ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Отчёт

Курсовой проект

«Решение математической модели SEIR-D для Новосибирской Области»

Выполнил студент Никифоров Иван Владимирович

Ф.И.О.

Группы ИВ-122

Новосибирск – 2023

Постановка задачи

Решение системы уравнений модели SEIR-D для Новосибирской области с коэффициентами из статьи «Математическое моделирование и прогнозирование COVID-19 в Москве и Новосибирской области».
Использовать метод Эйлера на участке от нулевого до девяностого дня с точностью 2 знака после запятой.

Теория

Метод Эйлера — простейший численный метод решения систем обыкновенных дифференциальных уравнений. Впервые описан Леонардом Эйлером в 1768 году. Метод Эйлера является явным, одношаговым методом первого порядка точности.

Классический метод Эйлера не подходит для решения поставленной задачи из-за того, что он имеет первый порядок точности, что недопустимо в ходе выполнения работы, но есть модифицированный метод Эйлера, который имеет второй порядок точности.
Повысить точность и устойчивость вычисления можно с помощью явного метода Эйлера следующего вида

Прогноз:

$$\widetilde{y_i} = y_{i-1} + (x_i - x_{i-1}) f(x_{i-1}, y_{i-1})$$

Коррекция

$$\widetilde{y_i} = y_{i-1} + (x_i - x_{i-1}) \frac{f(x_{i-1}, y_{i-1}) + f(x_i, \widetilde{y_i})}{2}$$

Этот метод подходит для решения данной задачи.

Выполнение

Перед реализацией функций стоит обозначить в программе все данные константы из таблицы 11 в статье:

Модель	α_E	α_I	κ	ρ	β	ν	ε_{CH}	μ	c^{isol}	E_0	R_0
SEIR-HCD	0.001	0.224	0.108	–	0.013	0.006	0.055	0.072	–	1001	–
SEIR-D	0.999	0.999	0.042	0.952	0.999	–	–	0.0188	0	99	24

Таблица 11 из статьи

Перенес данные в программу:

```
const double alpha_I = 0.999, alpha_E = 0.999, k = 0.042, ro = 0.952, beta = 0.999,
    mu = 0.0188, c_isol = 0, E_0 = 99, R_0 = 24, g = 0;
```

Константы в реализации программы

После чего были реализованы основные функции из системы уравнений в статье:

```
double D_function(double I){
    double answer = mu * I;
    return answer;
}

double I_function(double E, double I){
    double answer = (k * E) - (beta * I) - (mu * I);
    return answer;
}

double R_function(double E, double I, double R){
    double answer = (beta * I) + (ro * E) - (g * R);
    return answer;
}

double E_function(double S, double E, double I, double N){
   double answer = (1 * S) * ((alpha_I * I) / N + (alpha_E * E) / N) - (k + ro) * E;
   return answer;
}

double S_function(double S, double E, double I, double R, double N){
    double answer = (-1 * S) * ((alpha_I * I) / N + (alpha_E * E) / N) + g * R;
    return answer;
}
```

Реализация основных функций в программе

Также были реализованы дополнительные функции корректировки, упрощающие использование основных функций в вычислении данных:

```
double I_with_h(double E, double I, double h){
    double answer = h * I_function(E, I + h / 2 * I_function(E, I));
    return answer;
}

double R_with_h(double E, double I, double R, double h){
    double answer = h * R_function(E, I, R + h / 2 * R_function(E, I, R));
    return answer;
}

double E_with_h(double S, double E, double I, double N, double h){
    double answer = h * E_function(S, (E + h) / (2 * E_function(S, E, I, N)), I, N);
    return answer;
}

double S_with_h(double S, double E, double I, double R, double N, double h){
    double answer = h * S_function((S + h) / (2 * S_function(S, E, I, R, N)), E, I, R, N);
    return answer;
}
```

Реализация упрощающих функций в программе

Осталось лишь применить данные функции в основной, для этого обозначаем границы цикла, в нашем случае это количество дней от нуля до девяноста, после чего внутри цикла начинаем подсчет данных и вывод в терминал:

```
for(int i = 1; i < n; i++){
    t[i] = t_0 + i * h;
    D[i] = D[i - 1] + D_function(I[i - 1]);
    I[i] = I[i - 1] + I_with_h(E[i - 1], I[i - 1], h);
    R[i] = R[i - 1] + R_with_h(E[i - 1], I[i - 1], R[i - 1], h);
    E[i] = E[i - 1] + E_with_h(S[i - 1], E[i - 1], I[i - 1], N[i - 1], h);
    S[i] = S[i - 1] + S_with_h(S[i - 1], E[i - 1], I[i - 1], R[i - 1], N[i - 1], h);
    N[i] = D[i] + I[i] + R[i] + E[i] + S[i];
    printf("N[%d] = %.2lf S[%d] = %.2lf E[%d] = %.2lf I[%d] = %.2lf R[%d] = %.2lf D[%
}
```

