**mains.cpp**

#include <iostream>

#include <cmath>

#include <fstream>

#include "functionSS.h"

using namespace std;

int main() {

ofstream out\_ichi;

ofstream out\_ni;

ofstream out\_san;

out\_ichi.open("first\_out.txt");

out\_ni.open("second\_out.txt");

out\_san.open("third\_out.txt");

int i = 0;

int graph\_number = 0;

start\_values();

while( r[N] < 100.0 ){

i = i + 1;

//cout << r[N] << " " << i << endl;

u[N] = u\_function(u[N], 0.0, P[N - 1], r[N]);

for (int j = 0; j < N; j++) {

if (j < N - 1) {

u[j + 1] = u\_function(u[j + 1], P[j + 1], P[ j ], r[j + 1]);

}

r[j + 1] = r\_function(r[j + 1], u[j + 1]);

double V\_old = V[j];

V[j] = V\_function(r[j + 1], r[j]);

E[j] = E\_function(E[j], V\_old, V[j]);

P[j] = P\_function(E[j], V[j]);

//cout << u[j] << V[j] << E[j] << P[j] << endl;

}

//cout << r[N] << "\n";

if (i % step == 0) {

graph\_number++;

//cout << r[N] << graph\_number << endl;

for (int k = 0; k < N + 1; k++) {

if (graph\_number == 1) {

out\_ichi << u[k] << " " << 1.0 / V[k] << " " << E[k] << " " << r[k] << endl;

}

if (graph\_number == 2) {

out\_ni << u[k] << " " << 1.0 / V[k] << " " << E[k] << " " << r[k] << endl;

}

if (graph\_number == 3) {

out\_san << u[k] << " " << 1.0 / V[k] << " " << E[k] << " " << r[k] << endl;

}

}

}

}

out\_ichi.close();

out\_ni.close();

out\_san.close();

//system("pause");

return 0;

}

**FunctionSS.h**

#ifndef functionSS

#define functionSS

#include <iostream>

#include <cmath>

#include <fstream>

using namespace std;

#define N 4

#define MM 400000

#define step (unsigned)int(MM / 3.0)

#define delta\_t 1.0e-8

#define delta\_m double( 1.0 / N )

#define gamma 5.0 / 3.0

#define T0 3500.0

#define E0 3.15 \* 10e9

#define ro0 5.45 \* 10e-4

#define P0 1.15 \* 10e6

#define V0 1.0 / ro0

#define Rx 100.0

#define Ux 5.6 \* 10e4

#define tx 0.847\*10e-3

#define rox (1 / 3.0 \* ro0)

#define mx (rox \* (Rx \*\* 3) / 3.0)

#define Px ((1.0 / 3) \* ro0 \* E0)

#define Ex 3.15 \* 10e9

double u[N + 1];

double P[N + 1];

double Rho[N + 1];

double E[N + 1];

double r[N + 1];

double V[N + 1];

void start\_values() {

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

u[i] = 0.0;

}

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

r[0] = 0.0;

r[i] = pow(( pow(r[i - 1], 3) + delta\_m ), 1.0 / 3.0);

}

//for (int i = 0; i < N + 1; i++) { cout << r[i] << " " << i << endl; }

//system("pause");

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

Rho[i] = 3.0;

V[i] = 1.0 / Rho[i];

}

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

P[i] = Rho[i] \* (gamma - 1.0);

}

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

E[i] = 1.0;

}

}

double u\_function( double u, double P1, double P2, double r) {

return u - delta\_t / delta\_m \* pow(r, 2) \* (P1 - P2);

}

double r\_function(double r, double u) {

return r + u \* delta\_t;

}

double P\_function(double E, double V) {

return (gamma - 1.0) \* E / V;

}

double E\_function(double E, double V1, double V2) {

return E \* V2 / V1 \* (V1 + (gamma - 1.0) / 2.0 \* (V1 - V2)) / (V2 - (gamma - 1.0) / 2.0 \* (V1 - V2));

}

double V\_function(double r1, double r2) {

return (pow(r1, 3) - pow(r2, 3)) / 3.0 / delta\_m;

}

#endif