|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | | |
|  | | | |
| Кафедра прикладной математики | | | |
|  | | | |
| Лабораторная работа № 4 | | | |
| по дисциплине «Уравнения математической физики» | | | |
| **Решение несимметричных СЛАУ** | | | |
|  | | | |
|  | Группа | ПМ-83 |
| Бригада | 7 |
|  | леонович дарьяна |
|  | олейникова екатерина |
|  |  |
| Преподаватель | Патрушев И.И., Задорожный А.Г. |
|  |  |
|  | | | |
| Новосибирск  2021 | | | |

1. **Цель работы**

Изучить особенности реализации итерационных методов BSG, BSGStab, GMRES для СЛАУ с несимметричными разреженными матрицами. Исследовать влияние предобусловливания на сходимость этих методов.

1. **Задание**

Реализовать решение СЛАУ методом BSG без предобуславливания.

1. **Анализ задачи**

Выбираем начальное приближение и полагаем:

далее на каждой итерации производятся следующие вычисления:

Условие выхода из итерационного процесса:

Для решения СЛАУ точность eps зададим 1E-25, ограничение по итерациям – 500.

1. **Формат входных данных**

Для исследования свойств BSG применим реализованный метод решения СЛАУ в решении гармонической задачи.

1. **Тестирование**

Для проверки работоспособности программы решим следующую задачу:

Область по каждой из осей от 0 до 4 с шагом 1  
По всей границе заданы первые краевые условия

Параметры:

Искомая функция

Результат работы программы:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | время (LU) |  | время (МСГ) |  | время (BSG) |
| 1,09E-16 | 0,002 | 6,68E-16 | 0,014 | 5,64E-14 | 0,035 |

Погрешность и для итерационных решателей, и для прямого достаточно мала.

1. **Исследования**

***Значения параметров***

|  |  |  |  |
| --- | --- | --- | --- |
| лямбда | сигма | хи | омега |
| 100 | 0 | 8,81E-11 | 0,001 |
| 1000 | 1000 | 1,00E-10 | 1 |
| 10 000 | 100 000 | 1,00E-09 | 100 |
|  |  |  | 1,00E+06 |
|  |  |  | 1,00E+09 |

В таблицах выписаны некоторые комбинации параметров, отражающие определенные закономерности. Результаты для всех комбинаций параметров приведены в отдельном файле.

***На сетке малого размера***

Область по каждой из осей от 0 до 9 с шагом 1  
По всей границе заданы первые краевые условия

Искомая функция

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| омега | сигма | лямбда | хи |  | время (LU), сек |  | время (МСГ) , сек |  | время (BSG) , сек |
| 0,001 | 0 | 1000 | 1E-10 | 2,49E-16 | 0,046 | 2,70E-16 | 0,084 | 2,10E-16 | 0,854 |
| 1 | 1E+06 | 100 | 8,81E-11 | 1,91E-13 | 0,044 | 3,73E-05 | 0,845 | 6,48E-12 | 7,40E-01 |
| 1E+06 | 1E+06 | 100 | 8,81E-11 | 1,27E-07 | 0,043 | 2,52E+11 | 0,697 | 4,16E+01 | 5,87E-01 |
| 1E+06 | 1E+06 | 100 | 1E-09 | 1,11E-06 | 0,044 | 6,00E+15 | 0,659 | 8,77E-01 | 6,68E-01 |

***На сетке большого размера***

Область по оси х от 0 до 10 с шагом 1, по оси у от 0 до 19 с шагом 1, по оси z от 0 до 99 с шагом 1  
По всей границе заданы первые краевые условия

Искомая функция

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| омега | сигма | лямбда | хи |  | время (LU), сек |  | время (МСГ) , сек |  | время (BSG) , сек |
| 0,001 | 0 | 1000 | 1E-10 | 8,75E-16 | 13,38 | 3,39E-16 | 2,553 | 1,13E-15 | 2,773 |
| 1 | 1E+06 | 100 | 8,81E-11 | 4,11E-13 | 12,947 | 3,39E+00 | 12,504 | 5,14E-07 | 7,976 |
| 100 | 1E+06 | 100 | 8,81E-11 | 2,52E-11 | 12,908 | 8,08E+00 | 12,494 | 0,000493 | 7,925 |
| 1E+06 | 1E+06 | 100 | 8,81E-11 | 2,01E-07 | 12,948 | 5,34E+11 | 12,397 | 61,1351 | 7,891 |
| 1E+06 | 1E+06 | 100 | 1E-10 | 2,04E-07 | 12,962 | 5,16E+11 | 12,445 | 118,445 | 8,053 |
| 1E+06 | 1E+06 | 100 | 1E-09 | 1,05E-06 | 12,885 | 5,73E+15 | 12,366 | 0,575071 | 7,946 |

1. **Выводы**

* Метод BSG требует на один вектор размера N больше памяти, чем МСГ, но метод LU разложения все еще более затратен по памяти
* По формулам на каждом шаге итерации видно, что BSG дважды умножает матрицу на вектор, в то время как в МСГ вычисляется произведение матрицы на вектор один раз.
* Когда МСГ с LU предобусловливанием и BSG сходятся, BSG сходится немного (1-2сек.) медленнее МСГ с неполной LU факторизацией.
* BSG, как и МСГ, дает ответ заданной точности, LU дает ответ всегда, но точность улучшить нельзя. Для итерационного метода повышение точности приводит к увеличению времени работы.
* МСГ может не сходиться для сильно несимметричных или плохообусловленных матриц – выход из итерационного процесса происходит по количеству итераций (метод не сходится). Эта ситуация возникает при увеличении частоты колебания функции-источника: впервые эта ситуация возникает при = 1E+6, в случаях, когда = 0 МСГ сходится и для больших омега, но наихудший вариант комбинации коэффициентов дает погрешность решения 5,73E+15 в МСГ. LU при увеличении омега тоже дает решение с большей погрешностью, но это решение все еще можно считать близким к точному (погрешность равна 1,05E-06).
* BSG дает меньшую погрешность даже для высоких частот и возможно этот метод сойдется к решению, если ограничить число итераций большим чем 500 числом. Например, на большой сетке самая большая относительная погрешность BSG равна 118, а для МСГ самая большая погрешность 5,73E+15

1. **Листинг функции BSG**

void Solver::BSG()

{

std::cout.precision(15);

double s\_Az = 0, p\_r = 0, r\_r = 0, p\_r\_prev = 0;

double a = 0, B = 0, eps2 = 1e-10;

A.Ax(x0, y); // y = A \* x0

A.b - y; // y = B - A \* x0

r = y;

p = r;

z = r;

s = r;

r\_r = r \* r;

p\_r = r\_r;

normR = sqrt(r\_r) / normB;

for (iter = 1; iter < maxIter + 1 && normR >= eps; iter++)

{

A.Ax(z, y); // y = A \* z

s\_Az = s \* y;

a = p\_r / s\_Az;

// x(k) = x(k-1) + a(k) \* z(k-1)

// r(k) = r(k-1) - a(k) \* Az(k-1)

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

{

x0.vect[i] = x0.vect[i] + z.vect[i] \* a;

r.vect[i] = r.vect[i] - y.vect[i] \* a;

