МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

ПАВЛОВИЧ ВЛАДИСЛАВ ВИКТОРОВИЧ

МЕТОДЫ ЧИСЛЕННОГО АНАЛИЗА

Отчет по лабораторной работе №2
студента 3 курса 3 группы

Преподаватель
Полещук Максим Александрович

Минск 2016

1 Исходное уравнение

$$\begin{cases} u'' = \dfrac{1}{x}u' + 15 \cdot x^2, \\ u(1) = 0, \\ u(2) = 1. \end{cases} \tag{1}$$

15 - номер в списке группы.

2 Задачи

2.1 Задание 1

2.1.1 Условие

Методом Галёркина при $n = 2$ найти приближённое решение следующей первой краевой задачи для ОДУ 2-го порядка (1) с точным решением

$$u(x) = \frac{1}{24}(x^2 - 1)(3 \cdot 10(x^2 - 4) + 8). \tag{2}$$

2.1.2 Отчётность

1. Конкретный вид весовых функций ϕ_0, ϕ_1, ϕ_2.
2. Коэффициенты матрицы (значения соответствующих определённых интегралов)

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$

   вектора правой части

$$D = \begin{pmatrix} D_1 \\ D_2 \end{pmatrix}$$

   и вектора приближенного решения $\hat{a} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$ системы линейных уравнений.

3. Максимальная глобальная относительная погрешность приближённого $\hat{u}(x)$ и точного $u(x)$ решений — величина

$$\max_{n=\overline{1,N-1}} \left| \frac{u(x_n) - \hat{u}(x)}{u(x_n)} \right|, x_n = a + nh, h = \frac{b-a}{N}, N = 20,$$

   где параметр N — число узлов сетки.

4. Графики приближённого $\hat{u}(x)$ и точного $u(x)$ решений на одном рисунке.

2.2 Задание 2

2.2.1 Условие

Методом Ритца при найти приближённое решение краевой задачи из задания 1.

2.2.2 Отчётность

1. Конкретный вид весовых функций ϕ_0, ϕ_1, ϕ_2.

2. Коэффициенты матрицы (значения соответствующих определённых интегралов)

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$

   вектора правой части

$$D = \begin{pmatrix} D_1 \\ D_2 \end{pmatrix}$$

   и вектора приближенного решения

$$\hat{a} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$$

   системы линейных уравнений.

3. Максимальная глобальная относительная погрешность приближённого $\hat{u}(x)$ и точного $u(x)$ решений — величина

$$\max_{n=\overline{1,N-1}} \left| \frac{u(x_n) - \hat{u}(x)}{u(x_n)} \right|, x_n = a + nh, h = \frac{b-a}{N}, N = 20,$$

   где параметр N — число узлов сетки.

2.3 Задание 3

2.3.1 Условие

Методом стрельбы с использованием «классического» метода Рунге-Кутты четвёртого порядка точности, реализованного в лабораторной работе 1, с шагом $h = 0.25$ и методом Ньютона уточнения корней нелинейных уравнений найти приближённое решение краевой задачи из задания 1. Принять абсолютную погрешность равной $\varepsilon = 0.05$.

2.3.2 Отчётность

1. Число потребовавшихся итераций метода Ньютона.

2. Абсолютная погрешность второго краевого условия.

3. Максимальная глобальная относительная погрешность приближённого $\hat{u}(x)$ и точного $u(x)$ решений — величина

$$\max_{n=\overline{1,N-1}} \left| \frac{u(x_n) - \hat{u}(x)}{u(x_n)} \right|, x_n = a + nh, h = \frac{b-a}{N}, N = 20,$$

   где параметр N — число узлов сетки.

3 Отчёт

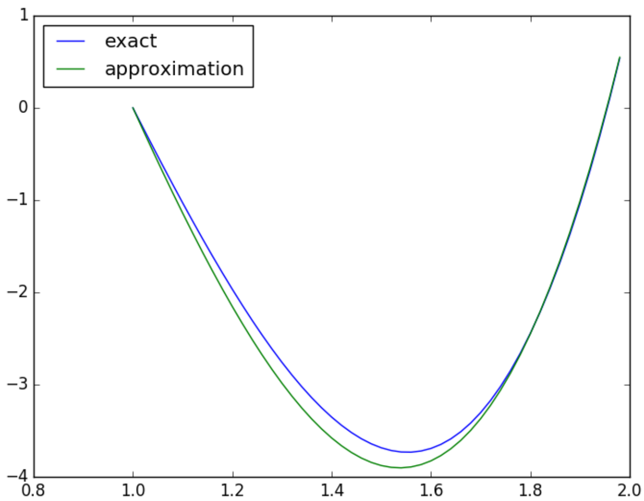
3.1 Задание 1

Коэффициенты вычислялись следующим образом:

$$\sum_{k=1}^{n} C_{ki} a_k - D_i = 0, i = 0, 1, \ldots, n, \qquad (19.7)$$

где $C_{ki} = \int\limits_a^b L(\varphi_k)\varphi_i(x)\,dx$, $D_i = \int\limits_a^b \varphi_i(x)(f(x) - L(\varphi_0))\,dx$.

1. Весовые функции
   (a) $\phi_0 = -1 + x$,
   (b) $\phi_1 = (-1 + x) \cdot (2 - x)$,
   (c) $\phi_2 = (-1 + x) \cdot (2 - x)^2$.
2. Коэффициенты матрицы
   $$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} -0.3411 & -0.1589 \\ -0.1589 & -0.1351 \end{pmatrix},$$
   вектора правой части
   $$D = \begin{pmatrix} D_1 \\ D_2 \end{pmatrix} = \begin{pmatrix} 5.864 \\ 3.303 \end{pmatrix},$$
   и вектора приближенного решения
   $$\hat{a} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} -12.84 \\ -9.345 \end{pmatrix},$$
   системы линейных уравнений.
3. Максимальная глобальная относительная погрешность приближённого $\hat{u}(x)$ и точного $u(x)$ решений — величина
   $$\max_{n=\overline{1,N-1}} \left| \frac{u(x_n) - \hat{u}(x)}{u(x_n)} \right| = 0.2141,$$
4. Графики приближённого $\hat{u}(x)$ и точного $u(x)$ решений на одном рисунке:



Листинг 1: Метод Галёркина

```
import matplotlib.pyplot as plt
import numpy as np
import sympy

s = 15
x = sympy.symbols('x')

def realf(x):
    return (x ** 2 - 1) * (3 * s * (x ** 2 - 4) + 8) / 24.0

def L(y):
    return sympy.diff(y, x, x) - sympy.diff(y, x) / x

n = 2
a = 1.0
b = 2.0
f = s * (x ** 2)
u = (x ** 2 - 1) * (3 * s * (x ** 2 - 4) + 8) / 24.0
phi = [x - 1, (x - 1) * (2 - x), (2 - x) * (x - 1) ** 2]
a11 = sympy.N(sympy.integrate(L(phi[1]) * phi[1], (x, a, b)))
a12 = sympy.N(sympy.integrate(L(phi[1]) * phi[2], (x, a, b)))
a21 = sympy.N(sympy.integrate(L(phi[2]) * phi[1], (x, a, b)))
a22 = sympy.N(sympy.integrate(L(phi[2]) * phi[2], (x, a, b)))
diff = f - L(phi[0])
d1 = sympy.N(sympy.integrate(phi[1] * diff, (x, a, b)))
d2 = sympy.N(sympy.integrate(phi[2] * diff, (x, a, b)))
A = sympy.Matrix([[a11, a12], [a21, a22]])
D = sympy.Matrix([[d1], [d2]])
sol = A.LUsolve(D)
print('a = {0}'.format(sol))
y = phi[0] + sol[0] * phi[1] + sol[1] * phi[2]
max_error = 0
for curx in np.linspace(a + (b - a) / 20.0, b, num=20):
    max_error = max(abs((realf(curx) - y.subs(x, curx)) / realf(curx)), max_er

print('phi_0 = {0}'.format(phi[0]))
print('phi_1 = {0}'.format(phi[1]))
print('phi_2 = {0}'.format(phi[2]))
print('A = {0}'.format(A))
print('D = {0}'.format(D))
print('max error = {0}'.format(max_error))

xx = np.arange(a, b, 0.02)

plt.plot(xx, realf(xx), label="exact")
plt.plot(xx, xx - 1 + sol[0] * (xx - 1) * (2 - xx) + sol[1] * (2 - xx) * (xx -
plt.legend(loc=2)

plt.show()
```

