Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу

Лабораторна робота №3

З курсу «Чисельні методи»

З теми «Методи розв’язання нелінійних систем»

Варіант №5

Виконав студент 2 курсу групи КА-01
Вагін Олександр Вікторович
Перевірила старший викладач
Хоменко Ольга Володимирівна

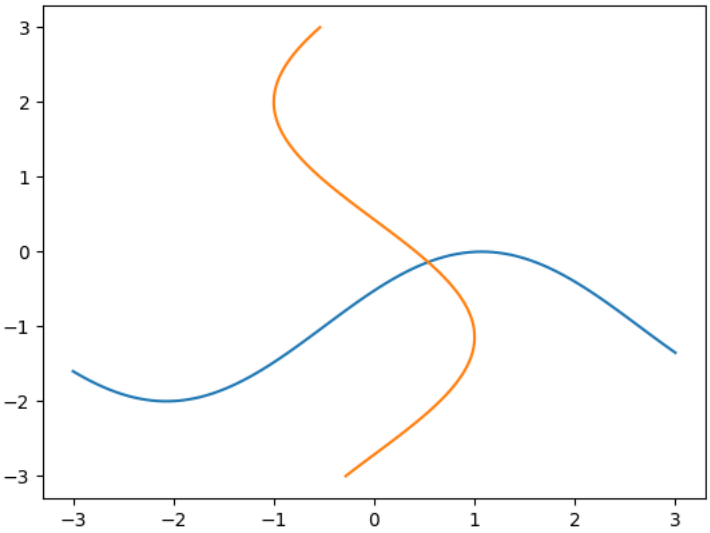
Київ-2022

Завдання №1

Розв’язати систему рівнянь методом простих ітерацій з точністю 0,00001:

$$\begin{cases} \sin(x+0{,}5) - y = 1 \\ \cos(y-2) + x = 0 \end{cases}$$

Визначимо початкове наближення, побудувавши графіки кривих системи:



Отже, візьмемо $x^{(0)} = (0{,}5; -0{,}1)$

Перепишемо систему у вигляді:

$y = \sin(x+0{,}5) - 1 = \varphi_2$

$x = -\cos(y-2) = \varphi_1$

Візьмемо частинні похідні:

$$\frac{\partial \varphi_1}{\partial x} = 0 \quad \frac{\partial \varphi_1}{\partial y} = \sin(y-2)$$

$$\frac{\partial \varphi_2}{\partial x} = \cos(x+0{,}5) \quad \frac{\partial \varphi_2}{\partial y} = 0$$

Перевіримо виконання умов збіжності. Будемо розглядати окіл точки $x^{(0)}$:

$G = \{|x - 0{,}5| \le 0{,}1; |y + 0{,}1| \le 0{,}1\}$

$$\max_{\vec{x}\in G}\left(\max_{j}\sum_{i=1}^{n}\left|\frac{\partial\varphi_i}{\partial x_j}\right|\right)\le q\le 1$$

$$\left|\frac{\partial\varphi_1}{\partial x}\right|=0\qquad\left|\frac{\partial\varphi_1}{\partial y}\right|=|\sin(y-2)|$$

$$\left|\frac{\partial\varphi_2}{\partial x}\right|=|\cos(x+0{,}5)|\qquad\left|\frac{\partial\varphi_1}{\partial y}\right|=0$$

$$\left|\frac{\partial\varphi_1}{\partial x}\right|+\left|\frac{\partial\varphi_2}{\partial x}\right|=|\cos(x+0{,}5)|\le\cos(0{,}4)<1$$

$$\left|\frac{\partial\varphi_1}{\partial y}\right|+\left|\frac{\partial\varphi_1}{\partial y}\right|=|\sin(y-2)|\le|\sin(-2)|<1$$

Умови збіжності виконуються. Якщо послідовні наближення не будуть виходити з області G, то ітераційний процес збіжний.

Будемо знаходити послідовні наближення за формулами:

$$x^{(k+1)}=-\cos\left(y^{(k)}-2\right)$$

$$y^{(k+1)}=\sin\left(x^{(k)}+0{,}5\right)-1$$

№ iteration	x	y	Δ
0	0.50000	-0.10000	0.000000
1	0.50485	-0.15853	0.058529
2	0.55448	-0.15592	0.049629
3	0.55230	-0.13036	0.025562
4	0.53082	-0.13143	0.021488
5	0.53173	-0.14228	0.010848
6	0.54088	-0.14181	0.009156
7	0.54049	-0.13715	0.004664
8	0.53656	-0.13735	0.003930
9	0.53673	-0.13934	0.001994
10	0.53841	-0.13926	0.001682
11	0.53834	-0.13840	0.000855
12	0.53762	-0.13844	0.000721
13	0.53765	-0.13881	0.000366
14	0.53795	-0.13879	0.000309
15	0.53794	-0.13863	0.000157

Загалом вийшло 23 ітерації та розв’язок: $(0{,}57386; -0{,}13868)$

Виконаємо перевірку отриманого розв’язку, обчисливши $F(\overrightarrow{x^*})$:

$$F(\overrightarrow{x^*}) = \begin{pmatrix} \cos(y-2)+x \\ \sin(x+0{,}5)-y-1 \end{pmatrix}_{(0{,}57386;-0{,}13868)} = \begin{pmatrix} 3{,}6 \cdot 10^{-5} \\ 1{,}7 \cdot 10^{-5} \end{pmatrix}$$

Маємо доволі близькі до нуля значення.

