Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра вычислительных систем

ОТЧЕТ
по курсовой работе

по дисциплине «**Вычислительная математика**»

Выполнил:
студент гр. ИС-142
«__» июня 2023 г. _____ /Григорьев Ю.В./

Проверил:
преподаватель
«__» июня 2023 г. _____ /Бублей Д.А./

Оценка «_____»

Новосибирск 2023

ОГЛАВЛЕНИЕ

ПОСТАНОВКА ЗАДАЧИ

Из статьи Сибирского Журнала Вычислительной Математики “Математическое моделирование и прогнозирование COVID-19 в Москве и Новосибирской области” (ссылка) решите систему уравнений модели SEIR-D (5) (приложение 1) для Новосибирской области с коэффициентами из таблицы 11 (приложение 2). Решение найдите с помощью метода Эйлера на участке времени от 0 до 90 дней с точностью до 2 знака после запятой.

Приложение 1:

В рамках модели SEIR-D распространение коронавируса COVID-19 описывается системой из 5 нелинейных обыкновенных дифференциальных уравнений на отрезке $t \in [t_0, T]$ [31] (схема модели приведена на рис. 1 справа):

$$\begin{cases} \dfrac{dS}{dt} = -c(t-\tau)\left(\dfrac{\alpha_I S(t) I(t)}{N} + \dfrac{\alpha_E S(t) E(t)}{N}\right) + \gamma R(t), \\ \dfrac{dE}{dt} = c(t-\tau)\left(\dfrac{\alpha_I S(t) I(t)}{N} + \dfrac{\alpha_E S(t) E(t)}{N}\right) - (\kappa + \rho) E(t), \\ \dfrac{dI}{dt} = \kappa E(t) - \beta I(t) - \mu I(t), \\ \dfrac{dR}{dt} = \beta I(t) + \rho E(t) - \gamma R(t), \\ \dfrac{dD}{dt} = \mu I(t). \end{cases} \tag{5}$$

Здесь $N = S + E + I + R + D$ — вся популяция.

Функция, использующая ограничения на передвижения граждан:

$$c(t) = 1 + c^{\text{isol}}\left(1 - \frac{2}{5}a(t)\right), \quad c(t) \in (0, 2).$$

Начальные данные:

$$S(t_0) = S_0, \quad E(t_0) = E_0, \quad I(t_0) = I_0, \quad R(t_0) = R_0, \quad D(t_0) = D_0. \tag{6}$$

Приложение 2:

Таблица 11. Восстановленные параметры для периода измерений 23.03.2020–31.05.2020, Новосибирская область

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

МЕТОД РЕШЕНИЯ ЗАДАЧИ

Для решения данной задачи используется модифицированный метод Эйлера с пересчётом (другие названия — метод Эйлера-Коши, схема “предиктор-корректор”) для системы дифференциальных уравнений, так как необходимо устойчивое решение со вторым порядком точности (до 0,01).

Решение этим методом выглядит следующим образом:

Прогноз:

$$\tilde{y}_i = y_{i-1} + (x_i - x_{i-1}) f(x_{i-1}, y_{i-1}).$$

Коррекция:

$$y_i = y_{i-1} + (x_i - x_{i-1}) \frac{f(x_{i-1}, y_{i-1}) + f(x_i, \tilde{y}_i)}{2}.$$

Или же:

$$\begin{cases} \tilde{y}_{i+1} = y_i + h f(x_i, y_i), \\ y_{i+1} = y_i + \dfrac{h}{2}\left[f(x_i, y_i) + f(x_{i+1}, \tilde{y}_{i+1})\right]. \end{cases}$$

АЛГОРИТМ РЕШЕНИЯ

1. Получение и вычисление входных данных в локальные переменные (a – день начала отсчёта, b – день конца отсчёта, N0 – всё население, S (S0) – восприимчивое население (N0 – E0 – I0 – R0 – D0), E (E0) – количество бессимптомно инфицированных, I (I0) – количество выявленных случаев, R (R0) – количество вылеченных / людей с постоянным иммунитетом, D (D0) – количество умерших).

2. Решение системы дифференциальных уравнений методом Эйлера-Коши первым приближением с размером шага $h = 1$. Сохранение результата переменной с наименьшим значением (d) в дополнительную переменную d1.

3. Запуск цикла для нахождения решения системы дифференциальных уравнений с заданной точностью с корректировкой размера шага разбиения в функции **main()**.
По правилу Рунге, если разница предыдущего и нынешнего решений (*d1* и *d* соответственно) (записывается в переменную *delta*) отличается более чем на заданную точность *eps*, делим шаг разбиения на 2 и решаем систему на следующей итерации с заданным шагом.

