Практическая работа по курсу "Введение в численные методы"

Выполнил: Иванов Роман 209 группа

Постановка задачи:

Найти решение задачи Коши

$$\frac{du}{dx} = \int_0^x f(t)dt,\ u = u(x), 0 < x < l$$

методом Адамса второго порядка. Функция f(t) задана и может быть найдена как в точках сетки x = ih, i = 0,1, ... , N, h = L/N, так и в любой точке отрезка [0, l]
а) Исследовать поведение решения на сгущающихся сетках (при увеличении n).
б) Выяснить, будет ли сходимость.

Вариант 4

$$f(t) = e^{-(t-1)^2}$$

отрезок $[0,1]$, начальное $N = 20$

Метод Адамса:

Пусть $u(x)$ - решение дифференциального уравнения (27). Для производной этой функции имеет место равенство

$$u'(x) = f(x, u(x)) = F(x). \tag{86}$$

Интегрируя его между двумя точками сетки, получим соотношение

$$u_{i+1} = u_i + \int_{x_i}^{x_{i+1}} F(x)dx. \tag{87}$$

Мы не можем использовать это соотношение непосредственно для перехода в процессе решения задачи от i-ой точки сетки к $(i+1)$-ой, поскольку функция $F(x)$ нам не известна. Чтобы сделать следующий шаг, нужно приближенно заменить эту функцию на такую функцию, которую можно вычислить. Опишем, как эта проблема решается в методе Адамса.

Пусть в процессе численного решения задачи мы довели расчет до точки x_i. В результате проведенных расчетов нам оказались известными величины y_j и $f(x_j, y_j)$, $0 \le j \le i$. Возьмем некоторое фиксированное целое число $m \le i$ и построим интерполяционный многочлен m-ой степени, принимающий в точках x_j, $i-m \le j \le i$ значения $f(x_j, u_j)$

$$P_m(x_j) = f(x_j, u_j),\ i-m \le j \le i. \tag{88}$$

Его можно записать по формуле Лагранжа

$$P_m(x) = \sum_{j=i-m}^{i} f(x_j, y_j) Q_{m,j}(x), \tag{89}$$

где $Q_{m,j}(x)$ специальные многочлены вида

$$Q_{m,j}(x) = \frac{(x-x_{i-m})\cdots(x-x_{j-1})(x-x_{j+1})\cdots(x-x_i)}{(x_j-x_{i-m})\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_i)}, \tag{90}$$

Главная идея метода Адамса заключается в том, чтобы для расчета y_{i+1} использовать формулу типа (87), приближенно заменяя в ней функцию $F(x)$ на интерполяционный многочлен $P_m(x)$, составленный согласно (89) по результатам предыдущих вычислений. Это приводит к рекуррентной формуле

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} P_m(x)dx = y_i + \sum_{j=i-m}^{i} a_j f(x_j, y_j), \tag{91}$$

где

$$a_j = \int_{x_i}^{x_{i+1}} Q_{m,j}(x)dx. \tag{92}$$

Рассмотрим более подробно данную схему численного решения задачи Коши в простейших случаях $m=0$ и $m=1$, когда технические трудности не закрывают прозрачную идею метода. При $m=0$ для аппроксимации функции $F(x)$ используется полином нулевой степени, т. е. постоянная

$$F(x) \approx P_0 = f(x_i, y_i).$$

В этом случае формула (91) переходит в рекуррентную формулу метода Эйлера

$$y_{i+1} = y_i + hf\left(x_i, y_i\right),$$

обеспечивающую первый порядок точности. Такой результат сам по себе тривиален. Мы привели его только для того, чтобы показать, что для метода Адамса, как и для метода Рунге-Кутта, исходной точкой является схема Эйлера.