3.2 Задание 2

1. Весовые функции:

   (a) $\phi_0 = -1 + x$,

   (b) $\phi_1 = (-1 + x) \cdot (2 - x)$,

   (c) $\phi_2 = (-1 + x) \cdot (2 - x)^2$,

   (d) $p(x) = 1$,

   (e) $q(x) = \dfrac{1}{x}$.

2. Коэффициенты матрицы

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} 0.3559 & 0.1774 \\ 0.1774 & 0.1393 \end{pmatrix},$$

   вектора правой части

$$D = \begin{pmatrix} D_1 \\ D_2 \end{pmatrix} = \begin{pmatrix} -5.803 \\ -3.280 \end{pmatrix},$$

   и вектора приближенного решения

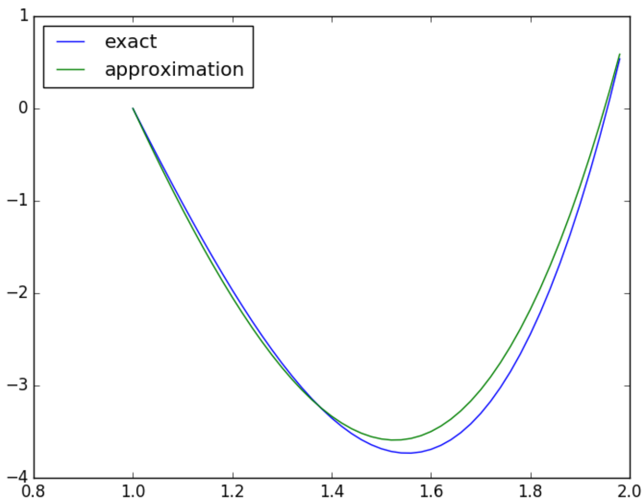
$$\hat{a} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} -12.50 \\ -7.631 \end{pmatrix},$$

   системы линейных уравнений.

3. Максимальная глобальная относительная погрешность приближённого $\hat{u}(x)$ и точного $u(x)$ решений — величина

$$\max_{n=\overline{1,N-1}} \left| \frac{u(x_n) - \hat{u}(x)}{u(x_n)} \right| = 0.2711,$$

4. Графики приближённого $\hat{u}(x)$ и точного $u(x)$ решений на одном рисунке:



Листинг 2: Метод Ритца

```
import sympy
import imp
import scipy.linalg as sla
import matplotlib.pyplot as plt
import numpy as np


def realf(x):
    return (x ** 2 - 1) * (3 * s * (x ** 2 - 4) + 8) / 24.0

def solve(p, q, f, phi, a, b):
    res = sympy.Integer(0)
    x = sympy.symbols('x', real=True)
    u = sympy.symbols('u', cls=sympy.Function)
    n = len(phi) - 1
    aa = np.zeros((n, n))
    for i in range(0, n):
        for j in range(0, n):
            cure = p * phi[i + 1].diff(x) * phi[j + 1].diff(x) + q * phi[i + 1]
            aa[i, j] = sympy.integrate(cure, (x, a, b))
    bb = np.zeros((n))
    for i in range(0, n):
        cure = p * phi[0].diff(x) * phi[i + 1].diff(x) + q * phi[0] * phi[i + 1
        bb[i] = sympy.integrate(cure, (x, a, b))
    return aa, bb

# My variant
s = 15

x = sympy.symbols('x', real=True)
p = 1
q = 1 / x
f = s * x * x
phi = [x - 1, (x - 1) * (2 - x), (x - 1) * (x - 1) * (2 - x)]
a = 1
b = 2
Am, Bm = solve(p, q, f, phi, a, b)
sol = sla.solve(Am, -Bm)
xx = np.linspace(a + (b - a) / 20.0, b, num=20)
approxf = lambda x : (xx - 1 + sol[0] * (xx - 1) * (2 - xx) + sol[1] * (2 - xx)
print('Max␣error:␣{}'.format(np.max(np.abs(realf(xx) - approxf(xx)))))
print('A:␣{}'.format(Am))
print('B:␣{}'.format(-Bm))
print('Solution:␣{}'.format(sol))
xx = np.arange(a, b, 0.02)
plt.plot(xx, (xx ** 2 - 1) * (3 * s * (xx ** 2 - 4) + 8) / 24.0, label="exact")
plt.plot(xx, xx - 1 + sol[0] * (xx - 1) * (2 - xx) + sol[1] * (2 - xx) * (xx -
plt.legend(loc=2)
plt.show()
```