Якщо випадковим чином задати початкові наближення на відрізку $[-50; 50]$, то отримаємо результат в середньому за 29 ітерацій

```
Random initial approximation
Attempt №1: initial approx - (-24.646262548914976, 49.70254038613392) root is (0.5378538535339918, -0.13868005751445744), iterations - 27
Attempt №2: initial approx - (-22.596011594766253, 30.809967728562) root is (0.5378483411985528, -0.138686623346764), iterations - 29
Attempt №3: initial approx - (6.062271009915932, -24.307789317165874) root is (0.5378483501914804, -0.13868684756798855), iterations - 30
Attempt №4: initial approx - (-48.01011295577797, 1.3564271346926589) root is (0.5378482788284269, -0.13868669562375602), iterations - 30
Attempt №5: initial approx - (25.065064596474656, 34.376319795270575) root is (0.5378482514775718, -0.13868661027612705), iterations - 30
```

Якщо скористаємося для пошуку розв’язку пакетом scipy, то отримаємо розв’язок: $(0{,}537853, -0{,}13868462)$

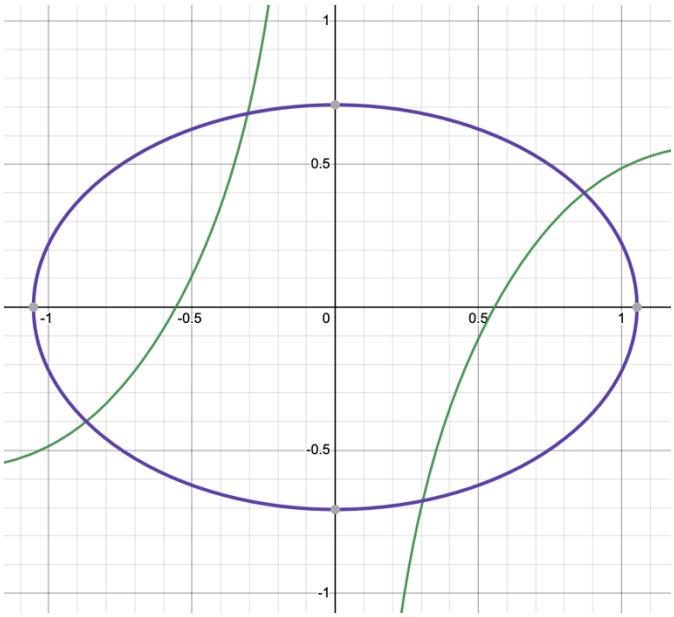
```
Via SciPy
Root is [ 0.537853   -0.13868462]
```

Завдання №2

Розв’язати методом Ньютона систему:

$$\begin{cases} \tan(xy+0{,}3) = x^2 \\ 0{,}9x^2 + 2y^2 = 1 \end{cases}$$

Визначимо початкове наближення, побудувавши графіки кривих системи:



З графіку видно, що система матиме 4 розв’язки. Візьмемо такі початкові наближення:

$$x_1^{(0)} = (-0{,}9; -0{,}4), x_2^{(0)} = (-0{,}3; 0{,}7), x_3^{(0)} = (0{,}3; -0{,}7), x_4^{(0)} = (0{,}9; 0{,}4)$$

Знайдемо матрицю Якобі:

$$f_1(x, y) = \tan(xy + 0{,}3) - x^2, \qquad f_2(x, y) = 0{,}9x^2 + 2y^2 - 1$$

$$W(x) = \begin{pmatrix} \dfrac{y}{\cos^2(xy + 0{,}3)} - 2x & \dfrac{x}{\cos^2(xy + 0{,}3)} \\ 1{,}8x & 4y \end{pmatrix}$$

За методом Ньютона отримали наближені розв’язки:

Root №1

№ iteration	x	y	Δ
0	-0.90000	-0.40000	0.000000
1	-0.92594	-0.44436	0.044356
2	-0.92948	-0.44382	0.003535
3	-0.92993	-0.44413	0.000449
4	-0.92999	-0.44416	0.000058
5	-0.93000	-0.44417	0.000008

Solution of system is: (-0.93, -0.44417)

