Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский государственный университет

телекоммуникаций и информатики»

(ФГОБУ ВО «СибГУТИ)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К курсовому проекту по дисциплине
«Вычислительная математика»
на тему
SEIR-D

Выполнил: студент Пономаренко В.О.

Группы ИА-231

Работу принял _____ Лукинов Виталий Леонидович

Защищена _____ Оценка _____

Новосибирск 2024

Постановка задачи

Решите систему уравнений (5) модель SEIR-D для Новосибирской области с коэффициентами из таблицы 11.Решение найдите с помощью метода Эйлера на участке времени от 0 до 90 дней с точностью до 2 знака после запятой.

Описание теории метода решения задачи

2. Математическая модель SEIR-D

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:

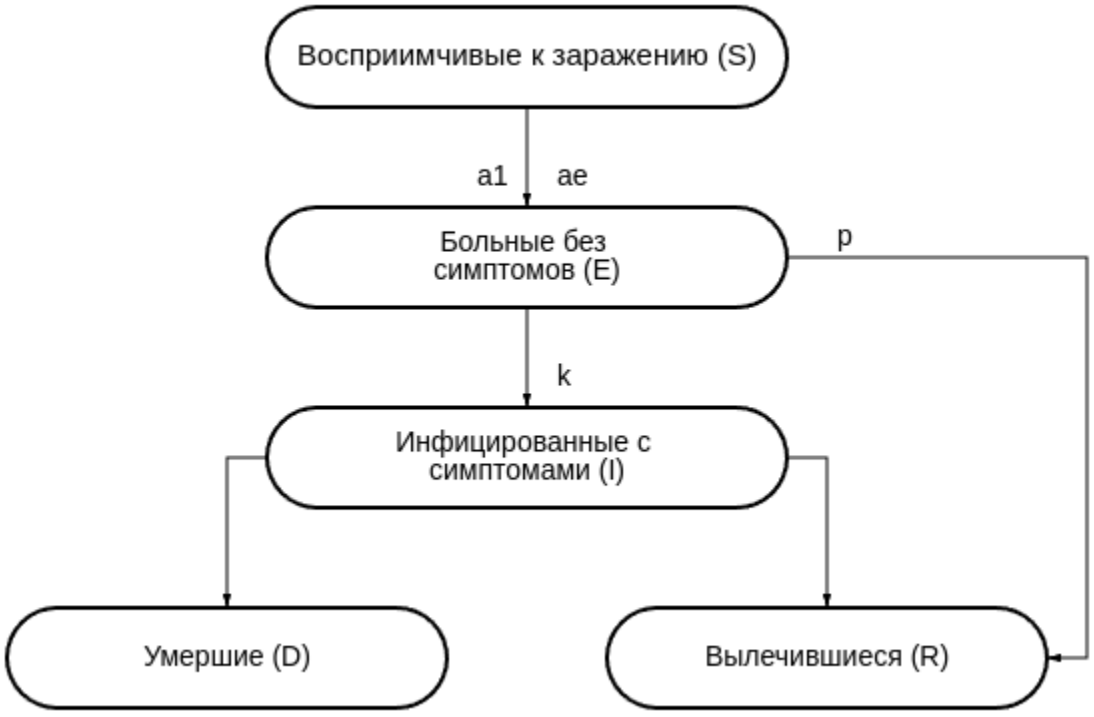
$$S(t_0) = S_0, \quad E(t_0) = E_0, \quad I(t_0) = I_0, \quad R(t_0) = R_0, \quad D(t_0) = D_0. \tag{6}$$

Из таблицы ниже мы извлекали числовые значения, которые затем использовали для вычисления решений наших дифференциальных уравнений:

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

Схема алгоритма решения



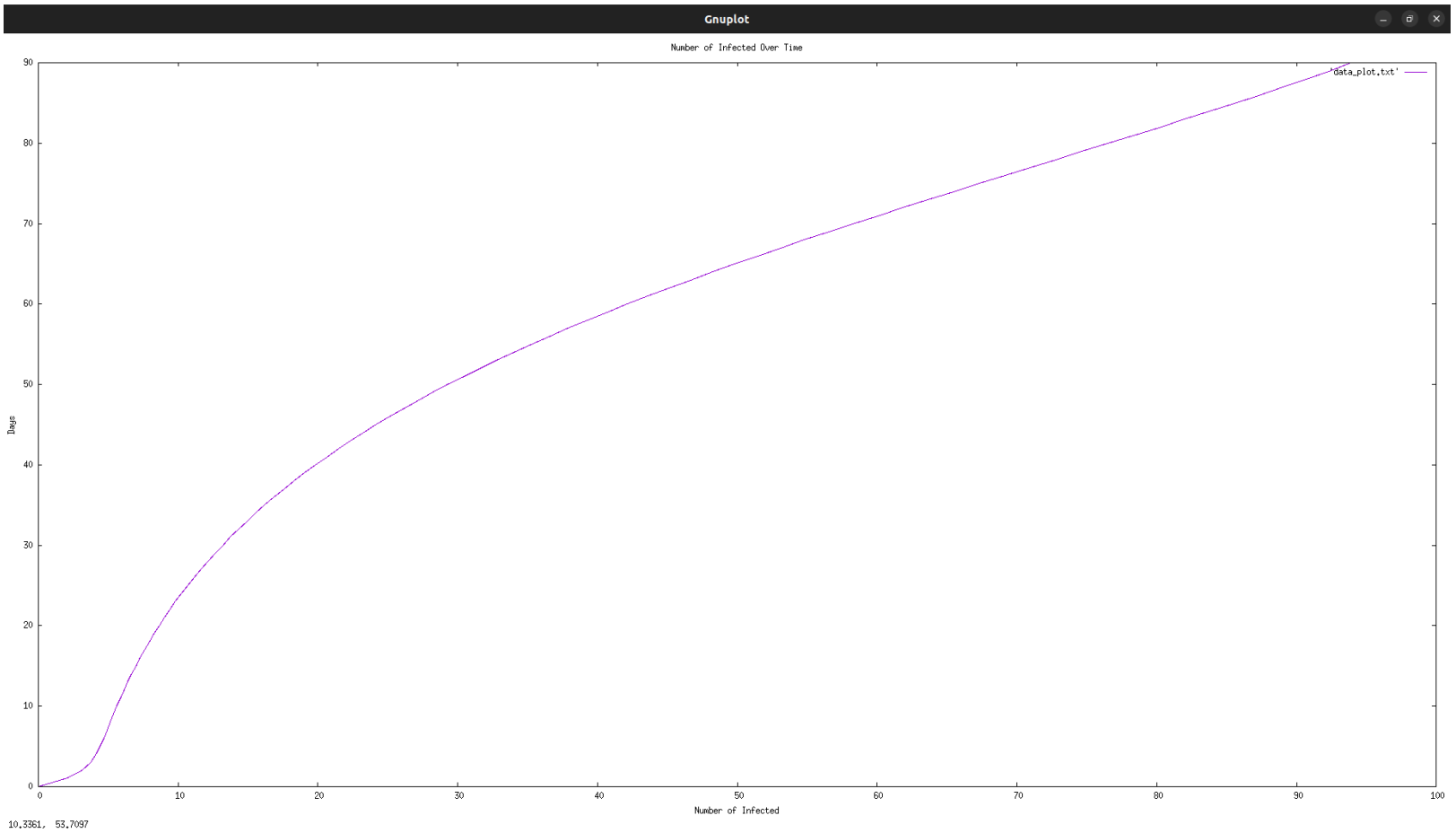
Листинг программы

```
#include <iostream> // Библиотека для стандартного ввода-вывода
#include <fstream> // Библиотека для работы с файлами
#include <vector> // Библиотека для работы с векторами
#include <cstdio> // Библиотека для использования временных файлов

// Объявление глобальных переменных
double a = 1, y = 0, cisol = 0, mu = 0.0188, beta = 0.999, rho = 0.952, kappa = 0.042,
alphaI = 0.999, alphaE = 0.999;
double h = 1, step = 1, t = 0, day = 90, N = 2798170, D0 = 0, I0 = 0, R0 = 24, E0 = 99, S0 =
N - E0 - R0;

// Объявление функций
double func_c(double c_isol);
double func_S(double S, double E, double I, double R, double N);
double delta_S(double S, double E, double I, double R, double N, double h);
double func_E(double S, double E, double I, double N);
double delta_E(double S, double E, double I, double N, double h);
double func_I(double E, double I);
double delta_I(double E, double I, double h);
double func_R(double E, double I, double R);
double delta_R(double E, double I, double R, double h);
double func_D(double I);

int main() {
// Инициализация векторов для хранения значений переменных
std::vector<double> S_t = {S0}, E_t = {E0}, I_t = {I0}, R_t = {R0}, D_t = {D0}, N_t =
{S_t.back() + E_t.back() + I_t.back() + R_t.back() + D_t.back()};
std::vector<int> days;
// Вычисление значений переменных на каждый день в течение периода day
for (int i = 0; i <= day; i++) {
days.push_back(t + i * h); // Добавление значения дня в вектор days
S_t.push_back(S_t.back() + delta_S(S_t.back(), E_t.back(), I_t.back(), R_t.back(), N_t.back(),
h)); // Расчет S и добавление в вектор S_t
E_t.push_back(E_t.back() + delta_E(S_t.back(), E_t.back(), I_t.back(), N_t.back(), h)); // Расчет
E и добавление в вектор E_t
I_t.push_back(I_t.back() + delta_I(E_t.back(), I_t.back(), h)); // Расчет I и добавление в
вектор I_t
R_t.push_back(R_t.back() + delta_R(E_t.back(), I_t.back(), R_t.back(), h)); // Расчет R и
добавление в вектор R_t
D_t.push_back(D_t.back() + func_D(I_t.back())); // Расчет D и добавление в вектор D_t
N_t.push_back(S_t.back() + E_t.back() + I_t.back() + R_t.back() + D_t.back()); // Обновление
общего числа популяции и добавление в вектор N_t
}
```

```
// Вывод результатов в файл result.txt
std::ofstream file("result.txt", std::ios::out);
for (int i = 0; i <= day; i++) {
char str_data[100];
// Форматирование данных и запись в файл
sprintf(str_data, "%d %.2lf %.2lf %.2lf %.2lf %.2lf\n", days[i], N_t[i], S_t[i], E_t[i], I_t[i],
R_t[i], D_t[i]);
