**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет прикладной математики и физики

Кафедра вычислительной математики и программирования

**Лабораторная работа № 4**

по курсу «Численные методы»

Студент: Гаврилов М.С.

Группа: 80-306б

Преподаватель: Ревизников Д.Л.

Оценка:

Москва, 2022

1. **Постановка задачи**  
   1. Реализовать методы Эйлера, Рунге-Кутты и Адамса 4-го порядка в виде программ, задавая в качестве входных данных шаг сетки . С использованием разработанного программного обеспечения решить задачу Коши для ОДУ 2-го порядка на указанном отрезке. Оценить погрешность численного решения с использованием метода Рунге – Ромберга и путем сравнения с точным решением.

|  |  |
| --- | --- |
| , |  |

2. Реализовать метод стрельбы и конечно-разностный метод решения краевой задачи для ОДУ в виде программ. С использованием разработанного программного обеспечения решить краевую задачу для обыкновенного дифференциального уравнения 2-го порядка на указанном отрезке. Оценить погрешность численного решения с использованием метода Рунге – Ромберга и путем сравнения с точным решением.

|  |  |
| --- | --- |
| (2x+1) y″+4xy′-4y=0,  y′ (0)= –1,  y′(1)+2y(1)=3 |  |

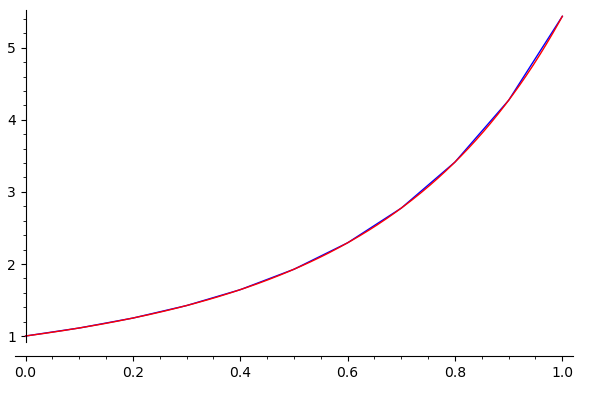
1. **Выполнение работы**

Задание 1.

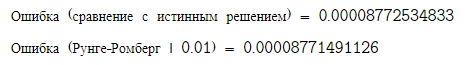
Функция, реализующая алгоритм Рунге-Кутты 4-го порядка:

|  |
| --- |
| **def** runge\_kutta\_4**(**x0**,**xl**,**y0**,**z0**,**h**,**eps **=** 0.001**):**  xcur **=** x0 #идем из 0 с шагом h  ycur **=** y0  zcur **=** z0  result **=** **[(**xcur**,**ycur**,**zcur**)]**  **while** xcur **+** h **<** xl **+** eps**:**  K1 **=** h**\***fy**(**x **=** xcur**,**y **=** ycur**,** z **=** zcur**)**  L1 **=** h**\***fz**(**x **=** xcur**,**y **=** ycur**,** z **=** zcur**)**  K2 **=** h**\***fy**(**x **=** xcur **+** 0.5**\***h**,**y **=** ycur **+** 0.5**\***K1**,** z **=** zcur **+** 0.5**\***L1**)**  L2 **=** h**\***fz**(**x **=** xcur **+** 0.5**\***h**,**y **=** ycur **+** 0.5**\***K1**,** z **=** zcur **+** 0.5**\***L1**)**  K3 **=** h**\***fy**(**x **=** xcur **+** 0.5**\***h**,**y **=** ycur **+** 0.5**\***K2**,** z **=** zcur **+** 0.5**\***L2**)**  L3 **=** h**\***fz**(**x **=** xcur **+** 0.5**\***h**,**y **=** ycur **+** 0.5**\***K2**,** z **=** zcur **+** 0.5**\***L2**)**  K4 **=** h**\***fy**(**x **=** xcur **+** h**,**y **=** ycur **+** K3**,** z **=** zcur **+** L3**)**  L4 **=** h**\***fz**(**x **=** xcur **+** h**,**y **=** ycur **+** K3**,** z **=** zcur **+** L3**)**  dy **=** **(**1**/**6**)\*(**K1 **+** 2**\***K2 **+** 2**\***K3 **+** K4**)**  dz **=** **(**1**/**6**)\*(**L1 **+** 2**\***L2 **+** 2**\***L3 **+** L4**)**  xcur **=** xcur **+** h  ycur **=** ycur **+** dy  zcur **=** zcur **+** dz  result**.**append**((**xcur**.**n**(),**ycur**.**n**(),**zcur**.**n**()))**  **return** result |

Решение задачи методом Рунге-Кутты с шагом 0.1 (синее), наложенное на истинное решение (красное)



Погрешность метода РК4, оцененная с помощью метода Рунге-Ромберга, а также с помощью сравнения с истинным решением:



Ошибки вычислялись путем отыскания среднего модуля разности получившейся на функции и истинного решения (или более точного решения) в точках, где она определена.