Перейдем к исследованию варианта $m=1$. В этом случае для аппроксимации функции $F(x)$ используется полином первой степени, построенный по значениям функции f в двух точках $\left(x_{i-1}, y_{i-1}\right)$ и $\left(x_i, y_i\right)$:

$$P_1(x) = f\left(x_i, y_i\right)\frac{x - x_{i-1}}{h} - f\left(x_{i-1}, y_{i-1}\right)\frac{x - x_i}{h}.$$

Подставляя его в формулу (91) и проводя интегрирование, получим

$$y_{i+1} = y_i + \left\{\frac{3}{2}f\left(x_i, y_i\right) - \frac{1}{2}f\left(x_{i-1}, y_{i-1}\right)\right\}h. \qquad (93)$$

Отметим следующую особенность рекуррентной формулы (93) Для расчета очередного значения сеточной функции yi+1 нужно знать ее значения в двух предыдущих точках yi и yi-1. Таким образом, формула (93) начинает работать только со второй точки. Вычислить по ней y1 нельзя. Это значение решения разностной задачи приходится вычислять каким-нибудь другим методом, например, методом Рунге-Кутта.

Программная реализация на python 3.6:

Рассмотрим вспомогательную функцию g(x) равную:

$$\int_0^x e^{-(t-1)^2}\,dt$$

Исходная задача Коши сводится к нахождению функции g(x) с последующим решением задачи Коши:
du/dx = g(x), u(0)=u0
Заметим, что dg/dx = e^(-(t-1)^2)
Тогда g(0), т.е.

$$\int_0^0 e^{-(t-1)^2}\,dt$$

равняется 0
Для нахождения функции g(x) необходимо решить задачу Коши вида: dg/dx = e^(-(t-1)^2), g(0)=0

Для этого задаём функцию:

$$f(t) = e^{-(t-1)^2}$$

Находим решение вспомогательной задачи Коши с помощью рекурентной формулы:

$$g_{i+1} = g_i + (\tfrac{3}{2}f_i - \tfrac{1}{2}f_{i-1})h$$

Находим решение исходной задачи Коши с помощью рекурентной формулы:

$$u_{i+1} = u_i + (\tfrac{3}{2}g_i - \tfrac{1}{2}g_{i-1})h$$

Инициализация параметров

In [1]:

```
import math
import matplotlib.pyplot as plt
import pandas as pd

N = 20
a = 0
b = 1
U0 = 0
```

Реализация функции

In [2]:

```
def func(t):
    return math.exp(-(t-1)**2)
```

Создание таблицы значений функции

In [3]:

```
def table_of_values(a, b, N):
    h = (b-a)/N

    mas_point = [[0] * 2 for i in range(N)]

    for i in range(N):
        mas_point[i][0] = a + h*i
        mas_point[i][1] = func(a + h*i)

    return mas_point
```

In [4]:

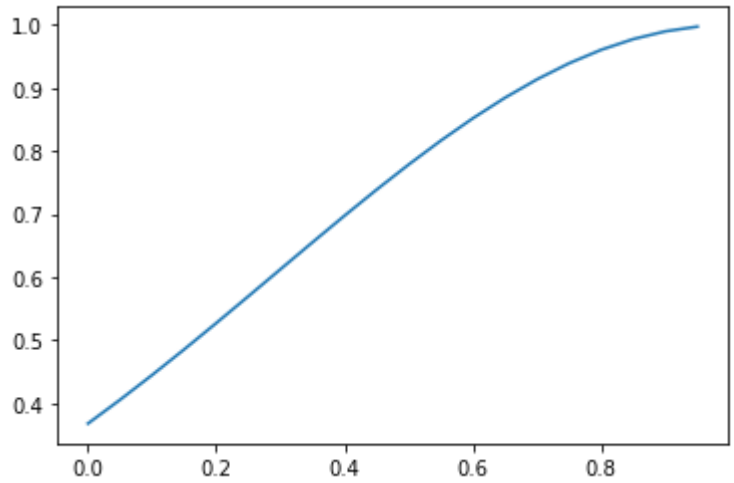
```
TableXY = table_of_values(a, b, N)
pd.DataFrame(TableXY, columns=["x", "y"])
```

Out[4]:

	x	y
0	0.00	0.367879
1	0.05	0.405555
2	0.10	0.444858
3	0.15	0.485537
4	0.20	0.527292
5	0.25	0.569783
6	0.30	0.612626
7	0.35	0.655406
8	0.40	0.697676
9	0.45	0.738968
10	0.50	0.778801
11	0.55	0.816686
12	0.60	0.852144
13	0.65	0.884706
14	0.70	0.913931
15	0.75	0.939413
16	0.80	0.960789
17	0.85	0.977751
18	0.90	0.990050
19	0.95	0.997503