}

A.ATx(s, y); // y = AT \* s

// p(k) = p(k-1) - a(k) \* ATs(k-1)

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

{

p.vect[i] = p.vect[i] - y.vect[i] \* a;

}

p\_r\_prev = p\_r;

p\_r = p \* r;

B = p\_r / p\_r\_prev;

// z(k) = r(k) + B(k) \* z(k-1)

// s(k) = p(k) + B(k) \* s(k-1)

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

{

z.vect[i] = r.vect[i] + z.vect[i] \* B;

s.vect[i] = p.vect[i] + s.vect[i] \* B;

}

r\_r = r \* r;

normR = sqrt(r\_r) / normB;

std::cout << iter << ". " << normR << std::endl;

}

}

1. **Текст программы**

***Generate.h***

#pragma once

#include <vector>

#include <fstream>

void Create\_grid\_from\_points(std::string path, double lambda = 1, double sigma = 1, double hi = 1);

void Create\_material\_file(std::string path, double lambda = 1, double sigma = 1, double hi = 1);

void Create\_time\_grid();

***Generate.cpp***

#include "Generate.h"

void Create\_material\_file(std::string path, double lambda, double sigma, double hi)

{

std::ofstream out("material.txt");

out << lambda << " " << sigma << " " << hi;

out.close();

}

void Create\_grid\_from\_points(std::string path, double lambda, double sigma, double hi)

{

std::ofstream out;

out.precision(15);

std::vector<double> all\_X, all\_Y, all\_Z;

std::ifstream in(path + "grid.txt");

double X, Y, Z, kx, ky, kz;

int Nx, Ny, Nz;

int count\_x, count\_y, count\_z;

in >> count\_x >> count\_y >> count\_z;

all\_X.resize(count\_x);

all\_Y.resize(count\_y);

all\_Z.resize(count\_z);

in >> all\_X[0] >> all\_Y[0] >> all\_Z[0];

for (int curr\_count\_x = 0; curr\_count\_x < count\_x - 1; )

{

in >> X >> Nx >> kx;

double hx;

if (kx == 1)

{

hx = (X - all\_X[curr\_count\_x]) / Nx;

for (int p = 1; p < Nx; p++)

{

all\_X[curr\_count\_x + p] = all\_X[curr\_count\_x] + hx \* p;

}

curr\_count\_x += Nx;

}

else

{

hx = (X - all\_X[curr\_count\_x]) \* (kx - 1) / (pow(kx, Nx) - 1);

for (int p = 0; p < Nx - 1; curr\_count\_x++, p++)

{

all\_X[curr\_count\_x + 1] = all\_X[curr\_count\_x] + hx \* pow(kx, p);

}

curr\_count\_x++;

}

all\_X[curr\_count\_x] = X;

}

for (int curr\_count\_y = 0; curr\_count\_y < count\_y - 1; )

{

in >> Y >> Ny >> ky;

double hy;

if (ky == 1)

{

hy = (Y - all\_Y[curr\_count\_y]) / Ny;

for (int p = 1; p < Ny; p++)

{

all\_Y[curr\_count\_y + p] = all\_Y[curr\_count\_y] + hy \* p;

}

curr\_count\_y += Ny;

}

else

{

hy = (Y - all\_Y[curr\_count\_y]) \* (ky - 1) / (pow(ky, Ny) - 1);

for (int p = 0; p < Ny - 1; curr\_count\_y++, p++)

{

all\_Y[curr\_count\_y + 1] = all\_Y[curr\_count\_y] + hy \* pow(ky, p);

}

curr\_count\_y++;

}

all\_Y[curr\_count\_y] = Y;

}

for (int curr\_count\_z = 0; curr\_count\_z < count\_z - 1; )

{

in >> Z >> Nz >> kz;

double hy;

if (kz == 1)

{

hy = (Z - all\_Z[curr\_count\_z]) / Nz;

for (int p = 1; p < Nz; p++)

{

all\_Z[curr\_count\_z + p] = all\_Z[curr\_count\_z] + hy \* p;

}

curr\_count\_z += Nz;

}

else

{

hy = (Z - all\_Z[curr\_count\_z]) \* (kz - 1) / (pow(kz, Nz) - 1);

for (int p = 0; p < Nz - 1; curr\_count\_z++, p++)

{

all\_Z[curr\_count\_z + 1] = all\_Z[curr\_count\_z] + hy \* pow(kz, p);

}

curr\_count\_z++;

}

all\_Z[curr\_count\_z] = Z;

}

in.close();

out.open("xyz.txt");

for (int i = 0; i < count\_z; i++)

{

for (int k = 0; k < count\_y; k++)

{

for (int j = 0; j < count\_x; j++)

{

out << all\_X[j] << "\t" << all\_Y[k] << "\t" << all\_Z[i] << "\n\n";

}

}

}

out.close();

// input area

out.open("elem.txt");

for (int i = 0; i < count\_z - 1; i++)

{

for (int k = 0; k < count\_y - 1; k++)

{

for (int j = 0; j < count\_x - 1; j++)

{

out << i\*count\_y\* count\_x + k \* count\_x + j << " " << i \* count\_y \* count\_x + k \* count\_x + j + 1 << " "

<< i \* count\_y \* count\_x + (k + 1) \* count\_x + j << " " << i \* count\_y \* count\_x + (k + 1) \* count\_x + j + 1 << " "

<< (i+1) \* count\_y \* count\_x + k \* count\_x + j << " " << (i + 1) \* count\_y \* count\_x + k \* count\_x + j + 1 << " "

<< (i + 1) \* count\_y \* count\_x + (k + 1) \* count\_x + j << " " << (i + 1) \* count\_y \* count\_x + (k + 1) \* count\_x + j + 1

<< " 0\n";

}

}

}

out.close();

out.open("info.txt");

out << count\_x \* count\_y \* count\_z << " 1 " << (count\_x - 1) \* (count\_y - 1) \* (count\_z - 1) << " 1 ";

out.close();

Create\_material\_file(path, lambda, sigma, hi);

out.open("S1.txt");

out << 2\*(count\_y \* count\_x + (count\_z - 2) \* count\_y + (count\_z - 2) \* (count\_x - 2)) << "\n";

// верх/низ

for (int k = 0; k < count\_y; k++)

{

for (int j = 0; j < count\_x; j++)

{

out << k \* count\_x + j << " " <<

(count\_z - 1) \* count\_y \* count\_x + k \* count\_x + j << " ";

}

}

// зад/перед

for (int i = 1; i < count\_z - 1; i++)

{

for (int k = 0; k < count\_y; k++)

{

out << i \* count\_y \* count\_x + k \* count\_x << " "

<< i \* count\_y \* count\_x + k \* count\_x + (count\_x - 1)

<< " ";

}

}

// право/лево

for (int i = 1; i < count\_z - 1; i++)

{

for (int j = 1; j < count\_x - 1; j++)

{

out << i \* count\_y \* count\_x + j << " "

<< i \* count\_y \* count\_x + (count\_y - 1) \* count\_x + j

<< " ";

