Министерство образования и науки РФ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Лабораторная работа № 4

“**Методы Адамса**”

по дисциплине «Численное моделирование динамических систем, описываемых обыкновенными дифференциальными уравнениями»

**Факультет:** ПМИ

**Группа:**  ПМ-92

**Студенты:**  Иванов В., Кутузов И.

**Преподаватель:**  Вагин Д. В.

Новосибирск

2021

**Условие**

На трех сетках h = [0.1, 0.05, 0.025] решить задачу

y'=4ty

t=[0,1]

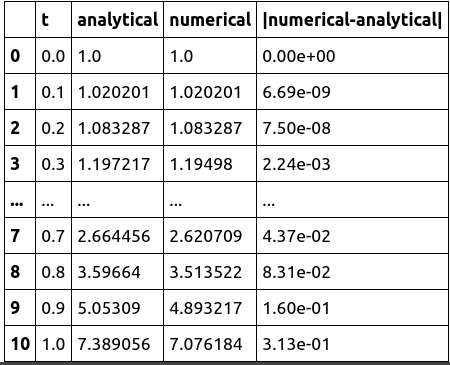
y(0)=1

с помощью схем:

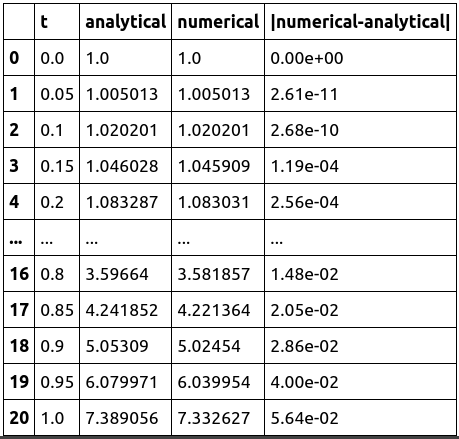
* явные и неявные методы Адамса 3-го и 4-го порядков
* метод прогноза-коррекции