Подсчет всех данных и их вывод в терминал

Также реализован блок, внутри которого выводятся все данные в текстовые файлы для упрощенного построения графиков с помощью утилиты gnuplot:

```
FILE* file_D = fopen("Data/D.txt", "w");
FILE* file_I = fopen("Data/I.txt", "w");
FILE* file_R = fopen("Data/R.txt", "w");
FILE* file_E = fopen("Data/E.txt", "w");
FILE* file_S = fopen("Data/S.txt", "w");
FILE* file_N = fopen("Data/N.txt", "w");
```

Блок для вывода данных в текстовый файл. Часть 1.

```c
for(int i = 0; i < n; i++) {
    fprintf(file_D, "%.0lf %.2lf\n", t[i], D[i]);
    fprintf(file_I, "%.0lf %.2lf\n", t[i], I[i]);
    fprintf(file_R, "%.0lf %.2lf\n", t[i], R[i]);
    fprintf(file_E, "%.0lf %.2lf\n", t[i], E[i]);
    fprintf(file_S, "%.0lf %.2lf\n", t[i], S[i]);
    fprintf(file_N, "%.0lf %.2lf\n", t[i], N[i]);
}
fclose(file_D);
fclose(file_I);
fclose(file_R);
fclose(file_E);
fclose(file_S);
fclose(file_N);
```

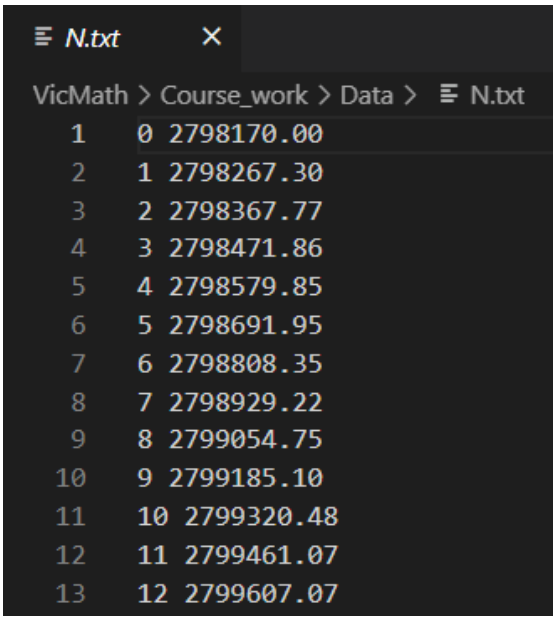
Блок для вывода данных в текстовый файл. Часть 2.