}

}

}

***FEM.h***

#pragma once

#include <stdio.h>

#include <fstream>

#include <vector>

#include <string>

#include <set>

#include "LU\_solver.h"

#include "Solver.h"

using namespace std;

struct node

{

double x, y, z;

};

struct elem

{

vector<int> nodes;

int material;

};

struct material

{

double lambda, sigma, hi;

};

class FEM

{

vector<node> all\_nodes;

vector<elem> all\_elems;

vector<material> all\_materials;

vector<vector<int>> S1;

vector<vector<double>> M1, G1;

vector<double> b;

vector<int> ia, ja;

vector<double> al, au, di;

double omega = 1;

double func\_s(double x, double y, double z);

double func\_c(double x, double y, double z);

double func\_S\_s(double x, double y, double z, int s\_id);

double func\_S\_c(double x, double y, double z, int s\_id);

double func\_true\_s(double x, double y, double z, int s\_id);

double func\_true\_c(double x, double y, double z, int s\_id);

void Set\_const\_matrix();

int Get\_M\_Loc(double hx, double hy, double hz, double gam, vector<vector<double>>& M\_loc);

int Get\_G\_Loc(double hx, double hy, double hz, double lam, vector<vector<double>>& G\_loc);

int Get\_Loc(vector<vector<double>>& p\_loc, vector<vector<double>>& c\_loc, vector<double>& fc\_loc, vector<double>& fs\_loc, int id);

int GeneratePortrait();

int AddLocal(int el\_id, vector<vector<double>>& p\_loc, vector<vector<double>>& c\_loc, vector<double>& fc\_loc, vector<double>& fs\_loc);

int Create\_grid(string file);

int SetS1();

public:

vector<double> q0;

int Make\_SLAU(string file, double om = 1);

void Solve\_SLAU(bool LU);

double Calc\_error();

};

***FEM.cpp***

#include "FEM.h"

double FEM::func\_s(double x, double y, double z)

{

// один материал везде

// div grad u = 0

return -omega \* all\_materials[0].sigma\* func\_true\_c(x,y,z,0)- omega \* omega \* all\_materials[0].hi\* func\_true\_s(x, y, z, 0);

}

double FEM::func\_c(double x, double y, double z)

{

// один материал везде

// div grad u = 0

return omega \* all\_materials[0].sigma \* func\_true\_s(x, y, z, 0) - omega \* omega \* all\_materials[0].hi \* func\_true\_c(x, y, z, 0);

}

double FEM::func\_true\_c(double x, double y, double z, int id)

{

switch (id)

{

case 0:

return //3

2 \* x + y - 3 \* z

;

default:

break;

}

return 1;

}

double FEM::func\_true\_s(double x, double y, double z, int id)

{

switch (id)

{

case 0:

return //5

4 \* x + 5 \* y - 7 \* z

;

default:

break;

}

return 1;

}

double FEM::func\_S\_c(double x, double y, double z, int s\_id)

{

return func\_true\_c(x, y, z, s\_id);

}

double FEM::func\_S\_s(double x, double y, double z, int s\_id)

{

return func\_true\_s(x, y, z, s\_id);

}

int FEM::Create\_grid(string file)

{

int N, Nmat, Kel, NS1, NS;

std::ifstream in;

in.open("info.txt");

in >> N >> Nmat >> Kel >> NS1;

in.close();

in.open("xyz.txt");

all\_nodes.resize(N);

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

{

in >> all\_nodes[i].x >> all\_nodes[i].y >> all\_nodes[i].z;

}

in.close();

q0.resize(2\*N);

in.open("S1.txt");

S1.resize(NS1);

for (int i = 0; i < NS1; i++)

{

int size = 0;

in >> size;

S1[i].resize(size);

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

{

in >> S1[i][j];

}

}

in.close();

in.open("material.txt");

all\_materials.resize(Nmat);

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

{

in >> all\_materials[i].lambda >> all\_materials[i].sigma >> all\_materials[i].hi;

}

in.close();

in.open("elem.txt");

all\_elems.resize(Kel);

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

{

all\_elems[i].nodes.resize(8);

for (int j = 0; j < 8; j++)

{

in >> all\_elems[i].nodes[j];

}

in >> all\_elems[i].material;

}

in.close();

return 0;

}

void FEM::Set\_const\_matrix()

{

M1.resize(2);

G1.resize(2);

for (int i = 0; i < 2; i++)

{

M1[i].resize(2);

G1[i].resize(2);

}

M1[0][0] = 1. / 3;

M1[0][1] = 1. / 6;

M1[1][0] = 1. / 6;

M1[1][1] = 1. / 3;

G1[0][0] = 1;

G1[0][1] = -1;

G1[1][0] = -1;

G1[1][1] = 1;

}

int FEM::Get\_M\_Loc(double hx, double hy, double hz, double gam, vector<vector<double>>& M\_loc)

{

for (int i = 0; i < 8; i++)

{

int x\_i = (i) % 2,

y\_i = ((i) / 2) % 2,

z\_i = ((i) / 4);

//cout << i << "\t" << x\_i << y\_i << z\_i << endl;

for (int j = 0; j < 8; j++) //for (int j = 0; j < i; j++) M\_loc[j][i] = M\_loc[i][j];

{

int x\_j = (j ) % 2,

y\_j = ((j ) / 2) % 2,

z\_j = ((j ) / 4);

M\_loc[i][j] = /\*gam \*\*/ hx \* M1[x\_i][x\_j] \* hy \* M1[y\_i][y\_j] \* hz \* M1[z\_i][z\_j];

}

}

return 0;

}

int FEM::Get\_G\_Loc(double hx, double hy, double hz, double lam, vector<vector<double>>& G\_loc)

{

for (int i = 0; i < 8; i++)

{

int x\_i = (i) % 2,

y\_i = ((i) / 2) % 2,

z\_i = ((i) / 4);

for (int j = 0; j < 8; j++)

{

int x\_j = (j) % 2,

y\_j = ((j ) / 2) % 2,

z\_j = ((j ) / 4);

G\_loc[i][j] = lam \* (

hx \* G1[x\_i][x\_j] \* hy \* M1[y\_i][y\_j] \* hz \* M1[z\_i][z\_j] +

hy \* M1[x\_i][x\_j] \* hy \* G1[y\_i][y\_j] \* hz \* M1[z\_i][z\_j] +

hz \* M1[x\_i][x\_j] \* hy \* M1[y\_i][y\_j] \* hz \* G1[z\_i][z\_j]);

}

}

return 0;

}

int FEM::Get\_Loc(vector<vector<double>>& p\_loc, vector<vector<double>>& c\_loc, vector<double>& fc\_loc, vector<double>& fs\_loc, int id)

{

vector<vector<double>> M\_loc(8, vector<double>(8));

vector<vector<double>> G\_loc(8, vector<double>(8));

vector<double> f\_c(8), f\_s(8);

double

x1 = all\_nodes[all\_elems[id].nodes[0]].x,

x2 = all\_nodes[all\_elems[id].nodes[1]].x,

y1 = all\_nodes[all\_elems[id].nodes[0]].y,

y2 = all\_nodes[all\_elems[id].nodes[3]].y,

z1 = all\_nodes[all\_elems[id].nodes[0]].z,

z2 = all\_nodes[all\_elems[id].nodes[4]].z;

double

hx = x2 - x1,

hy = y2 - y1,

hz = z2 - z1;