std::string data(str_data);
file << data;
}
file.close(); // Закрытие файла result.txt

// Создание временного файла с данными для построения графика
std::ofstream data_file("data_plot.txt", std::ios::out);
for (int i = 0; i <= day; i++) {
data_file << I_t[i] << " " << days[i] << std::endl; // Запись данных в файл в формате "I
день"
}
data_file.close(); // Закрытие файла data_plot.txt

// Создание скрипта для построения графика с помощью Gnuplot
std::ofstream script_file("plot_script.plt", std::ios::out);
script_file << "set title 'Number of Infected Over Time'\n"; // Установка заголовка графика
script_file << "set xlabel 'Number of Infected'\n"; // Установка подписи оси x
script_file << "set ylabel 'Days'\n"; // Установка подписи оси y
script_file << "set term x11\n"; // Установка среды выполнения Gnuplot на "x11"
script_file << "plot 'data_plot.txt' with lines\n"; // Построение графика из файла
data_plot.txt
script_file << "pause -1 'Press any key to exit'\n"; // Ожидание нажатия клавиши для
завершения
script_file.close(); // Закрытие файла plot_script.plt

system("gnuplot plot_script.plt"); // Запуск Gnuplot для построения графика

return 0; // Возврат 0 в случае успешного завершения программы
}

// Определение функций для вычисления дифференциальных уравнений
double func_c(double c_isol) {
return 1 + c_isol * (1 - 2 / 5 * a); // Вычисление коэффициента c
}

double func_S(double S, double E, double I, double R, double N) {
return -1 * func_c(cisol) * S / N * (alphaI * I + alphaE * E) + y * R; // Вычисление изменения S
}

double delta_S(double S, double E, double I, double R, double N, double h) {
```

```
return h * func_S(S + h / 2 * func_S(S, E, I, R, N), E, I, R, N); // Вычисление приращения S
}

double func_E(double S, double E, double I, double N) {
return func_c(cisol) * S / N * (alphaI * I + alphaE * E) - (kappa + rho) * E; // Вычисление
изменения E
}

double delta_E(double S, double E, double I, double N, double h) {
return h * func_E(S, E + h / 2 * func_E(S, E, I, N), I, N); // Вычисление приращения E
}

double func_I(double E, double I) {
return kappa * E - beta * I - mu * I; // Вычисление изменения I
}

double delta_I(double E, double I, double h) {
return h * func_I(E, I + h / 2 * func_I(E, I)); // Вычисление приращения I
}

double func_R(double E, double I, double R) {
return beta * I + rho * E - y * R; // Вычисление изменения R
}

double delta_R(double E, double I, double R, double h) {
return h * func_R(E, I, R + h / 2 * func_R(E, I, R)); // Вычисление приращения R
}

double func_D(double I) {
return mu * I; // Вычисление изменения D
}
```

