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

Кафедра прикладной математики
и кибернетики

ОТЧЁТ
по курсовой работе
по дисциплине «**Вычислительная математика**»

Выполнил:
студент гр. ИС-241
«8» мая 2024 г. Бондаренко А.А.

Проверил:
преподаватель
«__» мая 2024 г. _____ Чупрыно Л.А.

Оценка «_____»

Новосибирск 2024

ОГЛАВЛЕНИЕ

ПОСТАНОВКА ЗАДАЧИ

1. Условия выполнения работы

Результатом курсовой работы является отчет и программа на языке C/C++. Не допускается использовать сторонние вычислительные библиотеки для решения задачи.

Для защиты необходимо иметь:

- работающую программу (без комментариев в коде!);
- отчёт по курсовой работе – должен содержать:
  1) Постановка задачи;
  2) Описание теории метода решения задачи;
  3) Схема алгоритма решения;
  4) Листинг программы с комментариями;
  5) Результат работы программы (оформляется в виде таблице и в виде графиков, построенных на основе таблицы с помощью сторонних программ, например, Excel)

2. Задача

Решите систему уравнений SEIR-D, моделирующую распространение инфекции COVID-19 для Новосибирской области с заданными коэффициентами. Решение найдите с помощью метода Эйлера на участке времени от 0 до 90 дней с точностью до 2 знака после запятой.

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:

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

Иллюстрация 1: Система уравнений из приведённой статьи

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

Иллюстрация 2: Заданные коэффициенты из приведённой статьи

ОПИСАНИЕ ТЕОРИИ МЕТОДА РЕШЕНИЯ ЗАДАЧИ

Постановка задачи предполагает решение системы с точностью до 2 знаков после запятой методом Эйлера, однако он является методом первого порядка точности, то есть имеет слишком низкую точность для приведённой краевой задачи. Поэтому для решения задачи использовался метод Эйлера-Коши, являющийся методом второго порядка точности – таким образом можно прийти к решению необходимой в задаче работы точности.

Общий вид уравнений, подлежащий решению методом Эйлера-Коши:

$$\begin{cases} y' = f(x, y), \\ y(x_0) = y_0. \end{cases}$$

Иллюстрация 3: Общий вид уравнений для решения методом Эйлера-Коши

Принцип решения задачи Коши данным методом:

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

Иллюстрация 4: Общий вид решения по методу Эйлера-Коши

При таком подходе решение ищется на интервале $[x_0, b]$ с шагом его разбиения на n равных частей, где:

1) $h = \frac{b - x_0}{n}$,

2) $x = x_0 + hi$,

3) i = 0, 1, …, n.

СХЕМА АЛГОРИТМА РЕШЕНИЯ

1. Инициализация коэффициентов модели SEIR-D:
   ALPHA_E = 0.999 (коэффициент заражения между бессимптомно-инфицированным и восприимчивым населением)
   ALPHA_I = 0.999 (коэффициент заражения между инфицированным и восприимчивым населением (социальные факторы))
   K = 0.042 (частота появления симптомов в открытых случаях)
   MU = 0.0188 (коэффициент смертности)
   BETA = 0.999 (скорость выздоровления заражённых случаев)
   RO = 0.952 (скорость восстановления выявленных случаев)
   E0 = 99 (начальное количество бессимптомно инфицированных)
   R0 = 24 (начальное количество вылеченных)
   C = 1 (ограничение на передвижения граждан)

2. Задание функции, вычисляющей численность всей популяции на основе численности остальных категорий SEIR-D

3. Задание функции, вычисляющих дифференциальные уравнения системы
   dS_dt - восприимчивые к заражению
   dE_dt - бессимптомные больные
   dI_dt - инфицированные с симптомами
   dR_dt - вылечившиеся
   dD_dt – умершие

4. Задание функции, реализующей Метод Эйлера-Коши для решения дифференциальных уравнений. Использует стартовые значения приведённых параметров, в ходе работы изменяя их с шагом времени h

5. Открытие файла для сохранения значений работы программы в соответствии «номер строки – количество выявленных заражённых»

6. Цикл для запуска функции с методом Эйлера-Коши нужное количество раз (соответствует заданному количеству дней). Здесь же реализована запись в файл

7. Выход из цикла и закрытие файла по достижении нужного количества дней

Данный алгоритм позволяет получить решение искомой задачи в наглядной форме, по которой можно будет в дальнейшем продемонстрировать динамику модели распространения вируса.

РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ

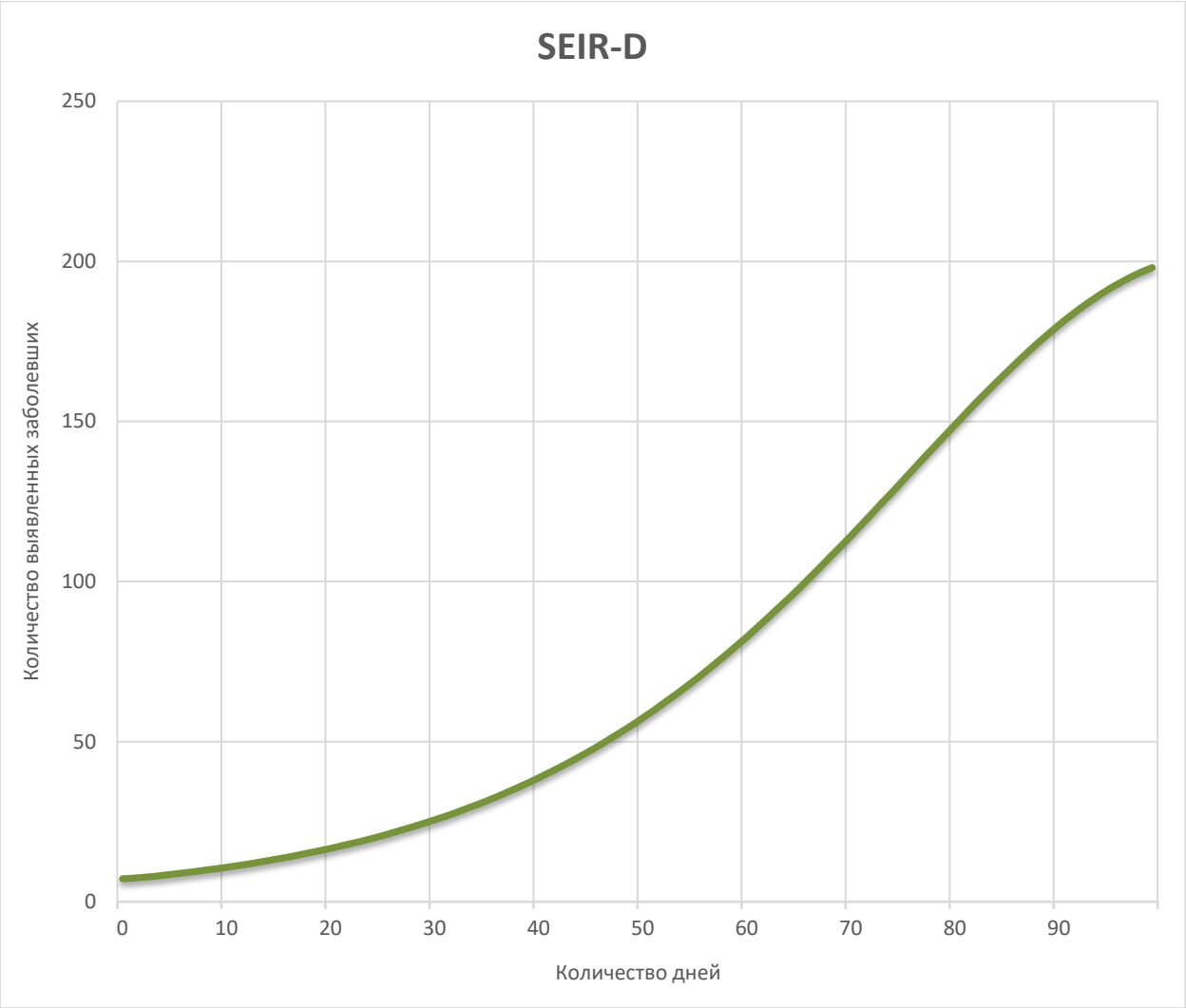

Иллюстрация 5: График динамики заболеваемости COVID-19 в Новосибирской области по SEIR-D

График был построен на основе данных, выведенных программой в файл и позднее занесённых в таблицу.

Общий вид результата соответствует приведённому в статье.

ЛИСТИНГ ПРОГРАММЫ

```
1 | #include <iostream>
2 | #include <fstream>
3 |
4 | #define ALPHA_E 0.999
5 | #define ALPHA_I 0.999
6 | #define K 0.042
7 | #define MU 0.0188
8 | #define BETA 0.999
9 | #define RO 0.952
10| #define E0 99
11| #define R0 24
12| #define C 1
13|
14| using namespace std;
15|
16| int population(double S, double E, double I, double R, double D)
17| {
18|     return S + E + I + R + D;
19| }
20|
21| double dS_dt(double N, double S, double E, double I)
22| {
23|     return -C * (ALPHA_I * S * I + ALPHA_E * S * E) / N;
24| }
25|
26| double dE_dt(double N, double S, double E, double I)
27| {
28|     return C * (ALPHA_I * S * I + ALPHA_E * S * E) / N - (K + RO) * E;
29| }
30|
31| double dI_dt(double E, double I)
32| {
33|     return K * E - BETA * I - MU * I;
34| }
35|
36| double dR_dt(double E, double I)
37| {
38|     return BETA * I + RO * E;
39| }
40|
41| double dD_dt(double I)
42| {
43|     return MU * I;
44| }
45|