Get\_M\_Loc(hx, hy, hz, 1, M\_loc);

Get\_G\_Loc(hx, hy, hz, all\_materials[all\_elems[id].material].lambda, G\_loc);

double

sigma = all\_materials[all\_elems[id].material].sigma,

hi = all\_materials[all\_elems[id].material].hi;

for (int i = 0; i < 8; i++)

{

node temp = all\_nodes[all\_elems[id].nodes[i]];

fc\_loc[i] = 0;

fs\_loc[i] = 0;

f\_c[i] = func\_c(temp.x, temp.y, temp.z);

f\_s[i] = func\_s(temp.x, temp.y, temp.z);

}

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 8; j++)

{

p\_loc[i][j] = G\_loc[i][j] - omega \* omega \* hi \* M\_loc[i][j];

c\_loc[i][j] = omega \* sigma \* M\_loc[i][j];

if (i != j)

{

p\_loc[j][i] = p\_loc[i][j];

c\_loc[j][i] = c\_loc[i][j];

}

fc\_loc[i] += f\_c[j] \* M\_loc[i][j];

fs\_loc[i] += f\_s[j] \* M\_loc[i][j];

}

}

return 0;

}

int FEM::AddLocal(int el\_id, vector<vector<double>>& p\_loc, vector<vector<double>>& c\_loc, vector<double>& fc\_loc, vector<double>& fs\_loc)

{

vector<int> global\_num(8);

for (int i = 0; i < 8; i++)

global\_num[i] = 2 \* all\_elems[el\_id].nodes[i];

for (int i = 0; i < 8; i++)

{

di[global\_num[i]] += p\_loc[i][i];

di[global\_num[i] + 1] += p\_loc[i][i];

int end0 = ia[global\_num[i] + 2] - 1;

int ind0 = end0;

al[ind0] += c\_loc[i][i]; //-c11

au[ind0] -= c\_loc[i][i];

// блоки нижнего треугольника

// верхняя строка блока

int beg = ia[global\_num[i]];

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

{

int end = ia[global\_num[i] + 1] - 1;

int ind = beg;

while (ja[ind] != global\_num[j])

{

ind++;

}

al[ind] += p\_loc[i][j]; //p21

al[ind + 1] -= c\_loc[i][j]; //-c21

}

// нижняя строка блока

int beg2 = ia[global\_num[i] + 1];

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

{

int end2 = ia[global\_num[i] + 2] - 1;

int ind2 = beg2;

while (ja[ind2] != global\_num[j])

{

ind2++;

}

al[ind2] += c\_loc[i][j]; //c21

al[ind2 + 1] += p\_loc[i][j]; //p21

}

// блоки верхнего треугольника

// левый столбец

beg = ia[global\_num[i]];

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

{

int end = ia[global\_num[i] + 1] - 1;

int ind = beg;

while (ja[ind] != global\_num[j])

{

ind++;

}

au[ind] += p\_loc[i][j]; // p21

au[ind + 1] += c\_loc[i][j]; // c21

}

// правый столбец

beg2 = ia[global\_num[i] + 1];

for (int j = 0; j < i; j++, beg++) {

int end2 = ia[global\_num[i] + 2] - 1;

int ind2 = beg2;

while (ja[ind2] != global\_num[j]) {

ind2++;

}

au[ind2] -= c\_loc[i][j]; // -c21

au[ind2 + 1] += p\_loc[i][j]; // p21

}

b[global\_num[i]] += fs\_loc[i];

b[global\_num[i] + 1] += fc\_loc[i];

}

return 0;

}

int FEM::GeneratePortrait()

{

int m = all\_nodes.size();

vector<set<int>> list(m);

for (int elem\_id = 0; elem\_id < all\_elems.size(); elem\_id++)

{

for (int i = 0; i < 8; i++)

{

for (int j = i + 1; j < 8; j++)

{

int ind1 = all\_elems[elem\_id].nodes[i];

int ind2 = all\_elems[elem\_id].nodes[j];

if (ind1 < ind2)

swap(ind1, ind2);

list[ind1].insert(ind2);

}

}

}

ia.resize(2\*m + 1);

//создание портрета по списку

ia[0] = 0;

ia[1] = 0;

ia[2] = 1;

for (int i = 1; i < m; i++)

{

ia[2 \* i + 1] = ia[2 \* i] + list[i].size() \* 2;

ia[2 \* (i + 1)] = ia[2 \* i + 1] + list[i].size() \* 2 + 1;

}

ja.resize(ia[2 \* m]);

au.resize(ia[2 \* m]);

al.resize(ia[2 \* m]);

di.resize(2 \* m);

b.resize(2 \* m);

for (int i = 1, k = 1; i < m; i++)

{

for (int j : list[i])

{

ja[k] = 2 \* j;

ja[k + 1] = 2 \* j + 1;

k += 2;

}

for (int j : list[i]) {

ja[k] = 2 \* j;

ja[k + 1] = 2 \* j + 1;

k += 2;

}

ja[k] = 2 \* i;

k++;

}

return 0;

}

int FEM::SetS1() // учет первых краевых

{

int NS1 = S1.size();

for (int i = 0; i < NS1; i++)

{

int s1\_id = i;

for (int j = 0; j < S1[i].size(); j++)

{

int node\_id = 2 \* S1[i][j],

node\_glob\_num = S1[i][j];

// first

di[node\_id] = 1;

b[node\_id] = func\_S\_s(all\_nodes[node\_glob\_num].x, all\_nodes[node\_glob\_num].y, all\_nodes[node\_glob\_num].z, s1\_id);

for (int k = ia[node\_id]; k < ia[node\_id + 1]; k++)

{

al[k] = 0;

}

for (int k = 0; k < ja.size(); k++)

{

if (ja[k] == node\_id)

{

au[k] = 0;

}

}

// second

node\_id++;

di[node\_id] = 1;

b[node\_id] = func\_S\_c(all\_nodes[node\_glob\_num].x, all\_nodes[node\_glob\_num].y, all\_nodes[node\_glob\_num].z, s1\_id);

for (int k = ia[node\_id]; k < ia[node\_id + 1]; k++)

{

al[k] = 0;

}

for (int k = 0; k < ja.size(); k++)

{

if (ja[k] == node\_id)

{

au[k] = 0;

}

}

}

}

return 0;

}

int FEM::Make\_SLAU(string file, double om)

{

omega = om;

Create\_grid("");

Set\_const\_matrix();

GeneratePortrait();

vector<vector<double>> p\_loc(8, vector<double>(8)),

c\_loc(8, vector<double>(8));

vector<double> fc\_loc(8), fs\_loc(8);

for (int i = 0; i < all\_elems.size(); i++)

{

Get\_Loc(p\_loc, c\_loc, fc\_loc, fs\_loc, i);

AddLocal(i, p\_loc, c\_loc, fc\_loc, fs\_loc);

}

SetS1();

return 0;

}

void FEM::Solve\_SLAU(bool LU)

{

if (!LU)