**Явный Адамс, 3-й порядок**

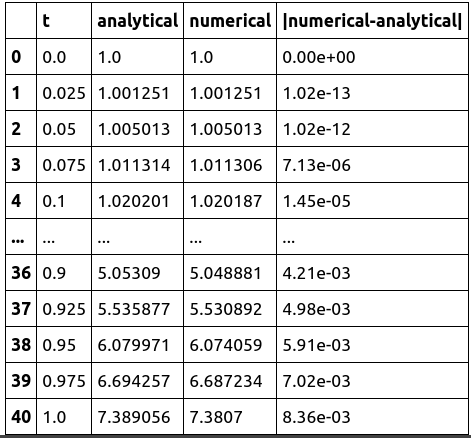
#### h = 0.1



#### h = 0.05

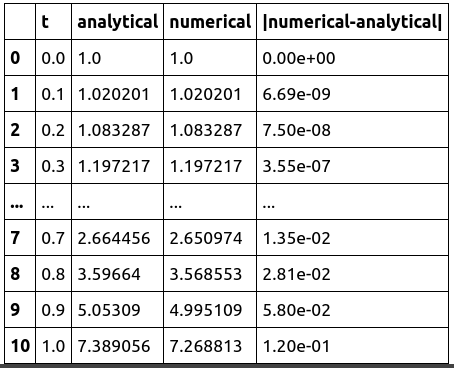


#### h = 0.025

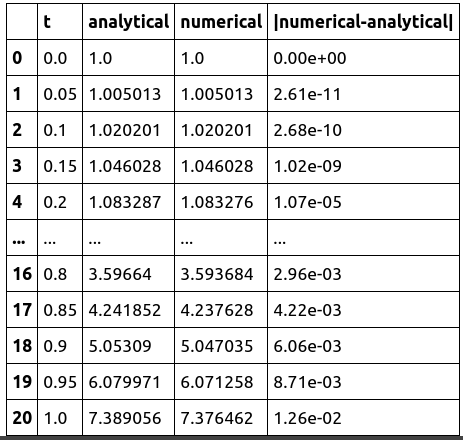
****

**Явный Адамс, 4-й порядок**

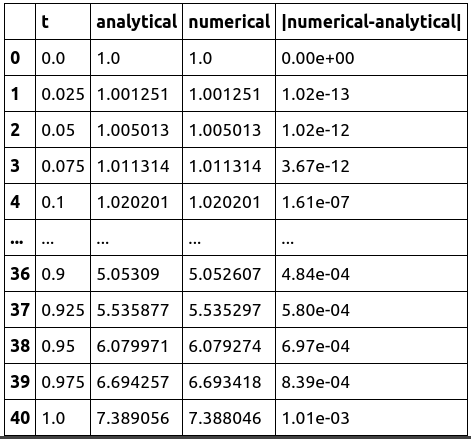
#### h = 0.1

****

#### h = 0.05

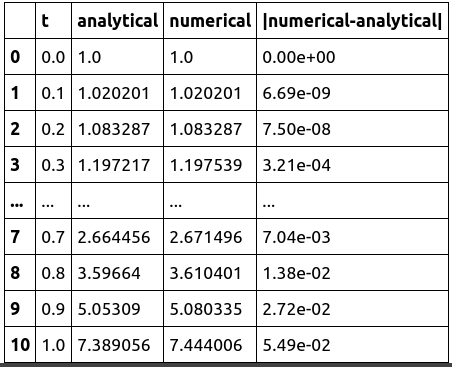


#### h = 0.025

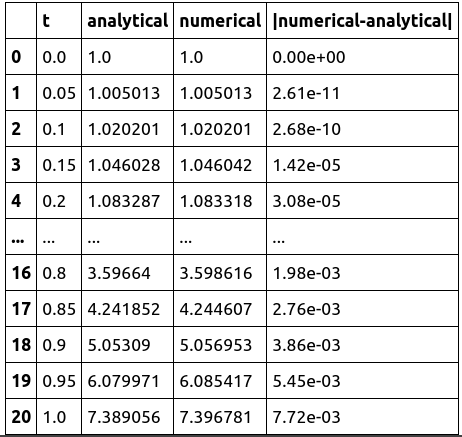
****

**Неявный Адамс, 3-й порядок**

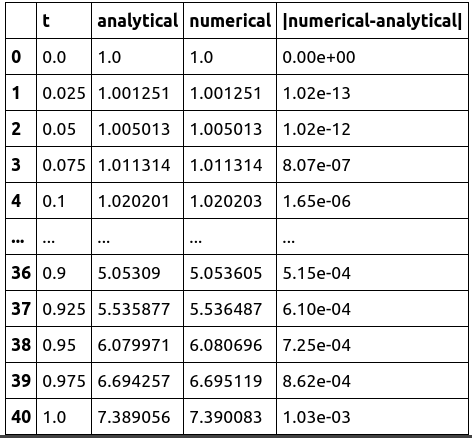
#### h = 0.1

****

#### h = 0.05

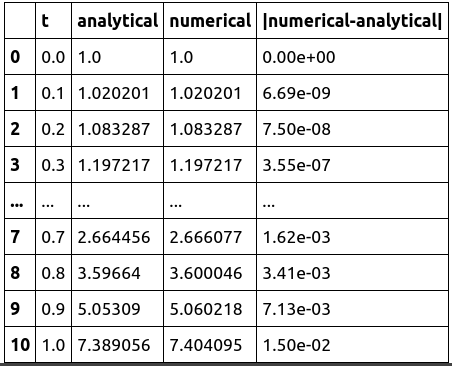


#### h = 0.025

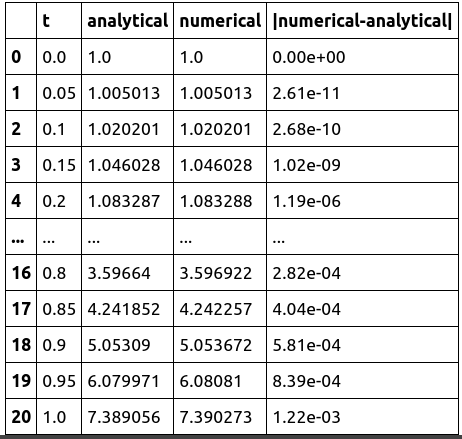
****

**Неявный Адамс, 4-й порядок**

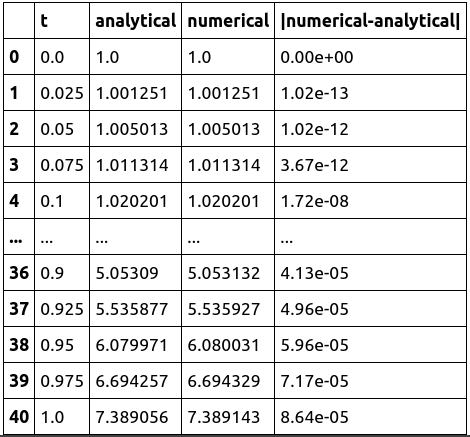
#### h = 0.1

****

#### h = 0.05

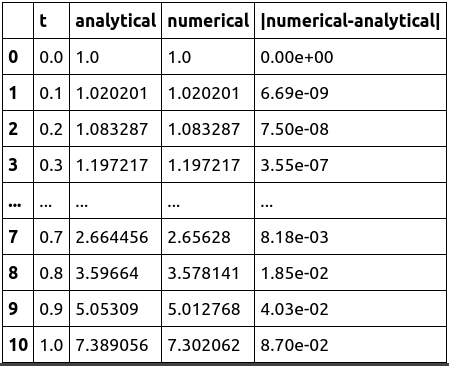


#### h = 0.025

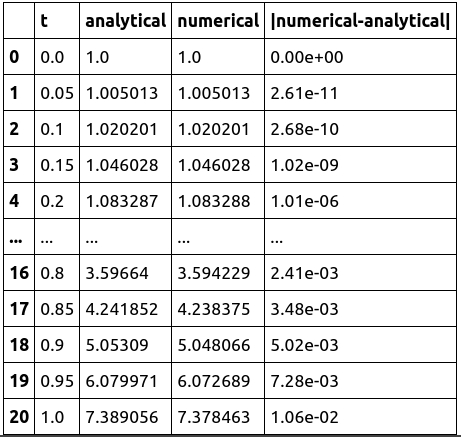
****

**Прогноз-коррекция, 4-й порядок**

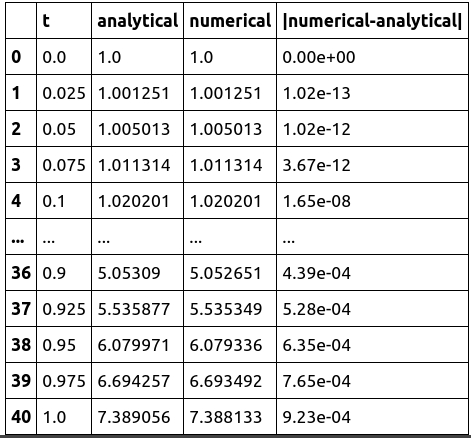
#### h = 0.1

****

#### h = 0.05



#### h = 0.025

****

**Код программы**

**import numpy as np**

**import pandas as pd**

**from matplotlib import pyplot as plt**

**interval = [0, 1]**

**def make\_grid(h):**

**return np.arange(interval[0], interval[1], h)**

**def analytical(h):**

**analytical = []**

**grid = make\_grid(h)**

**for tn in grid:**

**yn\_1 = f\_solve(tn)**

**analytical.append(yn\_1)**

**analytical.append(f\_solve(grid[-1]+grid[1]))**

**return analytical**

**def make\_results(grid, analytical, numerical):**

**error = np.absolute(np.array(numerical) - np.array(analytical))**

**error\_scientific = []**

**for x in error:**

**error\_scientific.append('{:.2e}'.format(x))**

**df = pd.DataFrame([grid, analytical, numerical, error\_scientific]).T**

**df\_middle = pd.DataFrame(['...', '...', '...', '...']).T.rename(index={0: '...'})**

**df\_middle.columns = ['t', 'analytical', 'numerical', '|numerical-analytical|']**

**df.columns = ['t', 'analytical', 'numerical', '|numerical-analytical|']**

**df.loc[len(grid), 't'] = grid[-1]+grid[1]**

**make\_plots(grid, analytical, numerical, error)**

**return pd.concat([df.head(), df\_middle, df.tail()])**

