = 0;

sumU = 0;

for (int k = 0,ku=i-j; k < jl; k++, ku++)

{

//sumL = L(i, k)\*U(k, j);

sumL += al[i][k] \* au[j][ku];

//sumU = L(j, k)\*U(k, i);

sumU += al[j][ku] \* au[i][k];

}

//U(j, i) = A(j, i) - sumU;

//L(i, j) = (A(i, j) - sumL) / U(j, j);

if (i != j)

{

au[i][jl] = au[i][jl] - sumU;

al[i][jl] = (al[i][jl] - sumL) / di[j];

}

}

di[i] = di[i] - sumU;

}

}

void FindY(form \*y)

{

form \*b = y;

for (int i = 0; i < n; i++)

{

form2 sum = 0;

for (int j = i-p, jl=0; j < i; j++, jl++)

{

if (j < 0) continue;

sum += al[i][jl] \* y[j];

}

y[i] = b[i] - sum;

}

}

void FindX(form \*x)

{

form \*y = x;

for (int i = n - 1; i >= 0; i--)

{

x[i] = y[i] / di[i];

for (int j = i - p, jl = 0; j < i; j++, jl++)

{

if (j < 0) continue;

y[j] -= x[i] \* au[i][jl];

}

}

}

Matrix(int n)

{

this->n = n;

p = n-1;

m = 2 \* p + 1;

al = new form \* [n];

au = new form \* [n];

di = new form[n];

for (int i = 0; i < n; i++)

{

al[i] = new form[p];

au[i] = new form[p];

for (int j = 0; j < p; j++)

{

al[i][j]=0;

au[i][j]=0;

}

di[i]=0;

}

}

Matrix(Matrix \*M)

{

this->p = M->p;

this->n = M->n;

al = M->al;

m = M->m;

au = M->au;

di = M->di;

}

Matrix(Matrix\* M,int n, int p)

{

this->p = p;

this->n = n;

m = 2 \* p + 1;

al = new form \* [n];

au = new form \* [n];

di = new form[n];

for (int i = 0; i < n; i++)

{

al[i] = new form[p];

au[i] = new form[p];

for (int j = 0; j < p; j++)

{

al[i][j]=M->al[i][j];

au[i][j]=M->au[i][j];

}

di[i] = M->di[i];

}

}

Matrix(std::ifstream &fl, std::ifstream &fu, std::ifstream &fd, int n, int k)

{

this->p = k;

this->n = n;

m = 2 \* k + 1;

al = new form\*[n];

au = new form\*[n];

di = new form[n];

for (int i = 0; i < n; i++)

{

al[i] = new form[k];

au[i] = new form[k];

for (int j = 0; j < k; j++)

{

fl >> al[i][j];

fu >> au[i][j];

}

fd >> di[i];

}

}

~Matrix()

{

for (int i = 0; i < n; i++) {

delete[]al[i];

delete[]au[i];

}

delete[]al;

delete[]au;

delete[]di;

}

form & operator ()(const int &i, const int &j)

{

bool contains = false; //проверить

if (abs(i - j) > p)

{

form tmp = 0;

return tmp;

}

if (i == j) return di[i];

if (j < i) return al[i][p - (i - j)];

return au[j][p - (j - i)];

}

};

Генерация Гилбертовой матрицы:

void Hilbert(Matrix &A, form\* b)

{

const double c1 = 1;

for (int i = 0; i < A.n; i++)

{

for (int j = 0; j <=i; j++)

{

A(i, j) = c1 / (i + j + c1);

A(j, i) = c1 / (i + j + c1);

}

}

for (int i = 0; i < A.n; i++)

{

double sum = 0;

for (size\_t j = 0; j < A.n; j++)

{

sum += A(i, j) \* (j + c1);

}

b[i] = sum;

}

}

Генерация Матрицы в Плотном формате:

form\*\* ForGauss( form\* b, int n, int m)

{

form \*\*A = new form\*[n];

for (int i = 0; i < n; i++)

{

A[i] = new form[m];

for (int j = 0; j < m; j++)

{

A[i][j] = 1.0 / (i + j + 1);

