НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС
"ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ"
НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ
"КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО"
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Лабораторна робота № 3
з предмету "Чисельні методи"
з теми «Методи розв'язання нелінійних систем»
Варіант № 13

Виконала:
студентка групи
КА-02
Шапошнікова Софія

Перевірила:
Хоменко О.В.

Київ 2022

Завдання 1

1. Розв'язати систему 1 методом простих ітерацій. Для цього:
   - визначити початкове наближення, побудувавши графіки кривих системи;
   - перевірити достатні умови збіжності з детальним поясненням (задати область, в якій перевірити виконання умов збіжності, можна робити фото написаного і вставляти в звіт);
   - реалізувати метод простих ітерацій. Розв'язати систему з точністю $\epsilon = 10^{-5}$
   - програмний код надіслати в класрум в окремому файлі та вставити текст програми у звіт.

2. Результати роботи програми оформити у звіті у вигляді таблиці. Якщо ітерацій більше 15, в таблицю записати лише перші 15.

3. Виконати перевірку, обчисливши $F(x_*)$

4. Задати декілька інших початкових наближень (які не близькі до розв'язку) та з'ясувати як змінюється при цьому ітераційний процес, написати про це у висновку.

5. Знайти розв'язок системи за допомогою fsolve бібліотеки scipy.optimize

Варіант 13

$$\begin{cases} \sin y + 2x = 2 \\ \cos(x-1) + y = 0,7 \end{cases}$$

Теоретичні відомості

Алгоритм розв'язання нелінійних систем методом простих ітерацій

1) Задати початкове наближення $\mathbf{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)})^T$ та мале додатнє ε (точність). Покласти $k = 0$.

2) Обчислити $\mathbf{x}^{(k+1)}$:

$$\begin{aligned} x_1^{(k+1)} &= \varphi_1(x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)}), \\ x_2^{(k+1)} &= \varphi_2(x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)}), \\ &\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ x_n^{(k+1)} &= \varphi_n(x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)}). \end{aligned} \tag{4.3}$$

3) Якщо $\Delta^{(k+1)} = max_i|x_i^{(k+1)} - x_i^{(k)}| \le \varepsilon$, зупинитись і покласти $\mathbf{x}_* := \mathbf{x}^{(k+1)}$. Якщо $\Delta^{(k+1)} > \varepsilon$, то покласти $k = k+1$ і перейти до пункту 2.

1 Допрограмовий етап

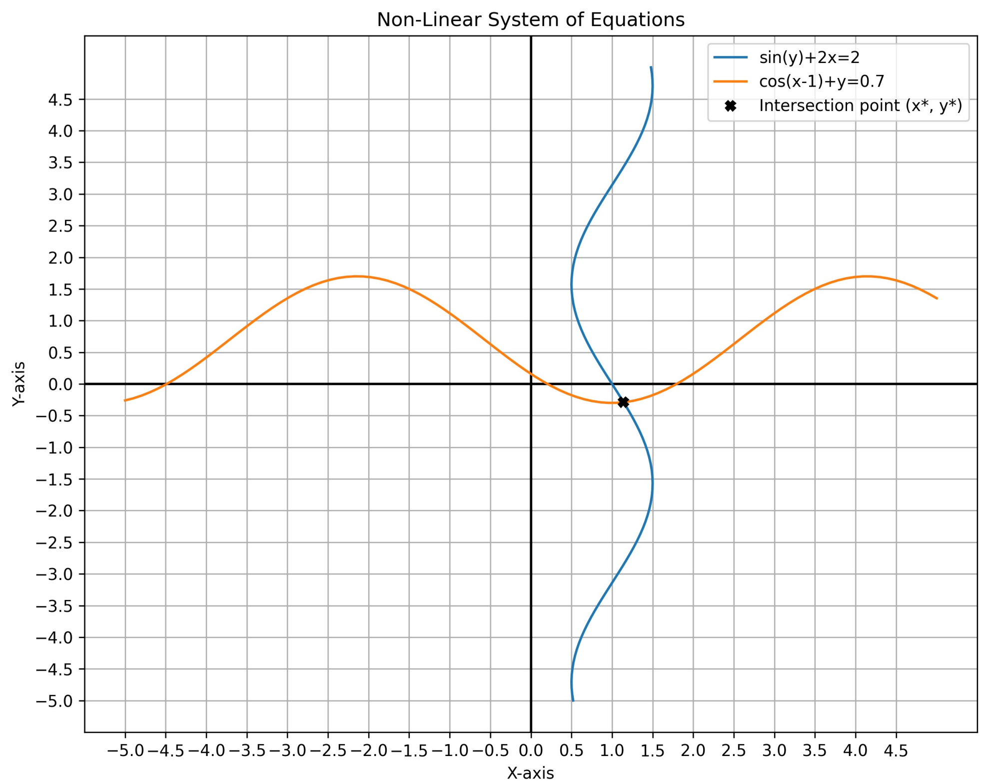
1.1 Визначимо початкове наближення, побудувавши графіки кривих системи