{

Solver t(ia, ja, di, al, au, b);

t.CGM\_LU();

//t.BSG();

t.getx0(q0);

}

else

{

LU\_solver t2(ia, ja, di, al, au, b);

t2.Solve\_task();

t2.getx0(q0);

}

}

double FEM::Calc\_error()

{

double res = 0, f\_s, f\_c, norm = 0;

for (int i = 0; i < all\_nodes.size(); i++)

{

f\_s = func\_true\_s(all\_nodes[i].x, all\_nodes[i].y, all\_nodes[i].z, 0);

f\_c = func\_true\_c(all\_nodes[i].x, all\_nodes[i].y, all\_nodes[i].z, 0);

res += (f\_s - q0[2 \* i]) \* (f\_s - q0[2 \* i]);

res += (f\_c - q0[2 \* i + 1]) \* (f\_c - q0[2 \* i + 1]);

norm += f\_s \* f\_s + f\_c \* f\_c;

}

return sqrt(res / norm);

}

***LU\_solver.h***

#pragma once

#include <iostream>

#include <fstream>

#include <vector>

#include <random>

using namespace std;

const double eps = 1E-20;

class LU\_solver

{

int N = 0; // Размер матрицы

vector<double> di; // Диагональ

vector<int> ia; // Портрет матрицы

vector<double> au; // верхний треугольник

vector<double> al; // нижний треугольник

vector<double> b; // Вектор

bool LU();

void revers();

void forward\_stroke(); // Прямой ход Ly = F

int input(string path);

void output();

public:

LU\_solver();

LU\_solver(string path);

LU\_solver(std::vector<int>& \_ia, std::vector<int>& \_ja, std::vector<double>& \_di,

std::vector<double>& \_al, std::vector<double>& \_au,

std::vector<double>& \_b);

int Solve\_task();

void getx0(vector<double> &q);

};

***LU\_solver.cpp***

#include "LU\_solver.h"

LU\_solver::LU\_solver()

{

}

LU\_solver::LU\_solver(string path)

{

input(path);

}

LU\_solver::LU\_solver(std::vector<int>& \_ia, std::vector<int>& \_ja, std::vector<double>& \_di,

std::vector<double>& \_al, std::vector<double>& \_au,

std::vector<double>& \_b)

{

N = \_di.size();

di = \_di;

b = \_b;

ia.resize(N + 1);

al.reserve(N \* 1000);

au.reserve(N \* 1000);

ia[0] = 0;

for (int i = 1; i < N; i++)

{

if (\_ia[i] != \_ia[i + 1])

{

int current = \_ja[\_ia[i]];

//end = i - current; // кол-во элементов в профиле

int k = \_ia[i];

for (; current < i; current++) // по всем элементам строки

{

if (k < \_ia[i+1] && \_ja[k] == current)

{

al.push\_back(\_al[k]);

au.push\_back(\_au[k]);

k++;

}

else

{

al.push\_back(0);

au.push\_back(0);

}

}

}

ia[i + 1] = al.size();

}

}

int LU\_solver::input(string path)

{

ifstream fin;

fin.open(path + "IN.txt");

fin >> N;

di.resize(N);

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

fin >> di[i];

ia.resize(N + 1);

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

fin >> ia[i];

au.resize(ia[N]);

for (int i = 0; i < ia[N]; i++)

fin >> au[i];

al.resize(ia[N]);

for (int i = 0; i < ia[N]; i++)

fin >> al[i];

b.resize(N);

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

fin >> b[i];

fin.close();

return 0;

}

void LU\_solver::output()

{

ofstream fout;

fout.open("out.txt");

fout.precision(2 \* sizeof(double) - 1);

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

fout << b[i] << "\n";

fout.close();

}

void LU\_solver::getx0(vector<double> &q)

{

if (q.size() != N)

q.resize(N);

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

q[i] = b[i];

}

void LU\_solver::forward\_stroke() // Прямой ход Ly = F

{

double res = 0;

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

{

int count = i - (ia[i + 1] - ia[i]);

res = 0;

for (int j = ia[i]; j < ia[i + 1]; j++, count++)

{

res += b[count] \* al[j];

}

b[i] = (b[i] - res) / di[i];

}

}

void LU\_solver::revers()

{

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

{

int j = i - (ia[i + 1] - ia[i]);

for (int k = ia[i]; k < ia[i + 1]; k++, j++)

{

b[j] -= au[k] \* b[i];

}

}

}

bool LU\_solver::LU()

{

for (int i = 1; i < N; i++)

{

int j0 = i - (ia[i + 1] - ia[i]);

for (int ii = ia[i]; ii < ia[i + 1]; ii++)

{

int j = ii - ia[i] + j0;

double sum\_l = 0, sum\_u = 0;

if (ia[j] < ia[j + 1])

{

int j0j = j - (ia[j + 1] - ia[j]);

int jjbeg = j0 < j0j ? j0j : j0; // max (j0, j0j)

int jjend = j < i - 1 ? j : i - 1; // min (j, i - 1)

for (int k = 0; k < jjend - jjbeg; k++)

{

int ind\_prev = ia[j] + jjbeg - j0j + k;

int ind\_now = ia[i] + jjbeg - j0 + k;

sum\_l += au[ind\_prev] \* al[ind\_now];

sum\_u += au[ind\_now] \* al[ind\_prev];

}

}

al[ii] -= sum\_l;

au[ii] -= sum\_u;

if (abs(di[j]) < eps) // matrix hasn't LU

{

cout << "di[" << j << "] = " << di[j] << endl;

return false;

}

au[ii] /= di[j];

di[i] -= al[ii] \* au[ii];

}

}

return true;

}

int LU\_solver::Solve\_task()

{

if (!LU())

{

cout << "The matrix has no LU decomposition" << endl;

return 1;

}

forward\_stroke();

revers();

return 0;

}

***Solver.h***

#pragma once

#include "MyMatrix.h"

class Solver

{

private:

MyMatrix A;

MyVector p,

z,

r,

s,

x0,

Ar,

y;

std::vector<double> L, D, U;

int N;

int maxIter;

double eps;

int iter;

double normR;

double normB;

public:

Solver(int size);

Solver(std::string filename);

Solver(MyMatrix \_A);

Solver(std::vector<int>& ia, std::vector<int>& ja, std::vector<double>& di, std::vector<double>& al, std::vector<double>& au,

std::vector<double>& b);

void CGM\_LU(); // conjugate gradient method whith LU factorization

void LOS\_LU(); // locally optimal scheme whith LU factorization

void FactLU(std::vector<double>& L, std::vector<double>& U, std::vector<double>& D);

void Direct(std::vector<double>& L, std::vector<double>& D, MyVector& y, MyVector& b);

void Direct(std::vector<double>& L, MyVector& y, MyVector& b);

void Reverse(std::vector<double>& U, MyVector& x, MyVector& y);

void Reverse(std::vector<double>& U, std::vector<double>& D, MyVector& x, MyVector& y);

void output(std::string filename);

void getx0(std::vector<double>& x);

};

***Solver.cpp***

#include "Solver.h"

Solver::Solver(int size)

{

N = size;

A.di.resize(N);

A.ia.resize(N + 1);