Результат работы программы



D	N	S	E	I	R	D
0	2798170.00	2798047.00	99.00	0.00	24.00	0.00
1	2798170.45	2797948.11	99.49	2.05	120.76	0.04
2	2798172.96	2797846.68	102.03	3.13	221.02	0.10
3	2798176.56	2797741.63	105.66	3.75	325.35	0.17
4	2798180.80	2797632.36	109.92	4.14	434.13	0.25
5	2798185.44	2797518.44	114.59	4.43	547.65	0.33
6	2798190.40	2797399.56	119.57	4.68	666.16	0.42
7	2798195.62	2797275.47	124.82	4.92	789.91	0.51
8	2798201.09	2797145.91	130.32	5.15	919.11	0.61
9	2798206.81	2797010.63	136.07	5.38	1054.03	0.71
10	2798212.79	2796869.38	142.07	5.62	1194.90	0.81
11	2798219.02	2796721.91	148.33	5.87	1341.98	0.92
12	2798225.51	2796567.95	154.86	6.13	1495.53	1.04
13	2798232.29	2796407.21	161.67	6.40	1655.84	1.16
14	2798239.35	2796239.42	168.77	6.68	1823.19	1.28
15	2798246.71	2796064.27	176.17	6.98	1997.87	1.42
16	2798254.38	2795881.45	183.88	7.28	2180.21	1.55
17	2798262.37	2795690.65	191.92	7.60	2370.51	1.70
18	2798270.69	2795491.52	200.29	7.93	2569.11	1.84
19	2798279.37	2795283.72	209.01	8.28	2776.36	2.00
20	2798288.40	2795066.88	218.10	8.64	2992.62	2.16
21	2798297.80	2794840.64	227.56	9.01	3218.26	2.33
22	2798307.59	2794604.61	237.41	9.41	3453.66	2.51
23	2798317.78	2794358.38	247.66	9.81	3699.24	2.69
24	2798328.39	2794101.54	258.33	10.24	3955.39	2.89
25	2798339.43	2793833.66	269.44	10.68	4222.56	3.09
26	2798350.91	2793554.29	280.99	11.14	4501.19	3.30
27	2798362.85	2793262.97	293.01	11.61	4791.74	3.51
28	2798375.27	2792959.23	305.51	12.11	5094.68	3.74
29	2798388.18	2792642.56	318.50	12.63	5410.51	3.98
30	2798401.59	2792312.46	332.01	13.16	5739.73	4.23
31	2798415.53	2791968.40	346.05	13.72	6082.88	4.49
32	2798430.01	2791609.85	360.63	14.30	6440.48	4.75