Root №2

№ iteration	x	y	Δ
0	-0.30000	0.70000	0.000000
1	-0.30392	0.68389	0.016113
2	-0.30396	0.68386	0.000045
3	-0.30396	0.68386	0.000002

Solution of system is: (-0.30396, 0.68386)

Root №3

№ iteration	x	y	Δ
0	0.30000	-0.70000	0.000000
1	0.30392	-0.68389	0.016113
2	0.30396	-0.68386	0.000045
3	0.30396	-0.68386	0.000002

Solution of system is: (0.30396, -0.68386)

Root №4

№ iteration	x	y	Δ
0	0.90000	0.40000	0.000000
1	0.92594	0.44436	0.044356
2	0.92948	0.44382	0.003535
3	0.92993	0.44413	0.000449
4	0.92999	0.44416	0.000058
5	0.93000	0.44417	0.000008

Solution of system is: (0.93, 0.44417)

В середньому для пошуку коренів було потрібно 4 ітерації та отримали такі наближені розв’язки:

$x_1^* = (-0{,}93; -0{,}44417), \qquad x_2^* = (-0{,}30396; 0{,}68386)$

$x_3^* = (0{,}30396; -0{,}68386), \qquad x_4^* = (0{,}93; 0{,}44417)$

Виконаємо перевірку отриманого розв’язку, обчисливши $F(\overrightarrow{x^*})$:

$$F(\overrightarrow{x^*}) = \begin{pmatrix} \tan(xy + 0{,}3) - x^2 \\ 0{,}9x^2 + 2y^2 - 1 \end{pmatrix}_{\overrightarrow{x_j^*}}$$

$$x_1^*: F(\overrightarrow{x^*}) = \begin{pmatrix} 1 \cdot 10^{-10} \\ -2 \cdot 10^{-6} \end{pmatrix}, \qquad x_2^*: F(\overrightarrow{x^*}) = \begin{pmatrix} 8 \cdot 10^{-13} \\ 6 \cdot 10^{-8} \end{pmatrix}$$

$$x_3^*: F(\overrightarrow{x^*}) = \begin{pmatrix} 8 \cdot 10^{-13} \\ 6 \cdot 10^{-8} \end{pmatrix}, \qquad x_4^*: F(\overrightarrow{x^*}) = \begin{pmatrix} 1 \cdot 10^{-10} \\ -2 \cdot 10^{-6} \end{pmatrix}$$

Оскільки отримані значення доволі близькі до нуля, то перевірка виконана.

Якщо випадковим чином задати початкові наближення на відрізку $[-50; 50]$, то отримаємо результат в середньому за 15 ітерацій з можливими випадками на більше ітерацій.

```
Random initial approximation
Attempt №1: initial approx - (-10.880470186717638, 13.459836621038257) root is (0.30396332890015265, -0.6838583662328589), iterations - 14
Attempt №2: initial approx - (-2.256624176387021, 5.69094670861535) root is (0.3039633575436964, -0.6838582449699683), iterations - 11
Attempt №3: initial approx - (16.920089320776682, -25.15883876813113) root is (0.30396333821791927, -0.6838583268027782), iterations - 17
Attempt №4: initial approx - (37.76105970203545, 1.79092714674087) root is (-0.9299966413048414, -0.44417080736432546), iterations - 40
Attempt №5: initial approx - (-13.799315286842507, -36.91500818031665) root is (0.9299967355205883, 0.4441708701667473), iterations - 19
```

Якщо скористаємося для пошуку розв’язку пакетом scipy, то отримаємо розв’язки:

$x_1^* = (-0{,}92999622; -0{,}44417053), \qquad x_2^* = (-0{,}30396334; 0{,}68385832)$

$x_3^* = (0{,}30396334; -0{,}68385832), \qquad x_4^* = (0{,}92999622; 0{,}44417053)$

```
Via SciPy
Root №1
[-0.92999622 -0.44417053]
Root №2
[-0.30396334  0.68385832]
Root №3
[ 0.30396334 -0.68385832]
Root №4
[0.92999622 0.44417053]
```