}

}

for (int i = 0; i < n; i++)

{

double sum = 0;

for (size\_t j = 0; j < n; j++)

{

sum += A[i][j] \* (j + 1);

}

b[i] = sum;

}

return A;

}

Решение СЛАУ методом Гаусса с ведущим элементом:

void Gauss(form \*\*A, form\*b, int n, int m)

{

for (size\_t i = 0; i < n; i++)

{

form max = A[i][i];

int l = i;

for (int j = i; j < n; j++)

{

if (A[j][i] > max)

{

l = j;

max = A[j][i];

}

}

form \*tmp = A[i];

std::swap(A[i], A[l]);

std::swap(b[i], b[l]);

for (size\_t j = i+1; j < n; j++)

{

form mult =A[j][i] / A[i][i];

for (size\_t k = i; k < n; k++)

{

if (k == i)

{

A[j][k] = 0;

}

else A[j][k] -= mult\*A[i][k];

}

b[j] -= b[i]\*mult;

}

}

form \*x = b;

for (int i = n - 1; i >= 0; i--)

{

x[i] = b[i] / A[i][i];

/\*if (i==n-1)

{

x[i] = 10;

}\*/

for (int j = 0; j < i; j++)

{

if (j < 0) continue;

b[j] -= x[i] \* A[j][i];

}

}

}

Исследования:

Результаты исследования смотреть в таблицах



<https://docs.google.com/spreadsheets/d/1Me2aIm2_faJNwXgvSabzSBDX3v18mzj4yAxmvIfBTwM/edit?usp=sharing>

1. Исследование на матрицах Ak

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | -2 | -3 | -1 | -2 | 0 | 0 | 0 | 0 | 0 |
| -1 | 11 | -4 | -2 | -3 | -1 | 0 | 0 | 0 | 0 |
| -2 | -3 | 12 | -1 | -3 | -2 | -1 | 0 | 0 | 0 |
| -3 | -2 | -2 | 16 | -1 | -3 | -4 | -1 | 0 | 0 |
| -2 | -4 | -1 | -3 | 18 | -1 | -2 | -3 | -2 | 0 |
| 0 | -3 | -2 | -1 | -2 | 18 | -1 | -2 | -3 | -4 |
| 0 | 0 | -1 | -2 | -3 | -3 | 16 | -4 | -2 | -1 |
| 0 | 0 | 0 | -1 | -3 | -2 | -4 | 15 | -4 | -1 |
| 0 | 0 | 0 | 0 | -1 | -1 | -3 | -2 | 9 | -2 |
| 0 | 0 | 0 | 0 | 0 | -1 | -2 | -3 | -1 | 7 |

|  |
| --- |
| F |
| -19 |
| -20 |
| -10 |
| -8 |
| 3 |
| -8 |
| 8 |
| 15 |
| 13 |
| 17 |

Выводы:

С увеличением k число обусловленности растёт, вместе с ним и растёт погрешность решения.

В связи со свойством конечности мантиссы для вещественных чисел, с увеличением k Ak -> A

Так, что для k = 6 для одинарной точности и k = 15, матрица численно не разрешима.

1. Исследование на Гилбертовой матрице

Вывод:

1. При использовании двойной точности только для скалярных произведений, несмотря на увеличение точности при накоплении сумм, имеется так же погрешность при округления и отбрасывания знаков в преобразовании чисел к 4байтовому формату, так что существенного выигрыша по погрешности получить не удаётся.
2. Погрешность переходит в первый знак, когда все числа в двойной точности на k=12, и k = 6 для одинарной.
3. Исследование Метода Гаусса:

Модификация метода Гаусса заключается в выборе ведущего элемента следующем образом: выбирается максимальный по модулю элемент из расположенных в столбце под изначальным ведущем.

Сравнение производилось на Матрице Гилберта 1-13 порядков.

В некоторых компонентах Гаусс точнее, но не на много.

1. Вычисление сложности Алгоритмов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Преобразование Матрицы | Прямой Ход | Обратный Ход | Всего |
| LU |  |  |  |  |
| Gauss |  |  |  |  |
| Gauss+ |  |  |  |  |