In [5]:

```
x = [TableXY[i][0] for i in range(N)]
y = [TableXY[i][1] for i in range(N)]

plt.plot(x, y)
plt.show()
```



Реализация метода Рунге-Кутта для вычисления одной точки

In [6]:

```
def Runge1(mas, a, b, N, U0):
    h = (b-a)/N
    return U0 + h/2*(mas[0][1] + mas[1][1])
```

Реализация метода Адамса

In [7]:

```
def Adams(mas, a, b, N, U0):
    h = (b-a)/N
    answer = [U0]
    answer.append(Runge1(mas, a, b, N, U0))

    for i in range(N-2):
        answer.append(answer[i+1] + h*(1.5*mas[i+1][1] - 0.5*mas[i][1]))

    return answer
```

In [8]:

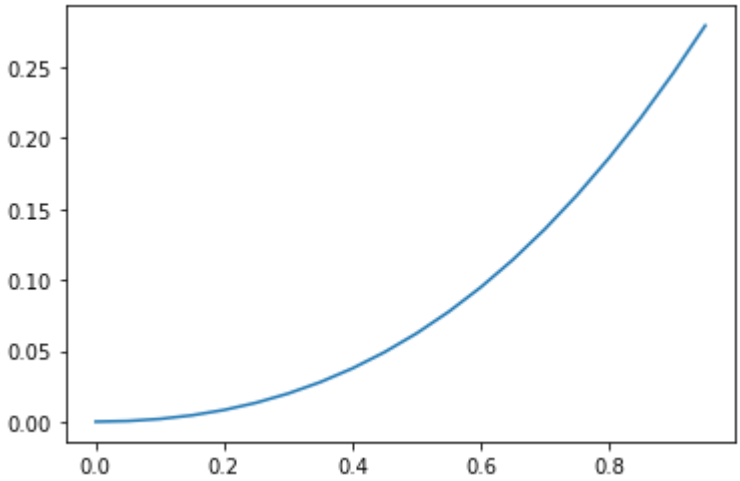
```
y = Adams(TableXY, a, b, N, U0)

for i in range(N):
    TableXY[i][1] = y[i]

y = Adams(TableXY, a, b, N, U0)

for i in range(N):
    TableXY[i][1] = y[i]

pd.DataFrame(TableXY, columns=["x", "y"])
```

Out[8]:

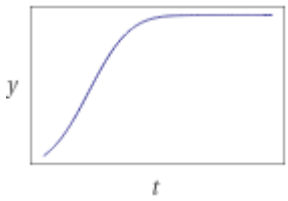
	x	y
0	0.00	0.000000
1	0.05	0.000483
2	0.10	0.001934
3	0.15	0.004492
4	0.20	0.008262
5	0.25	0.013348
6	0.30	0.019857
7	0.35	0.027898
8	0.40	0.037577
9	0.45	0.049002
10	0.50	0.062276
11	0.55	0.077501
12	0.60	0.094771
13	0.65	0.114176
14	0.70	0.135799
15	0.75	0.159713
16	0.80	0.185983
17	0.85	0.214663
18	0.90	0.245795
19	0.95	0.279412

In [9]:

```
plt.plot(x, y)
plt.show()
```



Аналитическое решение, полученное с помощью Wolfram Alpha



Исследование поведения решения на сгущающихся сетках

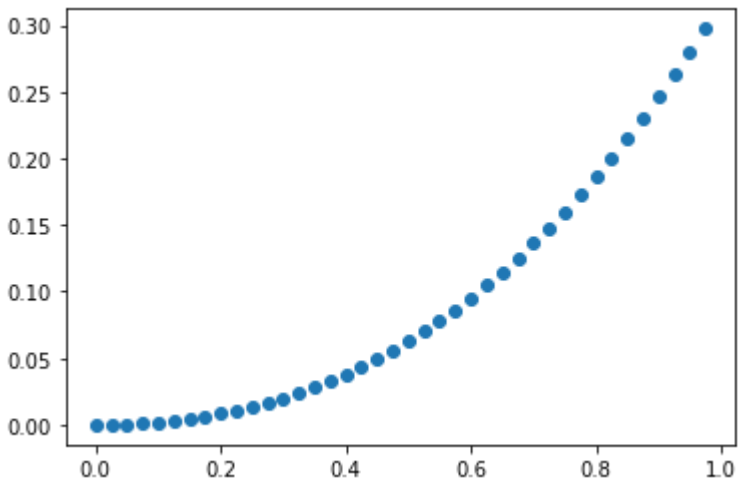
N = 40

In [10]:

```
N = 40

TableXY = table_of_values(a, b, N)

x = [TableXY[i][0] for i in range(N)]
y = Adams(TableXY, a, b, N, U0)

for i in range(N):
    TableXY[i][1] = y[i]

y = Adams(TableXY, a, b, N, U0)

for i in range(N):
    TableXY[i][1] = y[i]

plt.scatter(x, y)
```