Демонстрация работы программы

По окончанию работы программы в терминале выведены все данные, а также созданы несколько текстовых файлов с теми же данными:

```
ivan@DESKTOP-Q2TPQHF:~/VicMath/Course_work$ ./main
N[0] = 2798170.00 S[0] = 2798047.00 E[0] = 99.00 I[0] = 0.00 R[0] = 24.00 D[0] = 0.00
N[1] = 2798267.30       S[1] = 2798047.50       E[1] = 99.51    I[1] = 2.04     R[1] = 118.25   D[1] = 0.00
N[2] = 2798367.77       S[2] = 2798048.00       E[2] = 101.64   I[2] = 3.07     R[2] = 215.02   D[2] = 0.04
N[3] = 2798471.86       S[3] = 2798048.50       E[3] = 104.78   I[3] = 3.63     R[3] = 314.85   D[3] = 0.10
N[4] = 2798579.85       S[4] = 2798049.00       E[4] = 108.47   I[4] = 3.98     R[4] = 418.24   D[4] = 0.16
N[5] = 2798691.95       S[5] = 2798049.50       E[5] = 112.51   I[5] = 4.23     R[5] = 525.48   D[5] = 0.24
N[6] = 2798808.35       S[6] = 2798050.00       E[6] = 116.79   I[6] = 4.43     R[6] = 636.81   D[6] = 0.32
N[7] = 2798929.22       S[7] = 2798050.50       E[7] = 121.27   I[7] = 4.63     R[7] = 752.42   D[7] = 0.40
N[8] = 2799054.75       S[8] = 2798051.00       E[8] = 125.95   I[8] = 4.82     R[8] = 872.49   D[8] = 0.49
N[9] = 2799185.10       S[9] = 2798051.50       E[9] = 130.81   I[9] = 5.01     R[9] = 997.21   D[9] = 0.58
N[10] = 2799320.48      S[10] = 2798052.00      E[10] = 135.87  I[10] = 5.20    R[10] = 1126.74         D[10] = 0.67
N[11] = 2799461.07      S[11] = 2798052.50      E[11] = 141.11  I[11] = 5.40    R[11] = 1261.28         D[11] = 0.77
N[12] = 2799607.07      S[12] = 2798053.00      E[12] = 146.56  I[12] = 5.61    R[12] = 1401.02         D[12] = 0.87
N[13] = 2799758.69      S[13] = 2798053.50      E[13] = 152.22  I[13] = 5.83    R[13] = 1546.16         D[13] = 0.98
N[14] = 2799916.13      S[14] = 2798054.00      E[14] = 158.09  I[14] = 6.06    R[14] = 1696.89         D[14] = 1.09
N[15] = 2800079.63      S[15] = 2798054.50      E[15] = 164.19  I[15] = 6.29    R[15] = 1853.45         D[15] = 1.20
N[16] = 2800249.41      S[16] = 2798055.00      E[16] = 170.52  I[16] = 6.53    R[16] = 2016.04         D[16] = 1.32
```

Демонстрация вывода в терминал в ходе выполнения программы



Демонстрация одного из созданных текстовых файлов в ходе выполнения программы

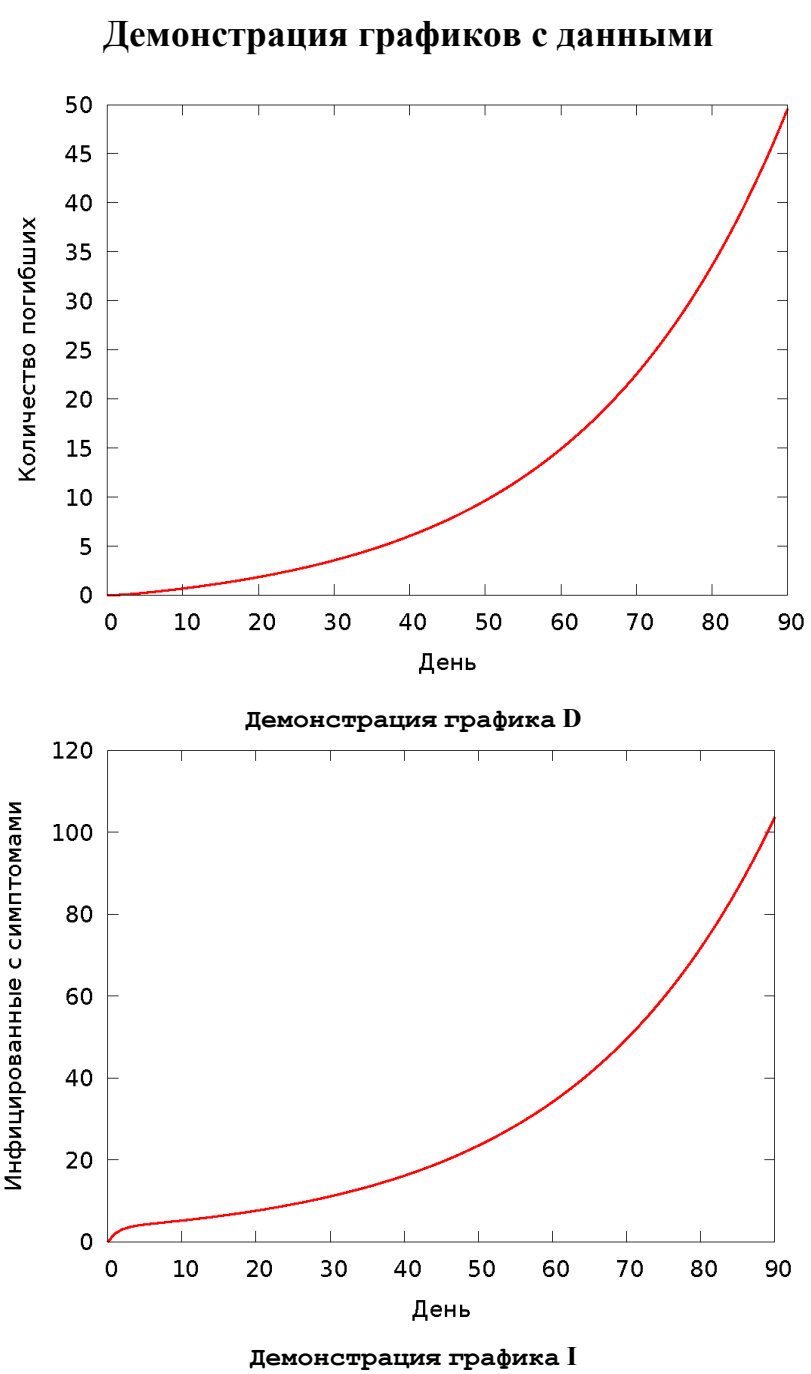
После выполнения программы есть 6 текстовых файлов с удобно записанными в них данными, теперь можно сделать графики с помощью утилиты gnuplot, для наглядности продемонстрирую один из сценариев:

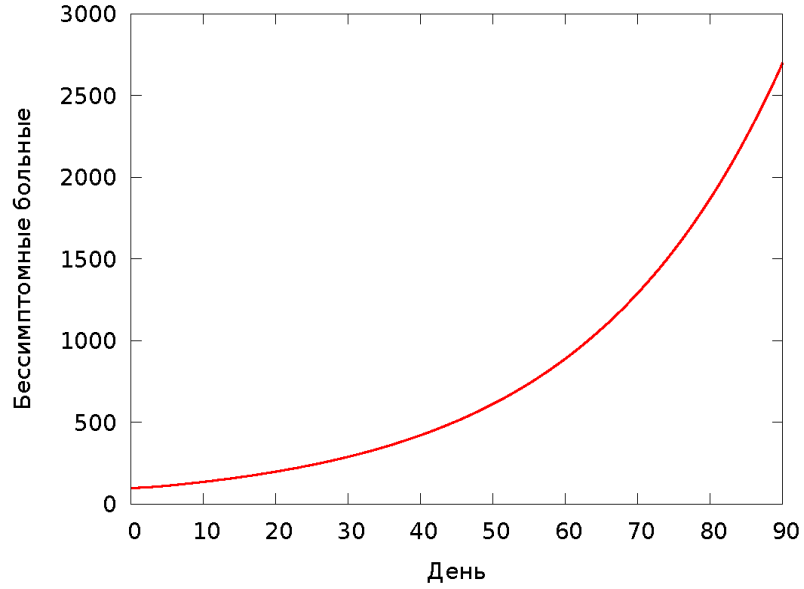
D.gp

VicMath > Course_work > gnuplot > D.gp

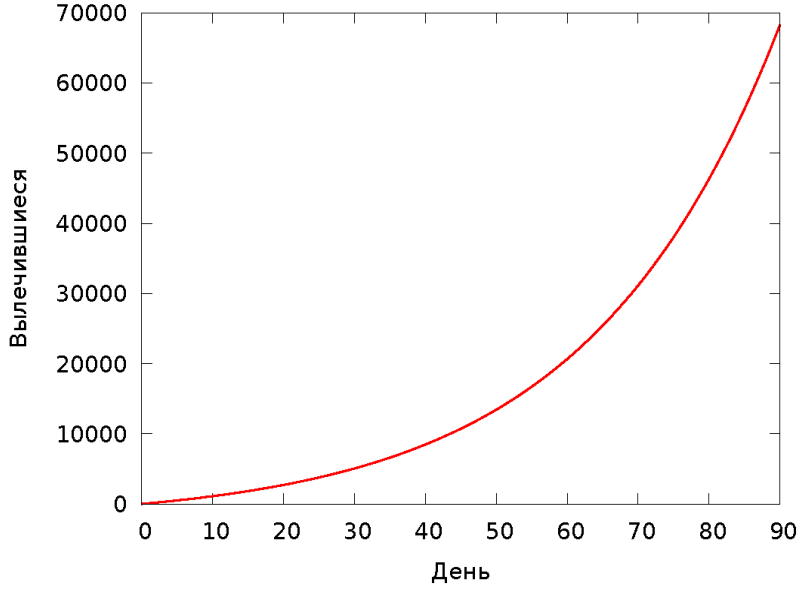
```
1  set terminal pdf color enhanced font 'Calibri,16' size 14cm,10cm
2  set output '../pdf/D.pdf'
3  set key inside left top font 'Calibri,16'
4  set xlabel "День" font 'Calibri,16'
5  set ylabel "Количество погибших" font 'Calibri,16'
6  plot '../Data/D.txt' using 1:2 title "" w l lw 1.5 lc rgb "red"
```