A.ja.resize((N \* N - N) / 2);

A.au.resize((N \* N - N) / 2);

A.al.resize((N \* N - N) / 2);

A.b.Size(N);

A.N = N;

A.ia[0] = 0;

A.ia[1] = 0;

A.di[0] = 1;

A.b.vect[0] += 1;

for (int i = 1; i < N; i++)

{

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

{

A.au[A.ia[i] + j] = 1. / (i + j + 1); // т.к. (i + 1) + (i - k + 1 + j) - 1, k = i

A.b.vect[j] += (i + 1) \* A.au[A.ia[i] + j];

A.al[A.ia[i] + j] = 1. / (i + j + 1);

A.b.vect[i] += (j + 1) \* A.al[A.ia[i] + j];

A.ja[A.ia[i] + j] = j;

}

A.ia[i + 1] = A.ia[i] + i;

A.di[i] = 1. / (i + i + 1);

A.b.vect[i] += (i + 1) \* A.di[i];

}

maxIter = 10000;

eps = 1E-14;

std::ofstream fout;

fout.precision(16);

fout.open("kuslau.txt");

fout << N << " " << maxIter << " " << eps;

fout.close();

fout.open("di.txt");

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

fout << A.di[i] << " ";

fout.close();

fout.open("ig.txt");

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

fout << A.ia[i] << " ";

fout.close();

fout.open("jg.txt");

for (int i = 0; i < A.ia[N]; i++)

fout << A.ja[i] << " ";

fout.close();

fout.open("ggu.txt");

for (int i = 0; i < A.ia[N]; i++)

fout << A.au[i] << " ";

fout.close();

fout.open("ggl.txt");

for (int i = 0; i < A.ia[N]; i++)

fout << A.al[i] << " ";

fout.close();

fout.open("pr.txt");

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

fout << A.b.vect[i] << " ";

fout.close();

x0.Size(N);

r.Size(N);

z.Size(N);

p.Size(N);

Ar.Size(N);

y.Size(N);

L.resize(A.ia[N]);

D.resize(N);

U.resize(A.ia[N]);

normB = A.b.Norm();

iter = 0;

normR = 0;

}

Solver::Solver(std::string filename)

{

std::ifstream in("kuslau.txt");

in >> N >> maxIter >> eps;

A.ReadMatrix(N);

x0.Size(N);

r.Size(N);

z.Size(N);

p.Size(N);

Ar.Size(N);

y.Size(N);

L.resize(A.ia[N]);

D.resize(N);

U.resize(A.ia[N]);

normB = A.b.Norm();

iter = 0;

normR = 0;

}

Solver::Solver(std::vector<int> &ia, std::vector<int>& ja, std::vector<double>& di, std::vector<double>& al, std::vector<double>& au,

std::vector<double>& b)

{

N = di.size();

maxIter = 500;

eps = 1E-15;

//A = \_A;

A.al = al;

A.au = au;

A.b.vect = b;

A.di = di;

A.ia = ia;

A.ja = ja;

A.N = N;

x0.Size(N);

r.Size(N);

z.Size(N);

p.Size(N);

Ar.Size(N);

y.Size(N);

L.resize(A.ia[N]);

D.resize(N);

U.resize(A.ia[N]);

normB = A.b.Norm();

iter = 0;

normR = 0;

}

Solver::Solver(MyMatrix \_A)

{

N = \_A.N;

maxIter = 10000;

eps = 1E-15;

A = \_A;

x0.Size(N);

r.Size(N);

z.Size(N);

p.Size(N);

Ar.Size(N);

y.Size(N);

L.resize(A.ia[N]);

D.resize(N);

U.resize(A.ia[N]);

normB = A.b.Norm();

iter = 0;

normR = 0;

}

void Solver::output(std::string filename)

{

std::ofstream out(filename);

out.imbue(std::locale("Russian"));

out.precision(15);

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

out << x0.vect[i] << std::endl;

}

void Solver::getx0(std::vector<double>& x)

{

if (x.size() != N)

x.resize(N);

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

x[i] = x0.vect[i];

}

void Solver::CGM\_LU()

{

std::cout.precision(15);

FactLU(L, U, D);

double r\_r = 0, Az\_z = 0;

double a = 0, B = 0;

A.Ax(x0, r); // r0 = A\*x0

A.b - r; // r0 = B - A\*x0

Direct(L, D, r, r); // r0 = L^(-1) \* (B - A\*x0)

Reverse(L, D, r, r); // r0 = L^(-T) \* L^(-1) \* (B - A\*x0)

A.ATx(r, y); // y0 = A^(T) \* L^(-T) \* L^(-1) \* (B - A\*x0)

Direct(U, r, y); // r0 = U-t \* A^(T) \* L^(-T) \* L^(-1) \* (B - A\*x0)

z = r; // z0 = r0

r\_r = r \* r;

normR = sqrt(r\_r) / normB;

for (iter = 1; iter < maxIter + 1 && normR >= eps; iter++)

{

Reverse(U, y, z); // y = U^(-1) \* z

A.Ax(y, p); // p = A \* U^(-1) \* z

Direct(L, D, p, p); // p = L-1 \* A \* U^(-1) \* z

Reverse(L, D, p, p); // p = L-t \* p

A.ATx(p, Ar); // Ar = At \* P

Direct(U, Ar, Ar); // Ar = U-t \* Ar

Az\_z = Ar \* z; // (Ar,z)

a = r\_r / Az\_z;

// x(k) = x(k-1) + z(k-1)\*a(k-1)

// r(k) = r(k-1) - AT\*A\*z(k-1)\*a(k-1)

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

{

x0.vect[i] = x0.vect[i] + z.vect[i] \* a;

r.vect[i] = r.vect[i] - Ar.vect[i] \* a;

}

// B(k) = (r(k), r(k)) / (r(k-1), r(k-1))

B = 1.0 / r\_r;

r\_r = r \* r;

B \*= r\_r;

// z(k) = r(k) + B(k)\*z(k-1)

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

{

z.vect[i] = r.vect[i] + z.vect[i] \* B;

}

normR = sqrt(r\_r) / normB;

//std::cout << iter << ". " << normR << std::endl;

}

// x0 = U^(-1) \* x0

Reverse(U, x0, x0);

}

void Solver::LOS\_LU()

{

std::cout.precision(15);

FactLU(L, U, D);

double p\_p = 0, p\_r = 0, r\_r = 0, Ar\_p = 0;

double a = 0, B = 0, eps2 = 1e-10;

A.Ax(x0, y); // y = A \* x0

A.b - y; // y = B - A \* x0

Direct(L, D, r, y); // r0 = L^(-1) \* (B - A \* x0)

Reverse(U, z, r); // z0 = U^(-1) \* r0

A.Ax(z, y); // y = A \* z0

Direct(L, D, p, y); // p0 = L^(-1) \* (A \* z0)

r\_r = r \* r;

normR = sqrt(r\_r) / normB;

for (iter = 1; iter < maxIter + 1 && normR >= eps; iter++)

{

p\_p = p \* p;

p\_r = p \* r;

a = p\_r / p\_p;

// x(k) = x(k-1) + a(k) \* z(k-1)