Запишемо систему у вигляді:

$$\begin{cases} x = 1 - \frac{\sin(y)}{2} = \phi_1(x) \\ y = 0,7 - \cos(x-1) = \phi_2(x) \end{cases}$$

Для вибору початкового наближення знайдемо координати точок перетину кривих, Що відповідають 1 і 2-му рівнянням системи:

$$x^{(0)} = (1.143; -0.29)^T$$


Non-Linear System of Equations
sin(y)+2x=2
cos(x-1)+y=0.7
Intersection point (x*, y*)
X-axis
Y-axis
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
−3.5
−4.0
−4.5
−5.0
−5.0 −4.5 −4.0 −3.5 −3.0 −2.5 −2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5

1.2 Перевіримо достатні умови збіжності

Теорема 4.1 (достатні умови збіжності методу простих ітерацій для нелінійних систем)

Нехай функції $\varphi_i(x)$ та $\varphi_i'(x)$, $i = 1, ..., n$ неперервні в області G, причому виконується умова

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

або умова

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

де q – деяка стала.

Якщо послідовні наближення $\mathbf{x}^{(k+1)} = \mathbf{\Phi}(\mathbf{x}^{(k)})$, $k = 0, 1, 2, ...$ не виходять з області G, то процес послідовних наближень збіжний $\mathbf{x}_* = lim_{k\to\infty}\mathbf{x}^{(k)}$ і вектор $\mathbf{x}_*$ в області G є єдиним розв'язком системи

Знайдемо частинні похідні:

$$\frac{\partial\varphi_1}{\partial x} = 0, \qquad \frac{\partial\varphi_1}{\partial y} = -\frac{\cos(y)}{2}$$
$$\frac{\partial\varphi_2}{\partial x} = \sin(x-1), \quad \frac{\partial\varphi_2}{\partial y} = 0$$

Задамо область, в якій перевіримо виконання умов збіжності:

$$x^{(0)} : G = \{|x_1 - 1,143| \leq 0,1; |x_2 + 0,29| \leq 0,1\}$$
$$\max_{x\in G} \max_{j} \sum_{i=1}^{n} \left|\frac{\partial\varphi_i(x)}{\partial x_j}\right| \leq q < 1$$

$$\left|\frac{\partial\varphi_1}{\partial x}\right| = 0, \qquad \left|\frac{\partial\varphi_1}{\partial y}\right| \leq \frac{\cos(-0,19)}{2} \approx 0,491 < 0,5$$
$$\left|\frac{\partial\varphi_2}{\partial x}\right| \leq \sin(1,043-1) \approx 0.4299 < 0,43, \quad \left|\frac{\partial\varphi_2}{\partial y}\right| = 0$$

$$\left|\frac{\partial\varphi_1}{\partial x}\right| + \left|\frac{\partial\varphi_2}{\partial x}\right| < 0 + 0,43 < 1$$
$$\left|\frac{\partial\varphi_1}{\partial y}\right| + \left|\frac{\partial\varphi_2}{\partial y}\right| < 0.5 + 0 < 1$$