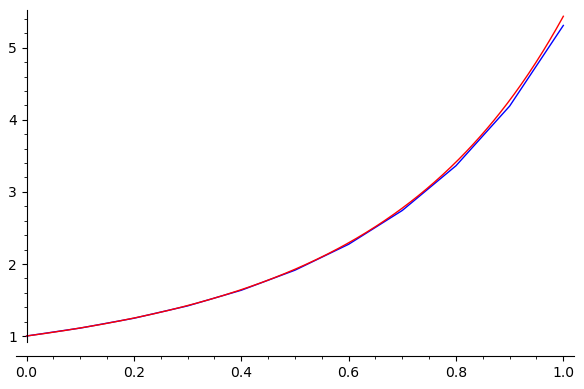
Функция, реализующая метод Эйлера (первый улучшенный, 2 порядка)

|  |
| --- |
| **def** euler**(**x0**,**xl**,**y0**,**z0**,**h**,**eps **=** 0.01**):** #первый улучшенный 2 порядок  xcur **=** x0 #идем из 0 с шагом h  ycur **=** y0  zcur **=** z0  result **=** **[(**xcur**,**ycur**,**zcur**)]**  **while** xcur **+** h **<** xl **+** eps**:**  dhx **=** h**/**2  dhy **=** **(**h**/**2**)\***fy**(**x **=** xcur**,**y **=** ycur**,** z **=** zcur**)**  dhz **=** **(**h**/**2**)\***fz**(**x **=** xcur**,**y **=** ycur**,** z **=** zcur**)**    dy **=** h**\***fy**(**x **=** xcur **+** dhx**,**y **=** ycur **+** dhy**,** z **=** zcur **+** dhz**)**  dz **=** h**\***fz**(**x **=** xcur **+** dhx**,**y **=** ycur **+** dhy**,** z **=** zcur **+** dhz**)**  xcur **=** xcur **+** h  ycur **=** ycur **+** dy  zcur **=** zcur **+** dz  result**.**append**((**xcur**.**n**(),**ycur**.**n**(),**zcur**.**n**()))**  **return** result |

Функция, реализующая метод Эйлера-Коши

|  |
| --- |
| **def** euler\_koshi**(**x0**,**xl**,**y0**,**z0**,**h**,**eps **=** 0.01**):** #Эйлер - Коши | 2 порядок  xcur **=** x0 #идем из 0 с шагом h  ycur **=** y0  zcur **=** z0  result **=** **[(**xcur**,**ycur**,**zcur**)]**  **while** xcur **+** h **<** xl **+** eps**:**  dx **=** h  estdy **=** **(**h**)\***fy**(**x **=** xcur**,**y **=** ycur**,** z **=** zcur**)**  estdz **=** **(**h**)\***fz**(**x **=** xcur**,**y **=** ycur**,** z **=** zcur**)**    dy **=** **(**h**/**2**)\*(**fy**(**x **=** xcur**,**y **=** ycur**,** z **=** zcur**)** **+** fy**(**x **=** xcur **+** dx**,**y **=** ycur **+** estdy**,** z **=** zcur **+** estdz**))**  dz **=** **(**h**/**2**)\*(**fz**(**x **=** xcur**,**y **=** ycur**,** z **=** zcur**)** **+** fz**(**x **=** xcur **+** dx**,**y **=** ycur **+** estdy**,** z **=** zcur **+** estdz**))**  xcur **=** xcur **+** h  ycur **=** ycur **+** dy  zcur **=** zcur **+** dz  result**.**append**((**xcur**.**n**(),**ycur**.**n**(),**zcur**.**n**()))**  **return** result |

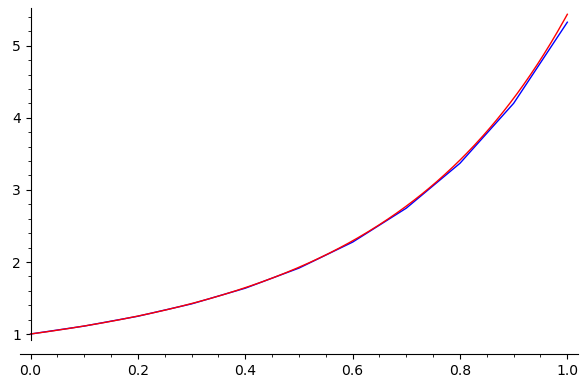
Решение, полученное улучшенным методом Эйлера, наложенное на истинное решение, шаг и цвета те же.