Демонстрация одного из сценариев для построения графика

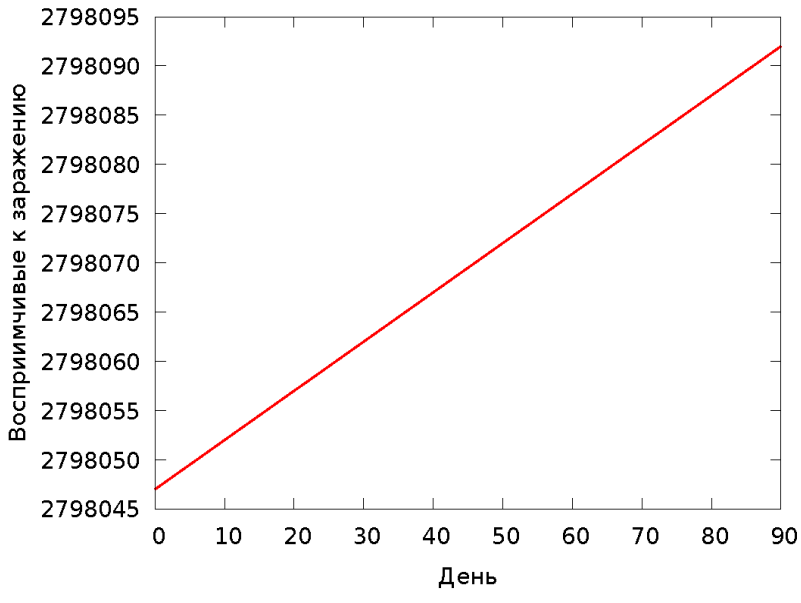




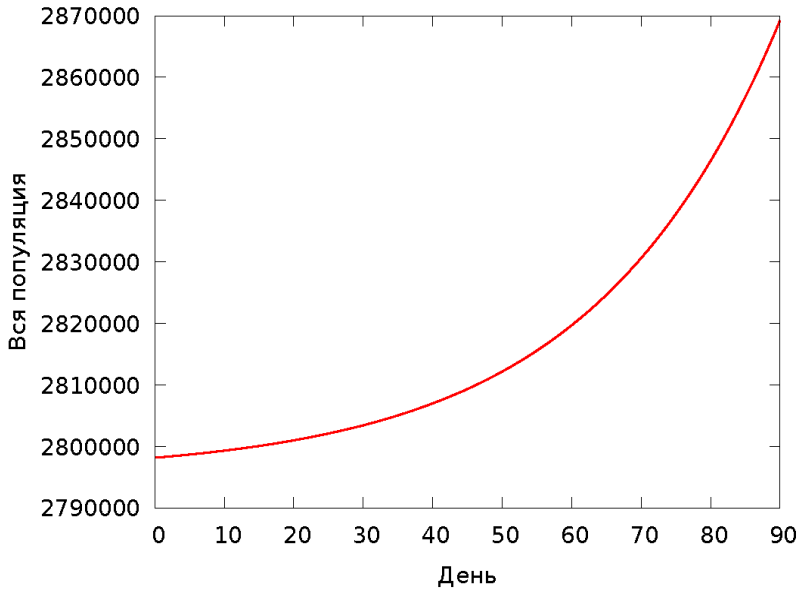
Демонстрация графика E



Демонстрация графика R



Демонстрация графика S



Демонстрация графика N

Вывод

В ходе выполнения данной курсовой работы я произвел расчеты из данной в статье системы уравнений модели SEIR-D с помощью модифицированного метода Эйлера, построил графики по полученным данным, а также узнал больше о COVID-19.