46| double euler_SEIRD(double S, double E, double I, double R, double D, int
n, double h)
47| {
48|     double s = S, e = E, i = I, r = R, d = D;
49|     double si, ei, ii, ri, di;
50|     double s1, e1, i1;
51|
52|     for (int k = 0; k < n; k++)
53|     {
54|         int N = population(s, e, i, r, d);
55|
56|         s1 = S + h * dS_dt(N, s, e, i);
57|         e1 = E + h * dE_dt(N, s, e, i);
58|         i1 = I + h * dI_dt(e, i);
59|
```

```
60|         s = S + (h / 2) * (dS_dt(N, S, E, I) + dS_dt(N, s1, e1, i1));
61|         e = E + (h / 2) * (dE_dt(N, S, E, I) + dE_dt(N, s1, e1, i1));
62|         i = i + (h / 2) * (dI_dt(E, I) + dI_dt(e1, i1));
63|         r = R + (h / 2) * (dR_dt(E, I) + dR_dt(e1, i1));
64|         d = D + (h / 2) * (dD_dt(I) + dD_dt(i1));
65|
66|         S = s;
67|         E = e;
68|         I = i;
69|         R = r;
70|         D = d;
71|     }
72|
73|     return K * e / 0.58;
74| }
75|
76| int main()
77| {
78|     double h = 1;
79|
80|     double S = 2798047;
81|     double E = E0;
82|     double I = 0;
83|     double R = R0;
84|     double D = 0;
85|
86|     ofstream fout;
87|
88|     fout.open("file.txt");
89|
90|     for (int k = 0; k < 100; k++)
91|     {
92|         fout << euler_SEIRD(S, E, I, R, D, k, h) << endl;
93|     }
94|
95|     fout.close();
96|
97|     return 0;
98| }
```