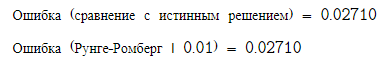
Его погрешность, оцененная методом Рунге-Ромберга, а также истинная ошибка:



Решение, полученное методом Эйлера-Коши, наложенное на истинное решение, шаг и цвета те же.



Его погрешность, оцененная методом Рунге-Ромберга, а также истинная ошибка:



Видно, что решения, полученные методами Эйлера, в отличие от решений метода РК4, имеют тенденцию «отваливаться» от истинного решения, и вообще менее точны. При этом Метод Эйлера-Коши точнее улучшенного метода Эйлера.

Функция, реализующая метод Адамса

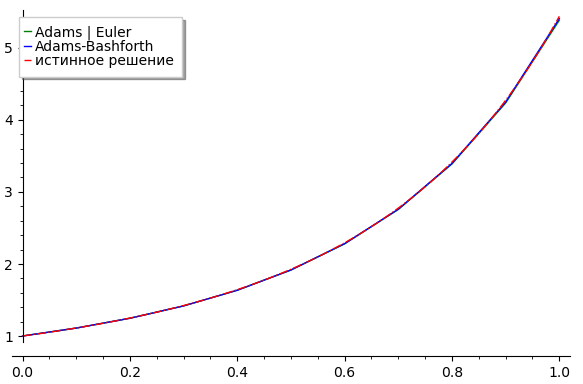
|  |
| --- |
| **def** adams**(**x0**,**xl**,**y0**,**z0**,**h**,**start **=** 0.4**,**start\_method **=** euler**,**eps **=** 0.01**):**    startlen **=** **(**start**\*(**xl **-** x0**))**  **if(**startlen**/**h **<** 4**):** #надо пройти хотя бы 4 точки  startlen **=** h**\***4      result **=** start\_method**(**x0**,**startlen**,**y0**,**z0**,**h**)** #старт    l **=** **len(**result**)**    xcur **=** result**[**l**-**1**][**0**]**  ycur **=** result**[**l**-**1**][**1**]**  zcur **=** result**[**l**-**1**][**2**]**    **while** xcur **+** h **<=** xl **+** eps**:**  l **=** **len(**result**)**    fyk1 **=** fy**(**x **=** result**[**l**-**1**][**0**],**y **=** result**[**l**-**1**][**1**],** z **=** result**[**l**-**1**][**2**])**  fzk1 **=** fz**(**x **=** result**[**l**-**1**][**0**],**y **=** result**[**l**-**1**][**1**],** z **=** result**[**l**-**1**][**2**])**    fyk2 **=** fy**(**x **=** result**[**l**-**2**][**0**],**y **=** result**[**l**-**2**][**1**],** z **=** result**[**l**-**2**][**2**])**  fzk2 **=** fz**(**x **=** result**[**l**-**2**][**0**],**y **=** result**[**l**-**2**][**1**],** z **=** result**[**l**-**2**][**2**])**    fyk3 **=** fy**(**x **=** result**[**l**-**3**][**0**],**y **=** result**[**l**-**3**][**1**],** z **=** result**[**l**-**3**][**2**])**  fzk3 **=** fz**(**x **=** result**[**l**-**3**][**0**],**y **=** result**[**l**-**3**][**1**],** z **=** result**[**l**-**3**][**2**])**    fyk4 **=** fy**(**x **=** result**[**l**-**4**][**0**],**y **=** result**[**l**-**4**][**1**],** z **=** result**[**l**-**4**][**2**])**  fzk4 **=** fz**(**x **=** result**[**l**-**4**][**0**],**y **=** result**[**l**-**4**][**1**],** z **=** result**[**l**-**4**][**2**])**    dy **=** h**/(**24**)\*(**55**\***fyk1 **-** 59**\***fyk2 **+** 37**\***fyk3 **-** 9**\***fyk4**)**  dz **=** h**/(**24**)\*(**55**\***fzk1 **-** 59**\***fzk2 **+** 37**\***fzk3 **-** 9**\***fzk4**)**    xcur **=** xcur **+** h  ycur **=** ycur **+** dy  zcur **=** zcur **+** dz  result**.**append**((**xcur**.**n**(),**ycur**.**n**(),**zcur**.**n**()))**  **return** result |