// r(k) = r(k-1) - a(k) \* p(k-1)

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

{

x0.vect[i] = x0.vect[i] + z.vect[i] \* a;

r.vect[i] = r.vect[i] - p.vect[i] \* a;

}

Reverse(U, y, r); // y = U^(-1) \* r(k)

A.Ax(y, Ar); // Ar = A \* U^(-1) \* r(k)

Direct(L, D, Ar, Ar); // Ar = L^(-1) \* A \* U^(-1) \* r(k)

Ar\_p = Ar \* p; // (Ar, p)

B = -(Ar\_p / p\_p);

// z(k) = U^(-1) \* r(k) + B(k) \* z(k-1)

// p(k) = L^(-1) \* A \* U^(-1) \* r(k) + B(k) \* p(k-1)

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

{

z.vect[i] = y.vect[i] + z.vect[i] \* B;

p.vect[i] = Ar.vect[i] + p.vect[i] \* B;

}

if (r\_r - (r\_r - a \* a \* p\_p) < eps2)

r\_r = r \* r;

else

r\_r = r\_r - a \* a \* p\_p;

normR = sqrt(r\_r) / normB;

std::cout << iter << ". " << normR << std::endl;

}

}

void Solver::BSG()

{

std::cout.precision(15);

double s\_Az = 0, p\_r = 0, r\_r = 0, p\_r\_prev = 0;

double a = 0, B = 0, eps2 = 1e-10;

A.Ax(x0, y); // y = A \* x0

A.b - y; // y = B - A \* x0

r = y;

p = r;

z = r;

s = r;

r\_r = r \* r;

p\_r = r\_r;

normR = sqrt(r\_r) / normB;

for (iter = 1; iter < maxIter + 1 && normR >= eps; iter++)

{

A.Ax(z, y); // y = A \* z

s\_Az = s \* y;

a = p\_r / s\_Az;

// x(k) = x(k-1) + a(k) \* z(k-1)

// r(k) = r(k-1) - a(k) \* Az(k-1)

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

{

x0.vect[i] = x0.vect[i] + z.vect[i] \* a;

r.vect[i] = r.vect[i] - y.vect[i] \* a;

}

A.ATx(s, y); // y = AT \* s

// p(k) = p(k-1) - a(k) \* ATs(k-1)

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

{

p.vect[i] = p.vect[i] - y.vect[i] \* a;

}

p\_r\_prev = p\_r;

p\_r = p \* r;

B = p\_r / p\_r\_prev;

// z(k) = r(k) + B(k) \* z(k-1)

// s(k) = p(k) + B(k) \* s(k-1)

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

{

z.vect[i] = r.vect[i] + z.vect[i] \* B;

s.vect[i] = p.vect[i] + s.vect[i] \* B;

}

r\_r = r \* r;

normR = sqrt(r\_r) / normB;

std::cout << iter << ". " << normR << std::endl;

}

}

void Solver::FactLU(std::vector<double>& L, std::vector<double>& U, std::vector<double>& D)

{

L = A.al;

U = A.au;

D = A.di;

double l, u, d;

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

{

d = 0;

int i0 = A.ia[k], i1 = A.ia[k + 1];

int i = i0;

for (; i0 < i1; i0++)

{

l = 0;

u = 0;

int j0 = i, j1 = i0;

for (; j0 < j1; j0++)

{

int t0 = A.ia[A.ja[i0]], t1 = A.ia[A.ja[i0] + 1];

for (; t0 < t1; t0++)

{

if (A.ja[j0] == A.ja[t0])

{

l += L[j0] \* U[t0];

u += L[t0] \* U[j0];

}

}

}

L[i0] -= l;

U[i0] -= u;

U[i0] /= D[A.ja[i0]];

d += L[i0] \* U[i0];

}

D[k] -= d;

}

}

// L\*y = B

void Solver::Direct(std::vector<double>& L, std::vector<double>& D, MyVector& y, MyVector& b)

{

y = b;

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

{

double sum = 0;

int k0 = A.ia[i], k1 = A.ia[i + 1];

int j;

for (; k0 < k1; k0++)

{

j = A.ja[k0];

sum += y.vect[j] \* L[k0];

}

double buf = y.vect[i] - sum;

y.vect[i] = buf / D[i];

}

}

// U^(T)\*y = B

void Solver::Direct(std::vector<double>& L, MyVector& y, MyVector& b)

{

y = b;

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

{

double sum = 0;

int k0 = A.ia[i], k1 = A.ia[i + 1];

int j;

for (; k0 < k1; k0++)

{

j = A.ja[k0];

sum += y.vect[j] \* L[k0];

}

y.vect[i] -= sum;

}

}

// U\*x = y

void Solver::Reverse(std::vector<double>& U, MyVector& x, MyVector& y)

{

x = y;

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

{

int k0 = A.ia[i], k1 = A.ia[i + 1];

int j;

for (; k0 < k1; k0++)

{

j = A.ja[k0];

x.vect[j] -= x.vect[i] \* U[k0];

}

}

}

// L^(T)\*x = y

void Solver::Reverse(std::vector<double>& U, std::vector<double>& D, MyVector& x, MyVector& y)

{

x = y;

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

{

int k0 = A.ia[i], k1 = A.ia[i + 1];

int j;

x.vect[i] /= D[i];

for (; k0 < k1; k0++)

{

j = A.ja[k0];

x.vect[j] -= x.vect[i] \* U[k0];

}

}

}

***MyMatrix.h***

#pragma once

#include "MyVector.h"

class MyMatrix

{

public:

std::vector<double> di;

std::vector<double> al;

std::vector<double> au;

MyVector b;

std::vector<int> ja;

std::vector<int> ia;

int N;

MyMatrix(void);

void ReadMatrix(int size);

void Ax(std::vector<double>& X, std::vector<double>& Y); // умножение матрицы на вектор y = Ax

void Ax(MyVector& X, MyVector& Y); // умножение матрицы на вектор y = Ax

void ATx(MyVector& X, MyVector& Y); // умножение транспонированной матрицы на вектор y = A^T\*x

MyMatrix& operator + (MyMatrix B);

MyMatrix operator \* (const double a);

MyMatrix& operator=(const MyMatrix& B);

};

***MyMatrix.cpp***

#include "MyMatrix.h"

MyMatrix::MyMatrix(void)

{

}

void MyMatrix::ReadMatrix(int size)

{

N = size;

std::ifstream in;

in.open("ig.txt");

ia.resize(N + 1);

for (int i = 0; i < N + 1; i++)

{

in >> ia[i];

}

in.close();

if (ia[0])

for (int i = 0; i < N + 1; i++)

{

ia[i]--;

}

in.open("jg.txt");

ja.resize(ia[N]);

for (int i = 0; i < ia[N]; i++)

{

in >> ja[i];

}

in.close();

if (ja[0])

for (int i = 0; i < ia[N]; i++)

{

ja[i]--;

}

in.open("di.txt");

di.resize(N);

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

{

in >> di[i];

}

in.close();

in.open("ggu.txt");

au.resize(ia[N]);

for (int i = 0; i < ia[N]; i++)

{

in >> au[i];