**def make\_plots(grid, analytical, numerical, error):**

**fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15,5))**

**grid = np.append(grid, grid[-1]+grid[1])**

**ax1.plot(error, grid,**

**marker='s', markersize=9, markerfacecolor='blue',**

**linestyle='-', linewidth=5,**

**color='cyan')**

**ax1.grid(color='grey', linestyle='dotted')**

**ax1.set\_xlabel('error', fontsize=15)**

**ax1.set\_ylabel('t', fontsize=15)**

**ax2.plot(grid, numerical, 'b-', label='numerical', alpha=0.50, linewidth=2)**

**ax2.plot(grid, analytical, 'g:', label='analytical', alpha=0.75, linewidth=5)**

**ax2.legend(loc='upper center', frameon=False)**

**ax2.set\_xlabel('y', fontsize=15)**

**ax2.set\_ylabel('t', fontsize=15)**

***# computes first n numerical values using classical runge-kutta method***

**def compute\_first\_elements(h, n):**

**yn = 1.0**

**elements = []**

**grid = make\_grid(h)**

**elements.append(yn)**

**for tn in grid[:n-1]:**

**kn1 = f(tn, yn)**

**kn2 = f(tn + h/2, yn + kn1\*h/2)**

**kn3 = f(tn + h/2, yn + kn2\*h/2)**

**kn4 = f(tn + h, yn + kn3\*h)**

**kn = (1/6)\*(kn1 + 2\*kn2 + 2\*kn3 + kn4)**

**yn\_new = yn + h\*kn**

**elements.append(yn\_new)**

**yn = yn\_new**

**return elements**

**def f(t, y):**

**return 4\*t\*y**

**def f\_solve(t):**

**return np.exp(2\*t\*t)**

**def explicit\_adams\_3(h):**

**numerical = []**

**grid = make\_grid(h)**

**yn\_2, yn\_1, yn = compute\_first\_elements(h, 3)**

**numerical.extend([yn\_2, yn\_1, yn]) *# append multiple elements***

**for tn in grid[2:]:**

**yn\_new = yn + (h/12)\*(23\*f(tn, yn)-16\*f(tn-h, yn\_1)+5\*f(tn-2\*h, yn\_2))**

**numerical.append(yn\_new)**

**yn\_2, yn\_1, yn = yn\_1, yn, yn\_new**

**return make\_results(grid, analytical(h), numerical)**

**def explicit\_adams\_4(h):**

**numerical = []**

**grid = make\_grid(h)**

**yn\_3, yn\_2, yn\_1, yn = compute\_first\_elements(h, 4)**

**numerical.extend([yn\_3, yn\_2, yn\_1, yn])**

**for tn in grid[3:]:**

**yn\_new = yn + (h/24)\*(55\*f(tn, yn)-59\*f(tn-h, yn\_1)+37\*f(tn-2\*h, yn\_2)-9\*f(tn-3\*h, yn\_3))**

**numerical.append(yn\_new)**

**yn\_3, yn\_2, yn\_1, yn = yn\_2, yn\_1, yn, yn\_new**

**return make\_results(grid, analytical(h), numerical)**

**def implicit\_adams\_3(h, eps):**

**numerical = []**

**grid = make\_grid(h)**

**yn\_2, yn\_1, yn = compute\_first\_elements(h, 3)**

**numerical.extend([yn\_2, yn\_1, yn])**

**for tn in grid[2:]:**

**yk = yn + (h/12)\*(23\*f(tn, yn)-16\*f(tn-h, yn\_1)+5\*f(tn-2\*h, yn\_2)) *# explicit adams***

**while True:**

**yk\_1 = yn + (h/12)\*(5\*f(tn+h, yk)+8\*f(tn, yn)-f(tn-h, yn\_1))**

**if abs(yk\_1-yk) < eps:**

**break**

**yk = yk\_1**

**yn\_new = yn + (h/12)\*(5\*f(tn+h, yk\_1)+8\*f(tn, yn)-f(tn-h, yn\_1))**

**numerical.append(yn\_new)**

**yn\_1, yn = yn, yn\_new**

**return make\_results(grid, analytical(h), numerical)**

**def implicit\_adams\_4(h, eps):**

**numerical = []**

**grid = make\_grid(h)**

**yn\_3, yn\_2, yn\_1, yn = compute\_first\_elements(h, 4)**

**numerical.extend([yn\_3, yn\_2, yn\_1, yn])**

**for tn in grid[3:]:**

**yk = yn + (h/24)\*(55\*f(tn, yn)-59\*f(tn-h, yn\_1)+37\*f(tn-2\*h, yn\_2)-9\*f(tn-3\*h, yn\_3))**

**while True:**

**yk\_1 = yn + (h/24)\*(9\*f(tn+h, yk)+19\*f(tn, yn)-5\*f(tn-h, yn\_1)+f(tn-2\*h, yn\_2))**

**if abs(yk\_1-yk) < eps:**

**break**

**yk = yk\_1**

**yn\_new = yn + (h/24)\*(9\*f(tn+h, yk\_1)+19\*f(tn, yn)-5\*f(tn-h, yn\_1)+f(tn-2\*h, yn\_2))**

**numerical.append(yn\_new)**

**yn\_3, yn\_2, yn\_1, yn = yn\_2, yn\_1, yn, yn\_new**

**return make\_results(grid, analytical(h), numerical)**

**def predictor\_corrector(h):**

**numerical = []**

**grid = make\_grid(h)**

**yn\_3, yn\_2, yn\_1, yn = compute\_first\_elements(h, 4)**

**numerical.extend([yn\_3, yn\_2, yn\_1, yn])**

**for tn in grid[3:]:**

**yn\_pred = yn + (h/24)\*(55\*f(tn, yn)-59\*f(tn-h, yn\_1)+37\*f(tn-2\*h, yn\_2)-9\*f(tn-3\*h, yn\_3)) *# explicit adams***

**yn\_corr = yn + (h/24)\*(9\*f(tn+h, yn\_pred)+19\*f(tn, yn)-5\*f(tn-h, yn\_1)+f(tn-2\*h, yn\_2)) *# implicit adams***

**numerical.append(yn\_corr)**

**yn\_3, yn\_2, yn\_1, yn = yn\_2, yn\_1, yn, yn\_pred**

**return make\_results(grid, analytical(h), numerical)**