Функция, реализующая улучшенный метод Адамса

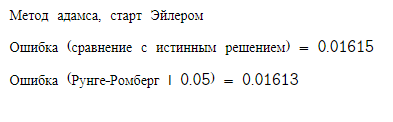
|  |
| --- |
| **def** adams\_bashforth**(**x0**,**xl**,**y0**,**z0**,**h**,**start **=** 0.4**,**start\_method **=** euler**,**eps **=** 0.01**):**    startlen **=** **(**start**\*(**xl **-** x0**))**  **if(**startlen**/**h **<** 4**):** #надо пройти хотя бы 4 точки  startlen **=** h**\***4      result **=** start\_method**(**x0**,**startlen**,**y0**,**z0**,**h**)** #старт    l **=** **len(**result**)**    xcur **=** result**[**l**-**1**][**0**]**  ycur **=** result**[**l**-**1**][**1**]**  zcur **=** result**[**l**-**1**][**2**]**    **while** xcur **+** h **<=** xl **+** eps**:**  l **=** **len(**result**)**    fyk1 **=** fy**(**x **=** result**[**l**-**1**][**0**],**y **=** result**[**l**-**1**][**1**],** z **=** result**[**l**-**1**][**2**])**  fzk1 **=** fz**(**x **=** result**[**l**-**1**][**0**],**y **=** result**[**l**-**1**][**1**],** z **=** result**[**l**-**1**][**2**])**    fyk2 **=** fy**(**x **=** result**[**l**-**2**][**0**],**y **=** result**[**l**-**2**][**1**],** z **=** result**[**l**-**2**][**2**])**  fzk2 **=** fz**(**x **=** result**[**l**-**2**][**0**],**y **=** result**[**l**-**2**][**1**],** z **=** result**[**l**-**2**][**2**])**    fyk3 **=** fy**(**x **=** result**[**l**-**3**][**0**],**y **=** result**[**l**-**3**][**1**],** z **=** result**[**l**-**3**][**2**])**  fzk3 **=** fz**(**x **=** result**[**l**-**3**][**0**],**y **=** result**[**l**-**3**][**1**],** z **=** result**[**l**-**3**][**2**])**    fyk4 **=** fy**(**x **=** result**[**l**-**4**][**0**],**y **=** result**[**l**-**4**][**1**],** z **=** result**[**l**-**4**][**2**])**  fzk4 **=** fz**(**x **=** result**[**l**-**4**][**0**],**y **=** result**[**l**-**4**][**1**],** z **=** result**[**l**-**4**][**2**])**    dpy **=** h**/(**24**)\*(**55**\***fyk1 **-** 59**\***fyk2 **+** 37**\***fyk3 **-** 9**\***fyk4**)**  dpz **=** h**/(**24**)\*(**55**\***fzk1 **-** 59**\***fzk2 **+** 37**\***fzk3 **-** 9**\***fzk4**)**      fpy **=** fy**(**x **=** xcur **+** h**,** y **=** ycur **+** dpy**,**z **=** zcur **+** dpz**)**  fpz **=** fz**(**x **=** xcur **+** h**,** y **=** ycur **+** dpy**,**z **=** zcur **+** dpz**)**    dy **=** **(**h**/**24**)\*(**9**\***fpy **+** 19**\***fyk1 **-** 5**\***fyk2 **+** fyk3**)**  dz **=** **(**h**/**24**)\*(**9**\***fpz **+** 19**\***fzk1 **-** 5**\***fzk2 **+** fzk3**)**  xcur **=** xcur **+** h  ycur **=** ycur **+** dy  zcur **=** zcur **+** dz  result**.**append**((**xcur**.**n**(),**ycur**.**n**(),**zcur**.**n**()))**  **return** result |

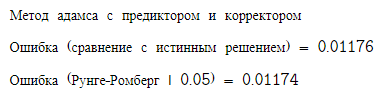
Для старта метода Адамса использовался метод Эйлера. Количество стартовых точек задается частью решения, которая должна быть пройдена стартующим методом, как аргумент функции. Если точек в этой части меньше четырех, вычисляются толь четыре первые точки.