Out[10]:

```
<matplotlib.collections.PathCollection at 0x2944fb95d30>
```



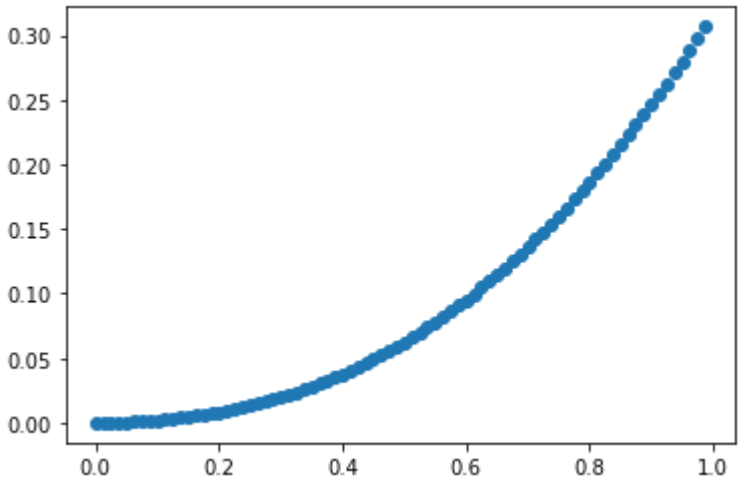
N = 80

In [11]:

```
N = 80

TableXY = table_of_values(a, b, N)

x = [TableXY[i][0] for i in range(N)]
y = Adams(TableXY, a, b, N, U0)

for i in range(N):
    TableXY[i][1] = y[i]

y = Adams(TableXY, a, b, N, U0)

for i in range(N):
    TableXY[i][1] = y[i]

plt.scatter(x, y)
```

Out[11]:

```
<matplotlib.collections.PathCollection at 0x2944fc29f40>
```



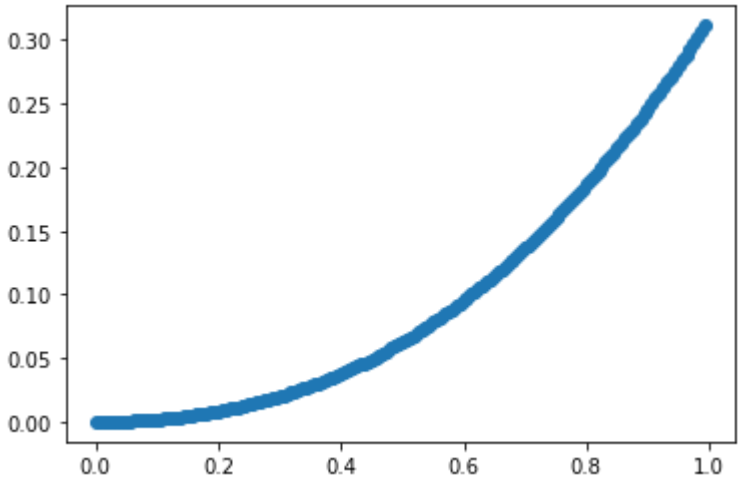
N = 160

In [12]:

```
N = 160

TableXY = table_of_values(a, b, N)

x = [TableXY[i][0] for i in range(N)]
y = Adams(TableXY, a, b, N, U0)

for i in range(N):
    TableXY[i][1] = y[i]

y = Adams(TableXY, a, b, N, U0)

for i in range(N):
    TableXY[i][1] = y[i]

plt.scatter(x, y)
```

Out[12]:

```
<matplotlib.collections.PathCollection at 0x2944fc8abe0>
```



Вывод

1) При увеличении числа точек, точность растет

1) Сходимость есть