Бачимо, що **умови збіжності виконуються**. Якщо послідовні наближення не будуть виходити з області G, то ітераційний процес збіжний

1.3 Реалізація методу простих ітерацій

Будемо знаходити послідовні наближення за формулами:

$$\begin{cases} x^{(k+1)} = 1 - \frac{\sin(y^{(k)})}{2} \\ y^{(k+1)} = 0,7 - \cos(x^{(k)} - 1) \end{cases}, k = 0, 1, \dots$$

$$\begin{cases} x^{(1)} = 1 - \frac{\sin(y^{(0)})}{2} = 1 - \frac{\sin(-0.29)}{2} \approx 1.14298 \\ y^{(1)} = 0,7 - \cos(x^{(0)} - 1) = 0,7 - \cos(1.143 - 1) \approx -0.289793 \end{cases}$$

Знайдемо $\Delta^{(1)}$:

$$|x^{(1)} - x^{(0)}| = |1,14298 - 1,143| = 0,00002$$
$$|y^{(1)} - y^{(0)}| = |-0,289793 - (-0,29)| = 0,000207$$
$$\Delta^{(1)} = \max\{0,00002; 0,000207\} = 0,000207 > 0,00001 = \epsilon$$

Отже, **продовжуємо ітераційний процес** програмними методами

In [5]:
```
import numpy as np
import copy
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
import handcalcs.render
from scipy.optimize import fsolve
```

Метод простої ітерації

Розв'яжемо систему нелінійних рівнянь

$$\begin{cases} \sin y + 2x = 2 \\ \cos(x - 1) + y = 0,7 \end{cases}$$

In [6]:
```
def F(variables) :
    (x,y)= variables
    f1 = 2 - np.sin(y) - 2*x
    f2 = 0.7 - np.cos(x-1) - y
    return [f1,f2]
```

In [7]:
```
result = fsolve(F, (1, 1))
print(result)
```

```
[ 1.14288476 -0.28980933]
```

In [8]:
```
print (F(result))
```

```
[-2.1316282072803006e-14, -2.3425705819590803e-14]
```

In [9]:
```
x1 = lambda y1: -(np.sin(y1))/2 +1
y2 = lambda x2: 0.7 - (np.cos(x2-1))
```

Метод простих ітерацій

```
In [11]: eps = 0.00001
         appr = np.array([1.143, -0.29])
```

```
In [12]: x1 = lambda y1: -(np.sin(y1))/2 +1
         y2 = lambda x2: 0.7 - (np.cos(x2-1))
```

```
In [29]: def simple_iter_solve(x1, y2, appr, eps):
             find_appr = lambda a0, a1: max(abs(a1[0]-a0[0]), abs(a1[1]-a0[1]))
             res, appr_hist = [], []
             while True:
                 new_appr = np.array([round(x1(appr[1]), 6), round(y2(appr[0]),6)])
                 res.append(new_appr)
                 appr_hist.append(find_appr(appr, new_appr))
                 if find_appr(appr, new_appr) < eps:
                     break
                 appr = copy.deepcopy(new_appr)
             return new_appr, np.array(res), np.array(appr_hist)
```

```
In [30]: solution, iter_sol, approximations = simple_iter_solve(x1, y2, appr, eps)
         solution, iter_sol, approximations
```

```
Out[30]: (array([ 1.142885, -0.28981 ]),
          array([[ 1.142976, -0.289793],
                 [ 1.142877, -0.289796],
                 [ 1.142878, -0.28981 ],
                 [ 1.142885, -0.28981 ]]),
          array([2.07e-04, 9.90e-05, 1.40e-05, 7.00e-06]))
```

```
In [31]: for i in approximations:
             print("%01.6f" % round(i,6))
```

```
0.000207
0.000099
0.000014
0.000007
```