Код програм:

```
import matplotlib.pyplot as plt
import numpy as np
from functools import partial
import random
from scipy.optimize import fsolve


# First task initial data
def y1(x):
    """Returns first equation of system"""
    return np.sin(x + 0.5) - 1


def x1(y):
    """Returns second equation of system"""
    return -np.cos(y - 2)


def print_intersection(f1, f2):
    """Builds graphic of two equations on the one plot"""
    plt.subplots()
    ls = np.linspace(-3, 3, 100)
    plt.plot(ls, f2(ls))
    plt.plot(f1(ls), ls)
    plt.show()


r5 = partial(round, ndigits=5)


def print_log(log):
    """Takes log and print it"""
    print("-" * 55)
    print('|' + "№ iteration".center(15) + '|'
          + 'x'.center(11) + '|'
          + 'y'.center(12) + '|'
          + 'Δ'.center(12) + '|')
    print("-" * 55)
    for i in range(len(log)):
        print('|' + str(i).center(15) + '|'
              + '{:.5f}'.format(log[i][0][0]).center(11) + '|'
              + '{:.5f}'.format(log[i][0][1]).center(12) + '|'
              + '{:.6f}'.format(log[i][1]).center(12) + '|')
        print("-" * 55)


def norm_max(vector):
    """Takes vector, returns its Chebyshev's norm"""
    return max(map(abs, vector))


def vector_difference(a, b):
    """Takes two vectors, returns their difference"""
    return tuple(t - s for t, s in zip(a, b))


def simple_iterations(x_0, eps, equations):
    """
    Solves system of nonlinear equations
    Takes:
        x_0 - initial approximation
```

```
        eps - precision
        equations - equations of system
    Returns:
        x_1 - vector of approximate solutions
        log - history of iterations
    """
    equations = equations[::-1]
    log = [(x_0, 0)]
    while True:
        x_1 = tuple(equations[i](x_0[i]) for i in range(len(x_0)))[::-1]
        delta = norm_max(vector_difference(x_1, x_0))
        log.append((x_1, delta))
        if delta <= eps:
            return x_1, log
        x_0 = x_1


def second_system(vector):
    """Returns equations of system from second task as tuple"""
    return np.tan(vector[0] * vector[1] + 0.3) - vector[0] ** 2, 0.7 *
(vector[0] ** 2) + 2 * (vector[1] ** 2) - 1


def jacobian(vector):
    """Returns jacoby matrix of the second system in the point"""
    x, y = vector
    m00 = y / np.cos(x * y + 0.3) ** 2 - 2 * x
    m01 = x / np.cos(x * y + 0.3) ** 2
    m10 = 1.8 * x
    m11 = 4 * y
    return (m00, m01), (m10, m11)


def newton_search(x_0, eps, jacoby, equations):
    log = [(x_0, 0)]
    while True:
        difference = tuple(np.linalg.solve(jacoby(x_0), tuple(-x for x in
equations(x_0))))
        x_1 = tuple(x + y for x, y in zip(difference, x_0))
        delta = norm_max(vector_difference(x_1, x_0))
        log.append((x_1, delta))
        if delta <= eps:
            return x_1, log
        x_0 = x_1


# First task
print("First task\n")

# Looking graphically for first initial approximation
print_intersection(x1, y1)

# Calculating solution
solution, history = simple_iterations((0.5, -0.1), 0.00001, (x1, y1))
print_log(history)
print("Solution of system is:", tuple(map(r5, solution)))

# Calculating solution with random initial approximation
print("\nRandom initial approximation")
for j in range(5):
    approx = random.uniform(-50, 50), random.uniform(-50, 50)
    solution, history = simple_iterations(approx, 0.00001, (x1, y1))
    print(f"Attempt №{j + 1}: initial approx - {approx} root is {solution},
iterations - {len(history)}")
```

```
# Calculating solution via SciPy
print("\nVia SciPy")
print("Root is", fsolve((lambda x: (np.cos(x[1] - 2) + x[0], np.sin(x[0] +
0.5) - x[1] - 1)), (0.5, -0.1)))

# Second task

print("\nSecond task\n")
initial_approximations = ((-0.9, -0.4), (-0.3, 0.7), (0.3, -0.7), (0.9, 0.4))

# Calculating solution
for j in range(len(initial_approximations)):
    print(f"Root №{j + 1}")
    solution, history = newton_search(initial_approximations[j], 0.00001,
jacobian, second_system)
    print_log(history)
    print("Solution of system is:", tuple(map(r5, solution)))
    print(second_system(solution))

# Calculating solution with random initial approximation
print("\nRandom initial approximation")
for j in range(5):
    approx = random.uniform(-50, 50), random.uniform(-50, 50)
    solution, history = newton_search(approx, 0.00001, jacobian,
second_system)
    print(f"Attempt №{j + 1}: initial approx - {approx} root is {solution},
iterations - {len(history)}")

# Calculating solution via SciPy
print("\nVia SciPy")
for j in range(len(initial_approximations)):
    print(f"Root №{j + 1}")
    print(fsolve(second_system, initial_approximations[j]))
```

Висновок:

В ході виконання роботи реалізовано методи простих ітерацій та Ньютона. З отриманих результатів можемо зробити висновок, що метод Ньютона працює набагато швидше за метод простих ітерацій, але він також є доволі чутливим до випадкових початкових наближень. Загалом метод простих ітерацій потребує до 30 ітерацій, а метод Ньютона, за умови «осмисленних» початкових наближень працює за п’ять.