3.3 Задание 3

Использованые формулы:

$$y_\alpha^{'} = \frac{y_\alpha}{x} + 15x^2,$$

y_α - приближение $u^{'}$, которое было найдено с начальным условием (1) $y_\alpha(1) = \alpha$.

$$u_\alpha^{'} = y_\alpha$$

$$\alpha = \alpha - \frac{-1 + u_\alpha(2)}{u_\alpha^{'}(2)}$$

1. Число итераций метода Ньютона = 72.

2. Абсолютная погрешность второго краевого условия = 0.04387.

3. Максимальная глобальная относительная погрешность

$$\max_{n=\overline{1,N-1}} \left| \frac{u(x_n) - \hat{u}(x)}{u(x_n)} \right| = 0.04456.$$

Листинг 3: Метод Стрельбы

```
import math
import pylab
import numpy as np
import matplotlib.pyplot as plt

a, b = 1.0, 2.0
A, B = 0.0, 1.0
n = 20
h = (b - a) / n
s = 15
u_approx_x = np.zeros(1)
u_approx_y = np.zeros(1)
du_approx_x = np.zeros(1)
du_approx_y = np.zeros(1)

x = np.arange(a, b + 0.0001, h)

def realf(x):
    return (x ** 2 - 1) * (3 * s * (x ** 2 - 4) + 8) / 24.0

def find_nearest_id(array, value):
    return np.abs(array - value).argmin()

def du(x, y):
    return du_approx_y[find_nearest_id(du_approx_x, x)]

def ddu(x, y):
    return  y / x + s * x * x

def gety(x, y, h, f):
    k1 = h * f(x, y)
```

```
    k2 = h * f(x + h * 0.5, y + k1 * 0.5)
    k3 = h * f(x + h * 0.5, y + k2 * 0.5)
    k4 = h * f(x + h, y + k3)
    return y + (k1 + 2.0 * k2 + 2.0 * k3 + k4) / 6.0

def runge_kutta1(h):
    global u_approx_x, u_approx_y
    curx = a
    while curx < b and h > 0.0:
        y1 = gety(u_approx_x[-1], u_approx_y[-1], h, ddu)
        curx += h
        u_approx_x = np.append(u_approx_x, curx + h)
        u_approx_y = np.append(u_approx_y, y1)


def runge_kutta2(h):
    global du_approx_x, du_approx_y
    curx = a
    while curx < b and h > 0.0:
        y1 = gety(du_approx_x[-1], du_approx_y[-1], h, du)
        curx += h
        du_approx_x = np.append(du_approx_x, curx + h)
        du_approx_y = np.append(du_approx_y, y1)

h = 0.25
alpha = 1
newton_iterations = 0
u_approx_x = np.array([a])
u_approx_y = np.array([A])
du_approx_x = np.array([a])
du_approx_y = np.array([alpha])

runge_kutta1(h * 0.5)
runge_kutta2(h)
while np.abs(u_approx_y[-1] - 1.0) > 0.05:
    print(np.abs(u_approx_y[-1] - 1.0), alpha)
    newton_iterations += 1
    alpha = alpha - (u_approx_y[-1] - 1.0) / du_approx_y[-1]
    u_approx_x = np.array([a])
    u_approx_y = np.array([A])
    du_approx_x = np.array([a])
    du_approx_y = np.array([alpha])
    runge_kutta1(h * 0.5)
    runge_kutta2(h)
print(newton_iterations)
print('Error: {}'.format(np.max(np.abs(realf(x) - yy))))
```