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 3](#_Toc35593781)

[МЕТОД РЕШЕНИЯ ЗАДАЧИ 4](#_Toc35593782)

[АЛГОРИТМ РЕШЕНИЯ 4](#_Toc35593783)

[ЛИСТИНГ ПРОГРАММЫ 5](#_Toc35593781)

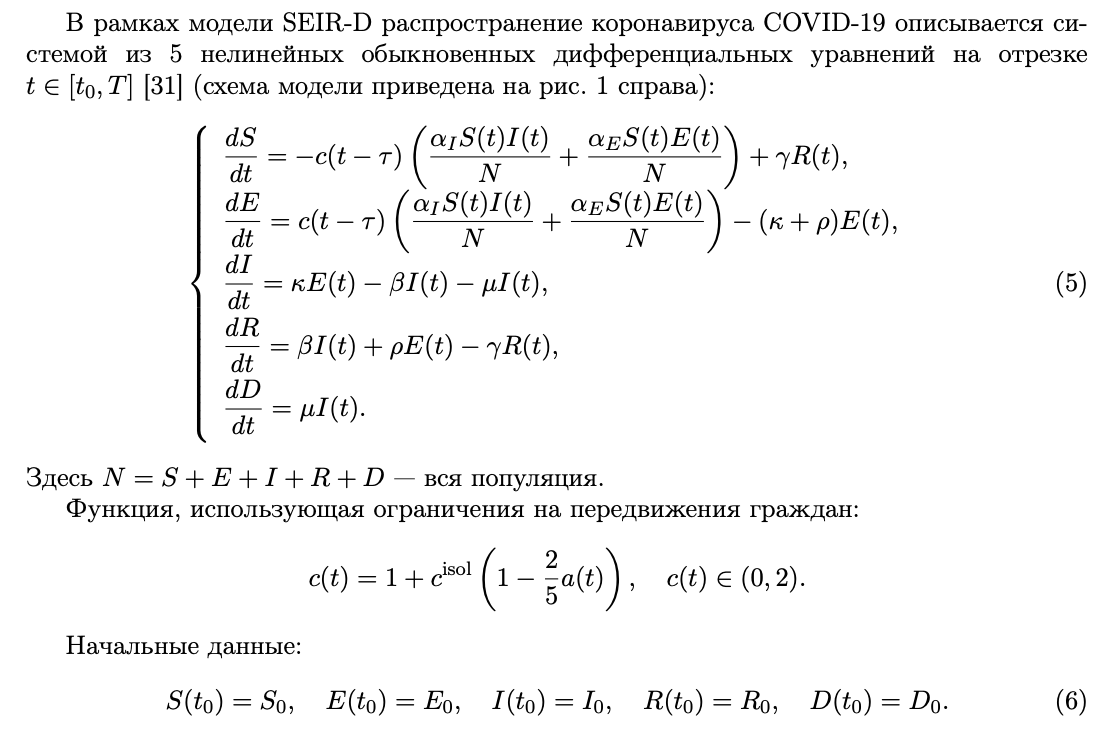
[РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ 8](#_Toc35593782)

# **ПОСТАНОВКА ЗАДАЧИ**

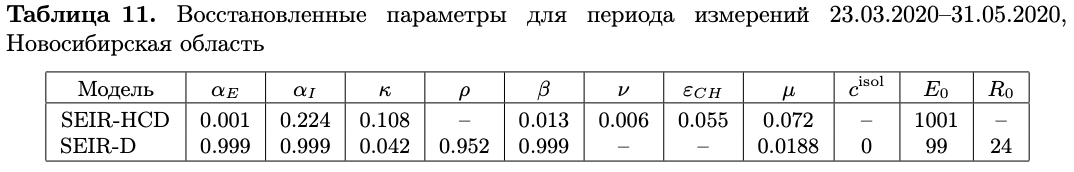
Из статьи Сибирского Журнала Вычислительной Математики “Математическое моделирование и прогнозирование COVID-19 в Москве и Новосибирской области” ([ссылка](https://eios.sibsutis.ru/pluginfile.php/176344/question/questiontext/477862/3/6312609/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F%20%D0%B2%20%D0%A1%D0%B8%D0%B1%D0%96%D0%92%D0%9C%20%D0%9A%D1%80%D0%B8%D0%B2%D0%BE%D1%80%D0%BE%D1%82%D1%8C%D0%BA%D0%BE.pdf)) 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:



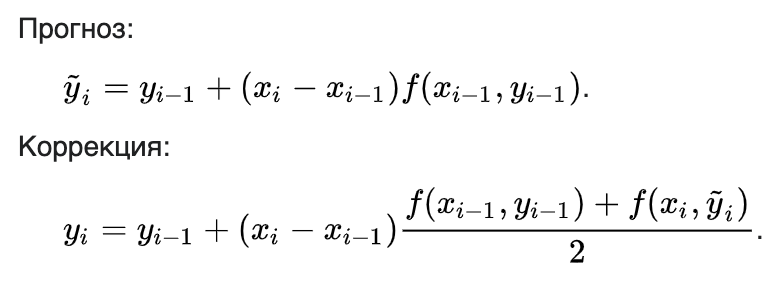
Приложение 2:



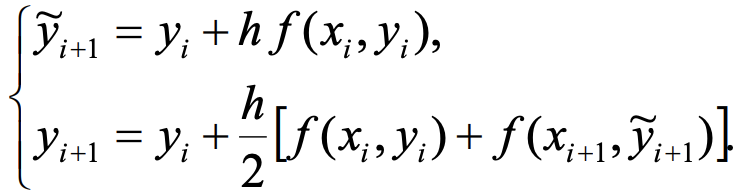
**МЕТОД РЕШЕНИЯ ЗАДАЧИ**

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:



Или же:



**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*.

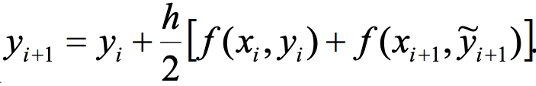
3.1. По формуле высчитывается количество разбиений для численного решения системы уравнений.

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}*.

 (1)

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

 (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**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110 | #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  // система дифференциальных уравнений модели SEIR-D  double dS\_dt(double S, double E, double I, double R, double D)  {  double N = S + E + I + R + D;  return -C \* (ALPHA\_I \* S \* I + ALPHA\_E \* S \* E) / N + GAMMA \* R;  }  double dE\_dt(double S, double E, double I, double R, double D)  {  double N = S + E + I + R + D;  return C \* (ALPHA\_I \* S \* I + ALPHA\_E \* S \* E) / N - (K + RO) \* E;  }  double dI\_dt(double S, double E, double I, double R, double D)  {  return K \* E - BETA \* I - MU \* I + 0 \* (S + R + D);  }  double dR\_dt(double S, double E, double I, double R, double D)  {  return BETA \* I + RO \* E - GAMMA \* R + 0 \* (S + D);  }  double dD\_dt(double S, double E, double I, double R, double D)  {  return MU \* I + 0 \* (S + E + R + D);  }  // метод Эйлера-Коши (метод Эйлера с пересчётом)  void euler\_modified(double a, double b, double h, double \*S, double \*E, double \*I, double \*R, double \*D)  {  int n = (int)ceil((b - a) / h) + 1;  double s = \*S, e = \*E, i = \*I, r = \*R, d = \*D;  double si, ei, ii, ri, di;  double s1, e1, i1, r1, d1;  for (int k = 0; k <= n; k++)  {  s1 = s + h \* dS\_dt(s, e, i, r, d);  e1 = e + h \* dE\_dt(s, e, i, r, d);  i1 = i + h \* dI\_dt(s, e, i, r, d);  r1 = r + h \* dR\_dt(s, e, i, r, d);  d1 = d + h \* dD\_dt(s, e, i, r, d);  // std::cout << k << ": S1 = " << s1 << "; E1 = " << e1 << "; I1 = " << i1 << "; R1 = " << r1 << "; D1 = " << d1 << std::endl;  si = s + (h / 2) \* (dS\_dt(s, e, i, r, d) + dS\_dt(s1, e1, i1, r1, d1));  ei = e + (h / 2) \* (dE\_dt(s, e, i, r, d) + dE\_dt(s1, e1, i1, r1, d1));  ii = i + (h / 2) \* (dI\_dt(s, e, i, r, d) + dI\_dt(s1, e1, i1, r1, d1));  ri = r + (h / 2) \* (dR\_dt(s, e, i, r, d) + dR\_dt(s1, e1, i1, r1, d1));  di = d + (h / 2) \* (dD\_dt(s, e, i, r, d) + dD\_dt(s1, e1, i1, r1, d1));  // std::cout << k << ": Si = " << si << "; Ei = " << ei << "; Ii = " << ii << "; Ri = " << ri << "; Di = " << di << std::endl;  s = s1;  e = e1;  i = i1;  r = r1;  d = d1;  // std::cout << k << ": S = " << s << "; E = " << e << "; I = " << i << "; R = " << r << "; D = " << d << std::endl;  }  \*S = s;  \*E = e;  \*I = i;  \*R = r;  \*D = d;  }  int main()  {  std::cout.precision(12);  std::cout.setf(std::ios::fixed);  // начальные данные  int a = 0, b = 90;  double eps = 1e-2, h = 1;  double e = E0, i = 0, r = R0, d = 0, s = N0 - i - e - r - d;  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;  euler\_modified(a, b, h, &s, &e, &i, &r, &d); // первая итерация  double delta, d1 = d;  int k = 1;  do // цикл до заданной точности решения  {  h = h / 2;  e = E0, i = 0, r = R0, d = 0, s = N0 - i - e - r - d;  euler\_modified(a, b, h, &s, &e, &i, &r, &d);  delta = fabs(d - d1);  d1 = d;  std::cout << k << ": delta = " << delta << ", h = " << h << std::endl;  k++;  } while (delta > eps);  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;  double n = s + e + i + r + d;  std::cout << "N (final) = " << (int)round(n) << " = N0 (начальное население) => ни один человек не потерян\n\n";  return 0;  } |

**РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

Скриншот:

