Расчетно-графическая работа №2

Мамылин Дмитрий, МТ-301

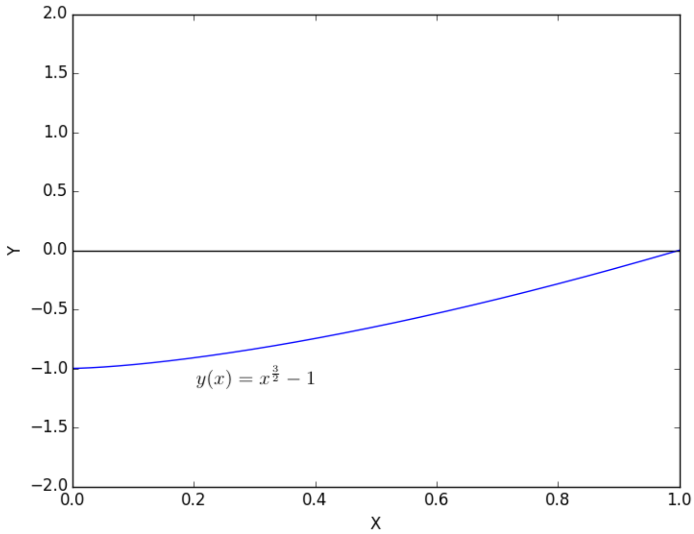
20 мая 2015 г.

Постановка задачи

Дана задача Коши:

$$y' = \frac{3}{2} \cdot \sqrt{x}$$
$$y(0) = -1$$

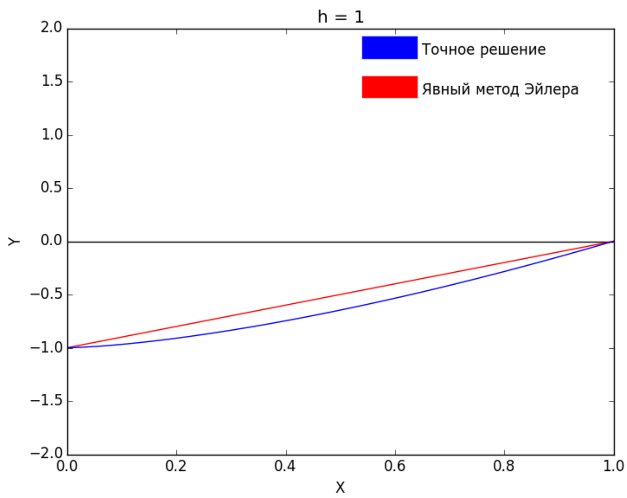
Будем решать задачу на отрезке $[0, 1]$. Зафиксируем на этом отрезке равномерную сетку $\{x_i\}_{i=0}^{n}$ с шагом h, где $x_0 = 0, x_1 = 1$.
Задачу необходимо решить явным методом Эйлера, неявным методом Эйлера и явным двушаговым методом Адамса.
Точное решение: $y(x) = x^{\frac{3}{2}} - 1$

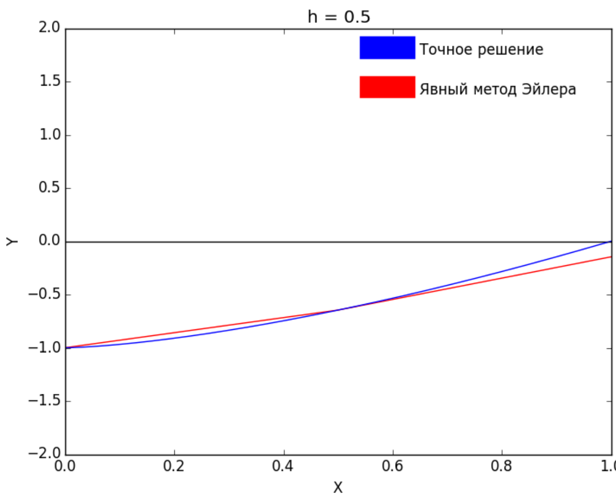


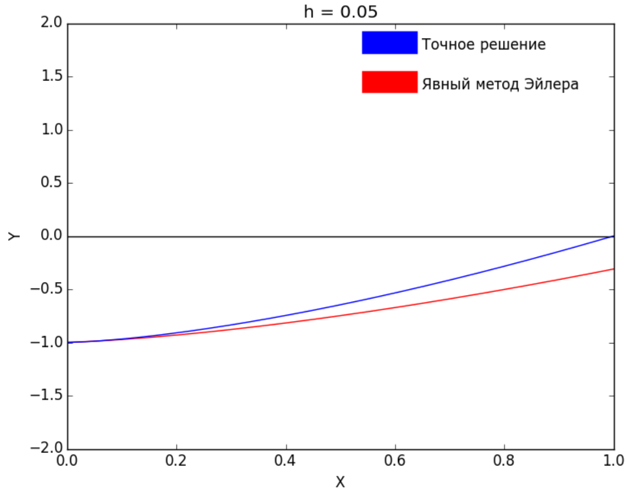
Решение явным методом Эйлера

Формула: $y_{i+1} = y_i + h \cdot \frac{3}{2} \cdot \sqrt{x_i}$, где h - шаг.

Явный метод Эйлера имеет порядок точности $O(h)$, поэтому для решения нелинейного уравнения: $z^2 - x_i = 0$ можно применить метод Ньютона, порядок точности которого $O(h^2)$.
$f_i(z) = z^2 - x_i$
$f_i'(z) = f'(z) = 2z$
Тогда метод примет вид: $z_{j+1} = z_j - \frac{f_i(z_j)}{f'(z_j)} = z_j - \frac{z_j^2 - x_i}{2z_j}$
Выберем начальную точку:
$f_i''(z) = f''(z) = 2 > 0$
То есть, надо выбирать z_0 так, чтобы выполнялось: $f_i(z_0) > 0$.
Поскольку $x_i \leq 1 \forall i$, то можно взять $z_0 = 1.1$, тогда: $f_i(x) > 0 \forall x \in [0, 1]$, что гарантирует сходимость.



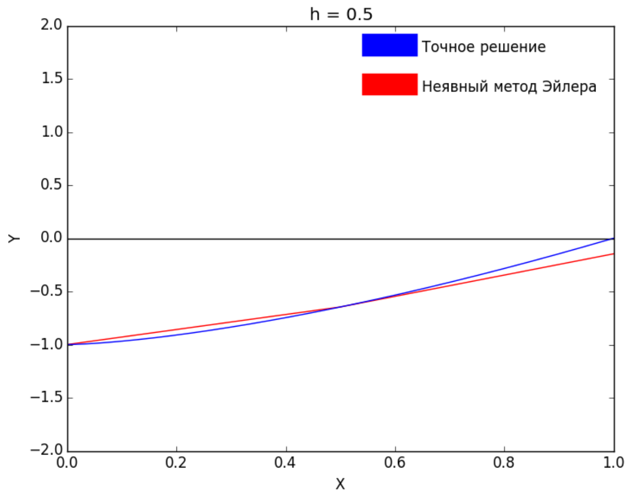


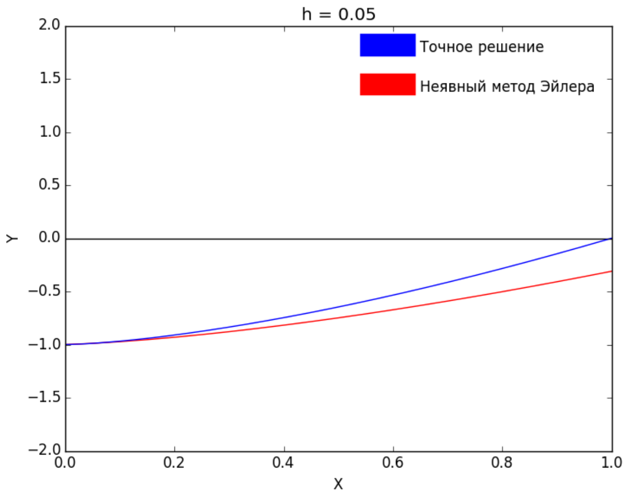


Решение неявным методом Эйлера

Формула: $y_{i+1} = y_i + h \cdot f_{i+1} = y_i + h \cdot \frac{3}{2} \cdot \sqrt{x_{i+1}}$, где h - шаг.

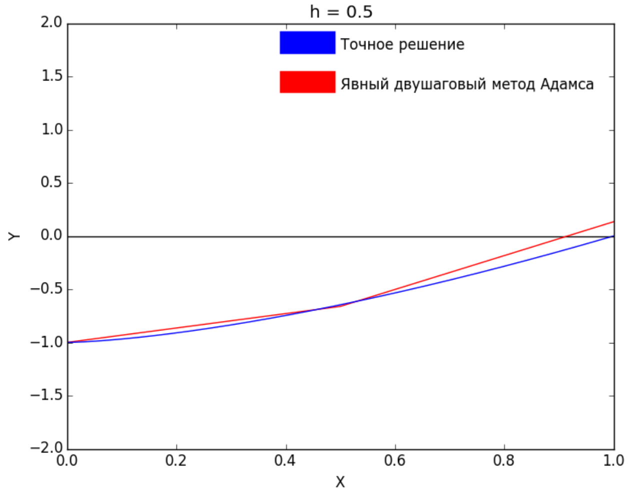
Аналогично явному методу Эйлера, неявный метод имеет порядок точности $O(h)$, поэтому для решения нелинейного уравнения: $z^2 - x_{i+1} = 0$ можно применить метод Ньютона.
В остальном рассуждения полностью повторяют предыдущий пункт.

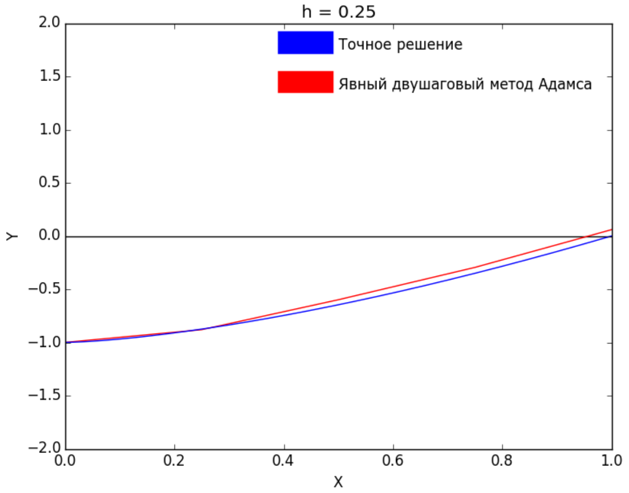


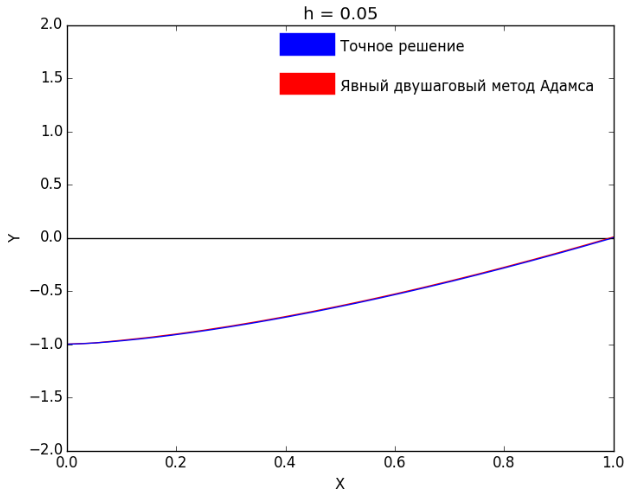


Решение явным двушаговым методом Адамса

Выведем формулу:
Интегральное представление точного решения: $y_{i+1} = y_i + \int_{x_i}^{xi+1} L_{k-1}(s)\,\mathrm{d}s$.
Запишем многочлен Лагранжа $L_{k-1}(x) = [\text{при } k = 2] = L_1(x)$ в форме Ньютона:
$L_1(x) = f_i + f(x_i, x_{i-1})(x - x_i) = f_i + \frac{f_i - f i-1}{h}(x - x_i)$.
Подставим в интегральное представление. После интегрирования и приведения подобных получим формулу: $y_{i+1} = y_i + \frac{h}{2}(3f_i - f_{i-1}) = y_i + \frac{h}{2}(\frac{9}{2}\sqrt{x_i} - \frac{3}{2}\sqrt{x_{i-1}})$, где $i = 1, 2, .., n-1$.
Будем проводить разгон методом Рунге-Кутты третьего порядка.
$y_1 = y_0 + \frac{1}{6}k_1 + \frac{1}{6}k_2 + \frac{4}{6}k_3$. Где:
$k_1 = hf(x_0, y_0) = h \cdot \frac{3}{2}\sqrt{x_0} = 0$
$k_2 = hf(x_0 + h, y_0 + k_1) = hf(h, y_0) = h \cdot \frac{3}{2}\sqrt{h}$
$k_3 = hf(x_0 + \frac{h}{2}, y_0 + \frac{1}{4}k_1 + \frac{1}{4}k_2) = hf(\frac{h}{2}, -1 + \frac{1}{4}k_2) = h \cdot \frac{3}{2}\sqrt{\frac{h}{2}}$
То есть: $y_1 = -1 + \frac{3h}{12}(\sqrt{h} + 4\sqrt{\frac{h}{2}}) = \frac{3h}{12}\sqrt{h}(1 + 4\sqrt{0.5}) - 1$.







Вывод

На данном примере методы Эйлера почти полностью повторяют друг друга, в то время как метод Адамса сходится быстрее, поскольку он является многошаговым, то есть при пересчете следующих значений учитываются предыдущие шаги.

Приложение

```
import math
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
from matplotlib.patches import Rectangle
import pylab

H = 0.5

def f(z, xi):
    return z**2 - xi

def f_1(z):
    return 2.0 * z

def newton(xi, z0, precision):
    z = z0
    while True:
        zprev = z
        z -= f(z, xi) / f_1(z)
        if abs(z - zprev) < precision:
            break
    return z

def explicitEuler(y0, grid):
    PRECISION = 0.0001
    Z = 1.1
    h = grid[1] - grid[0]
    result = [(grid[0], y0)]
    for i in range(len(grid)):
        yNext = result[i][1] + h * newton(grid[i], Z, PRECISION)
        result.append((grid[i], yNext))
    return result

def implicitEuler(y0, grid):
    PRECISION = 0.0001
    Z = 1.1
    h = grid[1] - grid[0]
    result = [(grid[0], y0)]
    for i in range(1, len(grid)):
        yNext = result[i-1][1] + h * newton(grid[i], Z, PRECISION)
        result.append((grid[i], yNext))
    return result

def twoStepAdams(y0, grid):
    PRECISION = 0.0001
    Z = 1.1
    y1 = (3*H / 12)*math.sqrt(H)*(1 + 4*math.sqrt(0.5)) - 1
    result = [(grid[0], y0), (grid[1], y1)]
    for i in range(2, len(grid)):
        yNext = result[i-1][1] + (H/2)*(newton(grid[i-1], Z, PRECISION)*9.0/2.0 -
                                        newton(grid[i-2], Z, PRECISION)*3.0/2.0)
        result.append((grid[i], yNext))
    return result
```