2 Результати

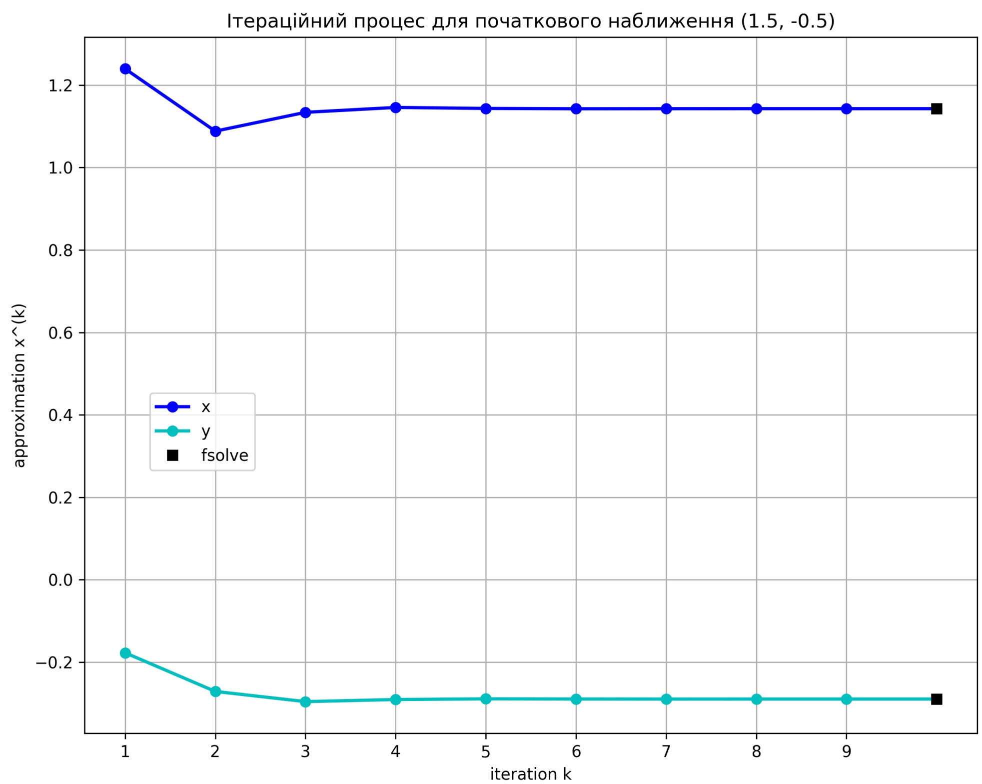
2.1 Табличне представлення:

№ ітерації	x	y	$\Delta^{(k)}$
0	1.143	-0.29	$-$
1	1.142976	-0.289793	0.000207
2	1.142877	-0.289796	0.000099
3	1.142878	-0.28981	0.000014
4	1.142885	-0.28981	0.000007

Відповідь: $x = (1.142885, -0.28981)^T$

2.2 Графічне представлення:

Зобразимо ітераційний процес на графіку. Для наочості речузьтатів, візьмемо інше початкове наближення:



3 Перевірка

$$\begin{cases} \sin y + 2x = 2 \\ \cos(x-1) + y = 0,7 \end{cases}$$

```
In [58]: x_ch = lambda y_c: np.arccos(0.7-y_c) + 1
         y_ch = lambda x_c: np.arcsin(2-2*x_c)

In [59]: solution

Out[59]: array([ 1.142885, -0.28981 ])

In [60]: x_ch(solution[1])

Out[60]: 1.1428800419755714

In [61]: y_ch(solution[0])

Out[61]: -0.28980983970736446
```

Підставивши розв'язки в систему, отримуємо відповідні результати(округлені до 5 знаків після коми):

$$\begin{cases} x = \arccos(0.7 - y) + 1 = 1.14288 \\ y = \arcsin(2 - 2x) = -0.28981 \end{cases}$$

4 Інші початкові наближення

```
In [40]: appr

Out[40]: array([ 1.143, -0.29 ])

In [48]: appr_list = [[2.5, -2.5], [0.01, -50], [-4,-0.3]]

In [51]: for pair in appr_list:
             print('Numb of iter: ', simple_iter_solve(x1, y2, pair, eps)[2].shape, end=' ')
             print(", result: ", simple_iter_solve(x1, y2, pair, eps)[0])

         Numb of iter:  (11,) , result:  [ 1.142885 -0.289809]
         Numb of iter:  (10,) , result:  [ 1.142884 -0.289809]
         Numb of iter:  (10,) , result:  [ 1.142886 -0.289809]
```

Бачимо, що при заданні інших початкових наближень(не близьких до розв'язку),відповідь не змінюється, проте змінюється кількість ітерацій змінюється

5 Розв'язок з допомогою функції fsolve

Метод простих ітерацій

Розв'яжемо систему нелінійних рівнянь

$$\begin{cases} \sin y + 2x = 2 \\ \cos(x-1) + y = 0,7 \end{cases}$$

In [6]:
```
def F(variables) :
    (x,y)= variables
    f1 = 2 - np.sin(y) - 2*x
    f2 = 0.7 - np.cos(x-1) - y
    return [f1,f2]
```

In [7]:
```
result = fsolve(F, (1, 1))
print(result)
```

```
[ 1.14288476 -0.28980933]
```

Маємо змогу порівняти результати:
Відповідь, знайдена з допомогою алгоритму методу простих ітерацій:
$x = (1.14288, -0.28981)^T$
Відповідь, знайдена з допомогою функції fsolve (округлена до 5 знаків після коми):
$x = (1.14288, -0.28981)^T$
Можемо помітити, що відповіді із заданою точністю співпадають

Завдання 2

1. Розв'язати систему 2 методом Ньютона (або спрощеним методом Ньютона). Для цього:
   - визначити початкове наближення, побудувавши графіки кривих системи;
   - реалізувати метод Ньютона (або спрощений метод Ньютона). За потреби можна використовувати функції linalg.solve та ін. Розв'язати систему з точністю $\epsilon = 10^{-5}$
   - програмний код надіслати в класрум в окремому файлі та вставити текст програми у звіт.

2. Результати роботи програми оформити у звіті у вигляді таблиці. Якщо ітерацій більше 15, в таблицю записати лише перші 15.

3. Виконати перевірку, обчисливши $F(x_*)$

4. Задати декілька інших початкових наближень (які не близькі до розв'язку) та з'ясувати як змінюється при цьому ітераційний процес, написати про це у висновку.

5. Знайти розв'язок системи за допомогою fsolve бібліотеки scipy.optimize

Варіант 13

$$\begin{cases} \operatorname{tg}(xy + 0,4) = x^2 \\ 0,8x^2 + 2y^2 = 1 \end{cases}$$

Теоретичні відомості

Алгоритм розв'язання нелінійних систем методом Ньютона

1) Задати початкове наближення $\mathbf{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)})^T$ та мале додатнє ε (точність). Покласти $k = 0$.

2) Розв'язати систему лінійних алгебраїчних рівнянь відносно поправки $\mathbf{\Delta x}^{(k)}$:

$$\mathbf{W}(\mathbf{x}^{(k)})\mathbf{\Delta x}^{(k)} = -\mathbf{F}(\mathbf{x}^{(k)}). \tag{4.6}$$

3) Обчислити наступне наближення: $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{\Delta x}^{(k)}$.

4) Якщо $\Delta^{(k+1)} = max_i|x_i^{(k+1)} - x_i^{(k)}| \le \varepsilon$, зупинитись і покласти $\mathbf{x}_* := \mathbf{x}^{(k+1)}$. Якщо $\Delta^{(k+1)} > \varepsilon$, то покласти $k = k + 1$ і перейти до пункту 2.