3. В цикле: получение результатов решения системы дифференциальных уравнений методом Эйлера-Коши (методом Эйлера с пересчётом) в функции **euler_modified()** и занесение его с решением из прошлой итерации в переменную *delta*.

3.1. По формуле $n = \frac{b-a}{h}$ высчитывается количество разбиений для численного решения системы уравнений.

3.2. Запускается цикл с количеством итераций равным *n*, на каждом шаге которого выполняется следующее:

3.2.1. Подстановка значений *{S, E, I, R, D, h}* в предиктор (1), получение значений *{S1, E1, I1, R1, D1}*.

$$\tilde{y}_{i+1} = y_i + h f(x_i, y_i) \quad (1)$$

3.2.2. Подстановка значений *{S1, E1, I1, R1, D1}* в корректор (2), получение значений *{Si, Ei, Ii, Ri, Di}*.

$$y_{i+1} = y_i + \frac{h}{2}[f(x_i, y_i) + f(x_{i+1}, \tilde{y}_{i+1})] \quad (2)$$

3.2.3. Изменение значений для следующей итерации:
{S, E, I, R, D} = {Si, Ei, Ii, Ri, Di}.

4. Выход из цикла: разница решений (*delta* = |*d* – *d1*|) на двух итерациях отличается не более чем на *eps*.

5. Сравнение общего числа населения (N) до и после численного решения системы дифференциальных уравнений (не должно отличаться).

6. Вывод результатов работы программы (численное решение на 90-й день отсчёта) в терминал.

ЛИСТИНГ ПРОГРАММЫ

Файл **main.cpp**

```
#include <cmath>
#include <iostream>

#define MU 0.0188      // коэф. смертности от COVID-19
#define BETA 0.999     // скорость выздоровления заражённых случаев
#define RO 0.952       // скорость восстановления выявленных случаев
#define ALPHA_E 0.999 // коэф. заражения между бессимптомно-инфицированным и восприимчивым населением
#define ALPHA_I 0.999 // коэф. заражения между инфицированным и восприимчивым населением (социальные факторы)
#define K 0.042        // частота появления симптомов в открытых случаях
#define N0 2798170     // население Новосибирской области
#define E0 99          // начальное количество бессимптомно инфицированных
#define R0 24          // начальное количество вылеченных
#define GAMMA 0        // скорость повторного заражения, раз (0 - устойчивый иммунитет)
#define C 1            // ограничение на передвижения граждан (изначально - 1 + C_ISOL * (...), сокращена до 1, т.к. C_ISOL = 0)
// N - вся популяция, S - восприимчивые, E - заражённые бессимптомные, I - инфицированные с симптомами, R - вылеченные, D - умершие