Листинг

SEIR-D.c

```c
#include <stdio.h>
#include "SEIR_D.h"

int main()
{
    int t_0 = 0, T = 90, h = 1;
    int delta_T = T - t_0;
    int n = (delta_T / h) + 1;
    double S[n], E[n], I[n], R[n], D[n], N[n], t[n];
    t[0] = t_0;
    D[0] = 0;
    R[0] = R_0;
    I[0] = 0;
    E[0] = E_0;
    S[0] = 2798170 - E[0] - R[0];
    N[0] = S[0] + E[0] + I[0] + R[0] + D[0];
          printf("N[0] = %.2lf S[0] = %.2lf E[0] = %.2lf I[0] = %.2lf R[0] = %.2lf D[0] = %.2lf\n", N[0], S[0], E[0], I[0], R[0], D[0]);
    for(int i = 1; i < n; i++){
        t[i] = t_0 + i * h;
        D[i] = D[i - 1] + D_function(I[i - 1]);
        I[i] = I[i - 1] + I_with_h(E[i - 1], I[i - 1], h);
        R[i] = R[i - 1] + R_with_h(E[i - 1], I[i - 1], R[i - 1], h);
        E[i] = E[i - 1] + E_with_h(S[i - 1], E[i - 1], I[i - 1], N[i - 1], h);
        S[i] = S[i - 1] + S_with_h(S[i - 1], E[i - 1], I[i - 1], R[i - 1], N[i - 1], h);
        N[i] = D[i] + I[i] + R[i] + E[i] + S[i];
          printf("N[%d] = %.2lf\t S[%d] = %.2lf\t E[%d] = %.2lf\t I[%d] = %.2lf\t R[%d] = %.2lf\t D[%d] = %.2lf\n", i, N[i], i, S[i], i, E[i], i, I[i], i, R[i], i, D[i]);
    }

    FILE* file_D = fopen("Data/D.txt", "w");
    FILE* file_I = fopen("Data/I.txt", "w");
    FILE* file_R = fopen("Data/R.txt", "w");
    FILE* file_E = fopen("Data/E.txt", "w");
    FILE* file_S = fopen("Data/S.txt", "w");
    FILE* file_N = fopen("Data/N.txt", "w");
    for(int i = 0; i < n; i++) {
        fprintf(file_D, "%.0lf %.2lf\n", t[i], D[i]);
        fprintf(file_I, "%.0lf %.2lf\n", t[i], I[i]);
        fprintf(file_R, "%.0lf %.2lf\n", t[i], R[i]);
        fprintf(file_E, "%.0lf %.2lf\n", t[i], E[i]);
        fprintf(file_S, "%.0lf %.2lf\n", t[i], S[i]);
        fprintf(file_N, "%.0lf %.2lf\n", t[i], N[i]);
    }
    fclose(file_D);
    fclose(file_I);
    fclose(file_R);
    fclose(file_E);
    fclose(file_S);
    fclose(file_N);

    return 0;
}



double D_function(double I){
    double answer = mu * I;
```

```
    return answer;
}

double I_function(double E, double I){
    double answer = (k * E) - (beta * I) - (mu * I);
    return answer;
}

double R_function(double E, double I, double R){
    double answer = (beta * I) + (ro * E) - (g * R);
    return answer;
}

double E_function(double S, double E, double I, double N){
   double answer = (1 * S) * ((alpha_I * I) / N + (alpha_E * E) / N) - (k + ro) * E;
   return answer;
}

double S_function(double S, double E, double I, double R, double N){
    double answer = (-1 * S) * ((alpha_I * I) / N + (alpha_E * E) / N) + g * R;
    return answer;
}

double I_with_h(double E, double I, double h){
    double answer = h * I_function(E, I + h / 2 * I_function(E, I));
    return answer;
}

double R_with_h(double E, double I, double R, double h){
    double answer = h * R_function(E, I, R + h / 2 * R_function(E, I, R));
    return answer;
}

double E_with_h(double S, double E, double I, double N, double h){
    double answer = h * E_function(S, (E + h) / (2 * E_function(S, E, I, N)), I, N);
    return answer;
}

double S_with_h(double S, double E, double I, double R, double N, double h){
    double answer = h * S_function((S + h) / (2 * S_function(S, E, I, R, N)), E, I, R, N);
    return answer;
}
```

SEIR-D.h

```
#ifndef SIER_D_H
#define SIER_D_H

const double alpha_I = 0.999, alpha_E = 0.999, k = 0.042, ro = 0.952, beta = 0.999,
    mu = 0.0188, c_isol = 0, E_0 = 99, R_0 = 24, g = 0;

double D_function(double I);
double I_function(double E, double I);
double R_function(double E, double I, double R);
double E_function(double S, double E, double I, double N);
double S_function(double S, double E, double I, double R, double N);
double I_with_h(double E, double I, double h);
double R_with_h(double E, double I, double R, double h);
double E_with_h(double S, double E, double I, double N, double h);
double S_with_h(double S, double E, double I, double R, double N, double h);

#endif
```