33 2798445.05 2791236.23 375.77 14.90 6813.11 5.03
34 2798460.66 2790846.98 391.50 15.53 7201.33 5.33
35 2798476.86 2790441.49 407.82 16.18 7605.74 5.63
36 2798493.67 2790019.16 424.76 16.85 8026.95 5.95
37 2798511.10 2789579.36 442.33 17.55 8465.59 6.28
38 2798529.18 2789121.43 460.55 18.28 8922.29 6.62
39 2798547.91 2788644.73 479.44 19.03 9397.73 6.98
40 2798567.32 2788148.56 499.01 19.81 9892.59 7.35
41 2798587.42 2787632.23 519.29 20.62 10407.55 7.74
42 2798608.23 2787095.03 540.28 21.46 10943.32 8.14
43 2798629.76 2786536.22 562.00 22.32 11500.65 8.56
44 2798652.02 2785955.07 584.47 23.22 12080.26 9.00
45 2798675.04 2785350.80 607.71 24.15 12682.93 9.45
46 2798698.83 2784722.65 631.73 25.11 13309.42 9.92
47 2798723.39 2784069.83 656.54 26.10 13960.51 10.41
48 2798748.74 2783391.53 682.15 27.12 14637.02 10.92
49 2798774.90 2782686.94 708.58 28.18 15339.74 11.45
50 2798801.86 2781955.24 735.84 29.27 16069.51 12.01
51 2798829.65 2781195.59 763.94 30.40 16827.14 12.58
52 2798858.26 2780407.16 792.88 31.56 17613.49 13.17
53 2798887.70 2779589.10 822.67 32.75 18429.40 13.79
54 2798917.98 2778740.55 853.32 33.98 19275.71 14.42
55 2798949.10 2777860.66 884.83 35.25 20153.27 15.09
56 2798981.05 2776948.58 917.19 36.55 21062.95 15.77
57 2799013.83 2776003.46 950.41 37.88 22005.59 16.49
58 2799047.44 2775024.44 984.49 39.25 22982.04 17.22
59 2799081.87 2774010.68 1019.41 40.66 23993.13 17.99
60 2799117.11 2772961.36 1055.16 42.10 25039.71 18.78
61 2799153.15 2771875.65 1091.74 43.58 26122.58 19.60
62 2799189.96 2770752.74 1129.13 45.08 27242.55 20.45
63 2799227.52 2769591.86 1167.31 46.63 28400.41 21.32
64 2799265.83 2768392.24 1206.25 48.20 29596.91 22.23
65 2799304.83 2767153.15 1245.92 49.81 30832.78 23.17
66 2799344.51 2765873.88 1286.31 51.44 32108.74 24.13

67 2799384.83 2764553.75 1327.37 53.11 33425.46 25.13
68 2799425.74 2763192.15 1369.07 54.80 34783.56 26.16
69 2799467.21 2761788.47 1411.36 56.52 36183.63 27.22
70 2799509.18 2760342.18 1454.19 58.26 37626.23 28.32
71 2799551.60 2758852.77 1497.52 60.03 39111.84 29.45
72 2799594.42 2757319.83 1541.28 61.81 40640.89 30.61
73 2799637.58 2755742.97 1585.42 63.62 42213.76 31.81
74 2799681.00 2754121.89 1629.87 65.44 43830.76 33.04
75 2799724.61 2752456.35 1674.55 67.27 45492.14 34.30
76 2799768.35 2750746.18 1719.40 69.11 47198.05 35.60
77 2799812.12 2748991.30 1764.34 70.96 48948.59 36.93
78 2799855.86 2747191.70 1809.28 72.81 50743.76 38.30
79 2799899.46 2745347.48 1854.13 74.66 52583.48 39.71
80 2799942.84 2743458.81 1898.80 76.51 54467.57 41.15
81 2799985.90 2741525.97 1943.20 78.35 56395.77 42.62
82 2800028.55 2739549.32 1987.23 80.17 58367.70 44.13
83 2800070.68 2737529.34 2030.78 81.99 60382.91 45.67
84 2800112.19 2735466.61 2073.75 83.78 62440.82 47.24
85 2800152.98 2733361.81 2116.03 85.55 64540.74 48.85
86 2800192.93 2731215.73 2157.52 87.29 66681.90 50.49
87 2800231.94 2729029.28 2198.10 89.00 68863.40 52.16
88 2800269.90 2726803.48 2237.66 90.67 71084.23 53.87
89 2800306.71 2724539.45 2276.09 92.29 73343.27 55.60
90 2800342.25 2722238.42 2313.28 93.88 75639.30 57.37