```

```
16 // система дифференциальных уравнений модели SEIR-D
17 double dS_dt(double S, double E, double I, double R, double D)
18 {
19     double N = S + E + I + R + D;
20     return -C * (ALPHA_I * S * I + ALPHA_E * S * E) / N + GAMMA * R;
21 }
22
23 double dE_dt(double S, double E, double I, double R, double D)
24 {
25     double N = S + E + I + R + D;
26     return C * (ALPHA_I * S * I + ALPHA_E * S * E) / N - (K + RO) * E;
27 }
28
29 double dI_dt(double S, double E, double I, double R, double D)
30 {
31     return K * E - BETA * I - MU * I + 0 * (S + R + D);
32 }
33
34 double dR_dt(double S, double E, double I, double R, double D)
35 {
36     return BETA * I + RO * E - GAMMA * R + 0 * (S + D);
37 }
38
39 double dD_dt(double S, double E, double I, double R, double D)
40 {
41     return MU * I + 0 * (S + E + R + D);
42 }
43
44 // метод Эйлера-Коши (метод Эйлера с пересчётом)
45 void euler_modified(double a, double b, double h, double *S, double *E,
   double *I, double *R, double *D)
46 {
47     int n = (int)ceil((b - a) / h) + 1;
48     double s = *S, e = *E, i = *I, r = *R, d = *D;
49     double si, ei, ii, ri, di;
50     double s1, e1, i1, r1, d1;
51
52     for (int k = 0; k <= n; k++)
53     {
54         s1 = s + h * dS_dt(s, e, i, r, d);
55         e1 = e + h * dE_dt(s, e, i, r, d);
56         i1 = i + h * dI_dt(s, e, i, r, d);
57         r1 = r + h * dR_dt(s, e, i, r, d);
58         d1 = d + h * dD_dt(s, e, i, r, d);
59         // std::cout << k << ": S1 = " << s1 << "; E1 = " << e1 << "; I1
   = " << i1 << "; R1 = " << r1 << "; D1 = " << d1 << std::endl;
60         si = s + (h / 2) * (dS_dt(s, e, i, r, d) + dS_dt(s1, e1, i1, r1,
   d1));
61         ei = e + (h / 2) * (dE_dt(s, e, i, r, d) + dE_dt(s1, e1, i1, r1,
   d1));
62         ii = i + (h / 2) * (dI_dt(s, e, i, r, d) + dI_dt(s1, e1, i1, r1,
   d1));
63         ri = r + (h / 2) * (dR_dt(s, e, i, r, d) + dR_dt(s1, e1, i1, r1,
   d1));
64         di = d + (h / 2) * (dD_dt(s, e, i, r, d) + dD_dt(s1, e1, i1, r1,
   d1));
65         // std::cout << k << ": Si = " << si << "; Ei = " << ei << "; Ii
   = " << ii << "; Ri = " << ri << "; Di = " << di << std::endl;
```

```
 66         s = s1;
 67         e = e1;
 68         i = i1;
 69         r = r1;
 70         d = d1;
 71         // std::cout << k << ": S = " << s << "; E = " << e << "; I = "
    << i << "; R = " << r << "; D = " << d << std::endl;
 72     }
 73     *S = s;
 74     *E = e;
 75     *I = i;
 76     *R = r;
 77     *D = d;
 78 }
 79
 80 int main()
 81 {
 82     std::cout.precision(12);
 83     std::cout.setf(std::ios::fixed);
 84
 85     // начальные данные
 86     int a = 0, b = 90;
 87     double eps = 1e-2, h = 1;
 88     double e = E0, i = 0, r = R0, d = 0, s = N0 - i - e - r - d;
 89     std::cout << "\nНачальные данные для модели SEIR-D:\nN0 = " << N0 <<
    " (всё население)\nS0 = " << (int)floor(s) << " (восприимчивое
    население)\nE0 = " << E0 << " (бессимптомно инфицированные)\nI0 = " <<
    (int)floor(i) << " (выявленные случаи / инфицированные с симптомами)\nR0
    = " << R0 << " (вылечившиеся)\nD0 = " << (int)floor(d) << " (умершие)\na
    = " << a << " (день начала отсчёта), b = " << b << " (день конца
    отсчёта), h = " << h << " (шаг разбиения)" << std::endl;
 90
 91     euler_modified(a, b, h, &s, &e, &i, &r, &d); // первая итерация
 92
 93     double delta, d1 = d;
 94     int k = 1;
 95     do // цикл до заданной точности решения
 96     {
 97         h = h / 2;
 98         e = E0, i = 0, r = R0, d = 0, s = N0 - i - e - r - d;
 99         euler_modified(a, b, h, &s, &e, &i, &r, &d);
100         delta = fabs(d - d1);
101         d1 = d;
102         std::cout << k << ": delta = " << delta << ", h = " << h <<
    std::endl;
103         k++;
104     } while (delta > eps);
105
106     std::cout << "\nРезультаты метода Эйлера-Коши (метода Эйлера с
    пересчётом):\nE (бессимптомно инфицированных) = " << (int)floor(e) <<
    "\nI (выявленные случаи / инфицированные с симптомами) = " <<
    (int)floor(i) << "\nD (количество умерших) = " << (int)floor(d) <<
    std::endl;
107     double n = s + e + i + r + d;
108     std::cout << "N (final) = " << (int)round(n) << " = N0 (начальное
    население) => ни один человек не потерян\n\n";
109     return 0;
110 }
```

РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ

Скриншот:

```
allenvox@MacBook-Pro-Yuriy seird % ./main

Начальные данные для модели SEIR-D:
N0 = 2798170 (всё население)
S0 = 2798047 (восприимчивое население)
E0 = 99 (бессимптомно инфицированные)
I0 = 0 (выявленные случаи / инфицированные с симптомами)
R0 = 24 (вылечившиеся)
D0 = 0 (умершие)
a = 0 (день начала отсчёта), b = 90 (день конца отсчёта), h = 1.000000000000 (шаг разбиения)
1: delta = 0.549132494740, h = 0.500000000000
2: delta = 0.273574598533, h = 0.250000000000
3: delta = 0.136536509793, h = 0.125000000000
4: delta = 0.068205190921, h = 0.062500000000
5: delta = 0.034086783203, h = 0.031250000000
6: delta = 0.017039432727, h = 0.015625000000
7: delta = 0.008518725917, h = 0.007812500000

Результаты метода Эйлера-Коши (метода Эйлера с пересчётом):
E (бессимптомно инфицированных) = 2487
I (выявленные случаи / инфицированные с симптомами) = 101
D (количество умерших) = 60
N (final) = 2798170 = N0 (начальное население) => ни один человек не потерян

allenvox@MacBook-Pro-Yuriy seird %
```