Решения обоими методами и истинное решение, на одном графике:



Погрешности методов:





Как и ожидалось, метод Адамса с предиктором и корректором точнее, чем обычный метод Адамса. Также, оба метода точнее метода Эйлера, которым осуществлялся старт.

Задание 2.

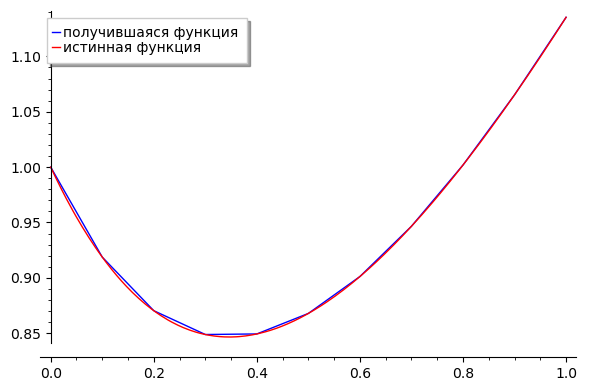
Функция, реализующая метод стрельбы:

|  |
| --- |
| **def** getlst**(**y0**,**h**):** #получить поледние значения для данного y0  res **=** runge\_kutta\_4**(**a**,**b**,**y0**,**z0**,**h**)**  **return** **(**res**[len(**res**)** **-** 1**][**1**],**res**[len(**res**)** **-** 1**][**2**])**  **def** getdiff**(**y0**,**h **=** 0.1**):** #получить отклонение от искомого значения  res **=** getlst**(**y0**,**h**)**  **return** test**(**yl **=** res**[**0**],**zl **=** res**[**1**])**  **def** shooting\_method**(**h**,** EPS **=** 0.01**,**trace **=** **False):**  rs **=** crappy\_newton**(**getdiff**,**T0**,**T1**,** EPS **=** EPS**,**trace**=** trace**,**h **=** h**)**  res **=** runge\_kutta\_4**(**a**,**b**,**rs**[**0**],-**1**,**h**)**  **return** res |

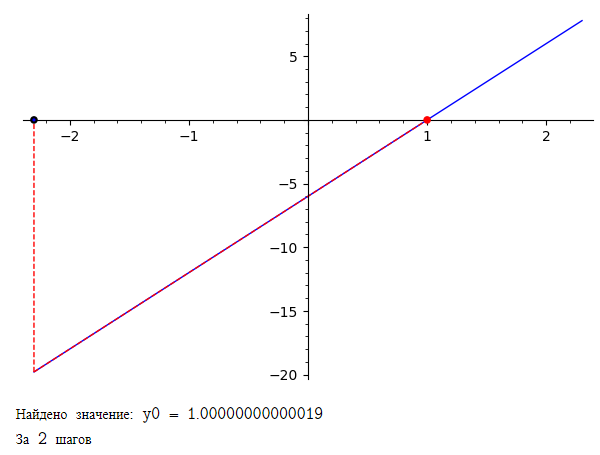
«Стрельба» производится методом РК4, «Пристраивание» – методом Ньютона с численным расчетом производной.

Решение, полученное методом, наложенное на истинное решение:

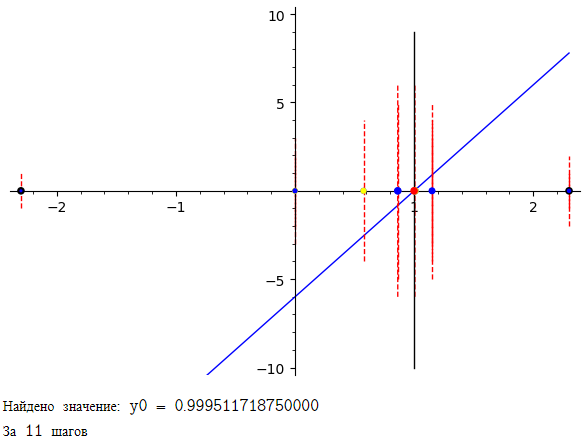
Шаг РК4 – 0.1



Графический трейсинг вычислений метода Ньютона:

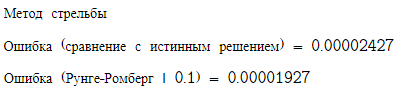


Трейсинг вычислений при пристреливании методом дихотомии:



Метод дихотомии требует существенно больше «выстрелов», и был использован из-за того, что его реализация у меня уже была, и его можно было применить без каких-либо модификаций к любой функции, и он был идеален для первичной проверки метода стрельбы на работоспособность.