}

in.close();

in.open("ggl.txt");

al.resize(ia[N]);

for (int i = 0; i < ia[N]; i++)

{

in >> al[i];

}

in.close();

b.Size(N);

b.ReadVector("pr.txt");

}

// y = Ax

void MyMatrix::Ax(MyVector& x, MyVector& y)

{

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

{

y.vect[i] = di[i] \* x.vect[i];

for (int j = ia[i]; j < ia[i + 1]; j++)

{

int k = ja[j];

y.vect[i] += al[j] \* x.vect[k];

y.vect[k] += au[j] \* x.vect[i];

}

}

}

// y = Ax

void MyMatrix::Ax(std::vector<double>& x, std::vector<double>& y)

{

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

{

y[i] = di[i] \* x[i];

for (int j = ia[i]; j < ia[i + 1]; j++)

{

int k = ja[j];

y[i] += al[j] \* x[k];

y[k] += au[j] \* x[i];

}

}

}

// y = A^(T)x

void MyMatrix::ATx(MyVector& x, MyVector& y)

{

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

{

y.vect[i] = di[i] \* x.vect[i];

for (int j = ia[i]; j < ia[i + 1]; j++)

{

int k = ja[j];

y.vect[i] += au[j] \* x.vect[k];

y.vect[k] += al[j] \* x.vect[i];

}

}

}

MyMatrix& MyMatrix::operator+ (MyMatrix B)

{

if (N != B.N)

{

std::cout << "A и B разного размера\n";

return \*this;

}

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

{

this->di[i] += B.di[i];

for (int j = ia[i]; j < ia[i + 1]; j++)

{

int k = ja[j];

if (k != B.ja[j])

{

std::cout << "A и B имеют разные портреты\n";

return \*this;

}

this->al[j] += B.al[j];

this->au[j] += B.au[j];

}

}

return \*this;

}

MyMatrix MyMatrix::operator\* (const double a)

{

MyMatrix C = \*this;

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

{

C.di[i] \*= a;

for (int j = ia[i]; j < ia[i + 1]; j++)

{

C.al[j] \*= a;

C.au[j] \*= a;

}

}

return C;

}

MyMatrix& MyMatrix::operator=(const MyMatrix& B)

{

if (this != &B)

{

this->al = B.al;

this->au = B.au;

this->b = B.b;

this->di = B.di;

this->ia = B.ia;

this->ja = B.ja;

this->N = B.N;

}

return \*this;

}

***MyVector.h***

#pragma once

#include <vector>

#include <iostream>

#include <math.h>

#include <fstream>

class MyVector

{

public:

std::vector<double> vect;

MyVector();

void Size(int N);

void ReadVector(std::string filename);

MyVector& operator=(const MyVector& a);

double operator\* (const MyVector& a);

MyVector& operator\* (const double a);

MyVector& operator-(MyVector& a);

MyVector& operator+(MyVector& a);

double Norm();

};

***MyVector.cpp***

#include "MyVector.h"

MyVector::MyVector()

{

}

void MyVector::Size(int N)

{

vect.resize(N);

}

// read from filename

void MyVector::ReadVector(std::string filename)

{

if (vect.size() < 1)

return;

std::ifstream in(filename);

for (int i = 0; i < vect.size(); i++)

{

in >> vect[i];

}

in.close();

}

// this = a; a = a

MyVector& MyVector::operator=(const MyVector& a)

{

if (this != &a)

this->vect = a.vect;

return \*this;

}

// this = a \* this

MyVector& MyVector::operator\*(const double a)

{

for (int i = 0; i < this->vect.size(); i++)

this->vect[i] \*= a;

return \*this;

}

// (this, a)

double MyVector::operator\* (const MyVector& a)

{

double res = 0;

if (this->vect.size() != a.vect.size())

return res;

for (int i = 0; i < this->vect.size(); i++)

res += this->vect[i] \* a.vect[i];

return res;

}

// a = this - a;

MyVector& MyVector::operator-(MyVector& a)

{

if (this->vect.size() != a.vect.size())

{

return \*this;

}

else

{

for (int i = 0; i < this->vect.size(); i++)

{

a.vect[i] = this->vect[i] - a.vect[i];

}

return a;

}

}

// this = this + a;

MyVector& MyVector::operator+(MyVector& a)

{

if (this->vect.size() != a.vect.size())

{

return \*this;

}

else

{

for (int i = 0; i < this->vect.size(); i++)

{

this->vect[i] = this->vect[i] + a.vect[i];

}

return \*this;

}

}

// || this ||

double MyVector::Norm()

{

return sqrt((\*this) \* (\*this));

}

***Source.cpp***

#include "FEM.h"

#include "Generate.h"

#include <time.h>

int main()

{

cout << "start make file grid\n";

Create\_grid\_from\_points("");

cout << "end make file grid\n";

ofstream fileLU("result\_LU.txt"), fileIter("result\_LOS.txt"), file("table.txt");

fileLU.imbue((locale)"Rus");

fileIter.imbue((locale)"Rus");

int iter = 400;

// one task

FEM task;

task.Make\_SLAU("");

clock\_t start = clock();

task.Solve\_SLAU(true);

clock\_t end = clock();

file << task.Calc\_error() << "\t" << (double)(end - start) / CLOCKS\_PER\_SEC << "\t";

for (int i = 0; i < task.q0.size(); i++)

{

fileLU << task.q0[i] << "\n";

}

start = clock();

task.Solve\_SLAU(false);

end = clock();

file << task.Calc\_error() << "\t" << (double)(end - start) / CLOCKS\_PER\_SEC << "\n";

for (int i = 0; i < task.q0.size(); i++)

{

fileIter << task.q0[i] << "\n";

}

// tasks with different params

/\*for (double om = 1E-3; om < 1E+10; om \*= 1000)

{

for (double sig = 0; sig < 1E+8; sig \*= 100)

{

for (double lam = 100; lam < 800001; lam \*= 10)

{

for (double hhi = 8.81E-11; hhi < 1E-8; hhi \*= 10)

{

FEM task;

Create\_material\_file("", lam, sig, hhi);

cout << "start make SLAU\n";

task.Make\_SLAU("", om);

cout << "end make SLAU\n";

file << om << "\t" << sig << "\t" << lam << "\t" << hhi << "\t";

clock\_t start = clock();

task.Solve\_SLAU(true);

clock\_t end = clock();

file << task.Calc\_error() << "\t" << (double)(end - start) / CLOCKS\_PER\_SEC << "\t";

for (int i = 0; i < task.q0.size(); i++)

{

fileLU << task.q0[i] << "\n";

}

start = clock();

task.Solve\_SLAU(false);

end = clock();

file << task.Calc\_error() << "\t" << (double)(end - start) / CLOCKS\_PER\_SEC << "\n";

for (int i = 0; i < task.q0.size(); i++)

{

fileIter << task.q0[i] << "\n";

}

cout << "\n--------------------------------\nNEED PARAM ITER" << iter << "\n";

iter--;

if (hhi == 8.81E-11)

{

hhi = 1E-12;

}

}

if (lam == 100000)

{

lam = 800000;

}

}

if (!sig)

sig = 1;

}

}\*/

return 0;

}