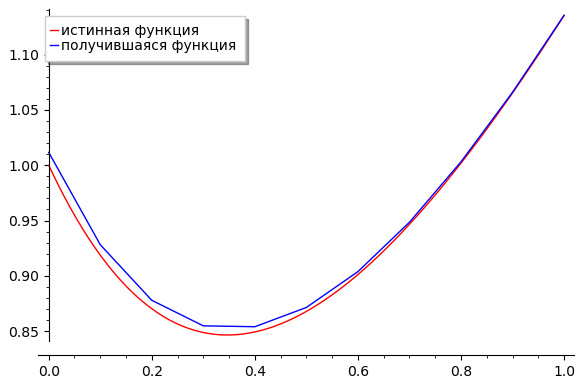
Погрешность метода:



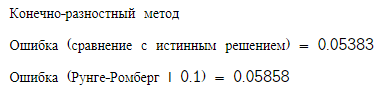
Функция, реализующая конечно-разностный метод:

|  |
| --- |
| **def** finite\_differ**(**p**,**q**,**f**,**a**,**b**,**ya**,**yb**,**h **=** 0.1**):**  l **=** b **-** a  ny **=** **int(**l**/**h **+** 1**)**#сколько делаем шагов  names **=** **[]**  **for** i **in** **range(int(**ny**)):**  names**.**append**(**"y" **+** **str(**i**))**  y **=** var**(**names**)** #задание переменных  xcur **=** a  eqs **=** **[]**  eqs**.**append**((-**3**\***y**[**0**]** **+** 4**\***y**[**1**]** **-** y**[**2**])/(**2**\***h**)** **==** ya**)**  **for** i **in** **range(**1**,** ny**-**1**):**  xcur **+=** h  eqs**.**append**((**y**[**i**+**1**]** **-** 2**\***y**[**i**]** **+** y**[**i**-**1**])/(**h**\*\***2**)** **+** p**(**x **=** xcur**)\*((**y**[**i**+**1**]** **-** y**[**i**-**1**])/(**2**\***h**))** **+** q**(**x **=** xcur**)\***y**[**i**]** **==** f**(**x **=** xcur**))**  xcur **+=** h  eqs**.**append**((**y**[**ny**-**3**]** **-** 4**\***y**[**ny**-**2**]** **+** 3**\***y**[**ny**-**1**])/(**2**\***h**)** **+** 2**\***y**[**ny**-**1**]** **==** yb**)**  sols **=** solve**(**eqs**,**y**)**  Y **=** **[]**  **for** k **in** sols**[**0**]:**  Y**.**append**(**k**.**rhs**().**n**())**  res **=** **[]**  xcur **=** 0  **for** k **in** Y**:**  res**.**append**((**xcur**,**k**))**  xcur **+=** h  **return** res |

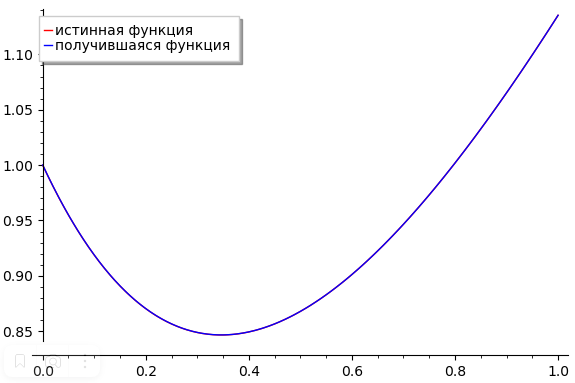
Полученное конечно-разностным методом решение (с шагом 0.1), наложенное на график истинного решения.



Оценка погрешности метода:



Несмотря на то, что на малых шагах решение слегка «отваливается» от истинного, с уменьшением шага эта проблема перестает проявляться столь ярко:

(h = 0.01)

1. **Вывод**

В ходе выполнения данной лабораторной работы, я получил опыт в реализации численных методов решения задачи Коши и краевой задачи. После курса уравнений математической физики было интересно взглянуть на то, как еще можно все это решать. Также из-за того, что я уже слышал о методе Рунге-Кутты решения дифференциальных уравнений, мне было интересно самостоятельно его реализовать, а также сравнить с другими методами решения задачи Коши, и убедиться, что он действительно один из самых точных. Здорово, что даже моя старая реализация дихотомии чуть-чуть пригодилась.