Теорема 4.2 (достатні умови збіжності метода Ньютона). Нехай функція $\mathbf{F}(\mathbf{x})$ неперервно диференційовна на відкритій опуклій множині $G \in R^N$. Нехай існують $r, \beta > 0$ такі, що $N(\mathbf{x}_*, r) \subset G$, та існує $\|\mathbf{W}^{-1}(\mathbf{x}_*)\| \le \beta$ і $\mathbf{W}(\mathbf{x}) \in Lip_\gamma(N(\mathbf{x}_*, r))$. Тоді існує $\varepsilon > 0$, що для всіх $\mathbf{x}^{(0)} \in N(x_*, \varepsilon)$ послідовність $\mathbf{x}_{(1)}$, $\mathbf{x}_{(2)}$,... що задається (4.6), збіжна до x_* і задовольняє нерівність

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}_*\| \le \beta\gamma\|\mathbf{x}^{(k)} - \mathbf{x}_*\|^2, \;\; k = 0, 1, 2, ...$$

Тут використані такі позначення:

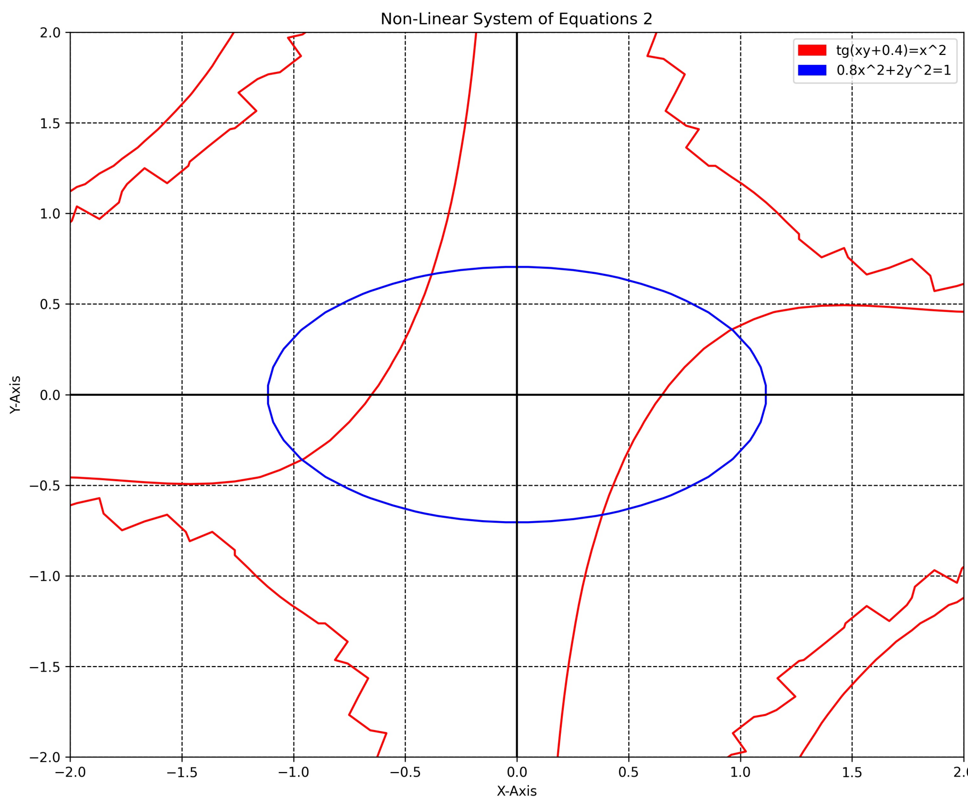
$N(\mathbf{x}, r)$ – відкритий окіл радіуса r з центром в точці x;

запис $Lip_\gamma(N(\mathbf{x}_*, r))$ означає, що $\mathbf{W}(\mathbf{x})$ неперервна за Ліпшицем, де γ – константа Ліпшиця, тобто $\|\mathbf{W}(\mathbf{y}) - \mathbf{W}(\mathbf{x})\| \le \gamma\|\mathbf{y} - \mathbf{x}\| \; \forall \mathbf{x}, \mathbf{y} \in N(\mathbf{x}_*, r)$.

6 Допрограмовий етап

6.1 Визначимо початкове наближення, побудувавши графіки кривих системи

Побудуємо графіки функцій системи і виберемо початкове наближення: знайдемо координати точок перетину кривих, що відповідають 1 і 2-му рівнянням системи



$$x^{(0)} = (0.95; 0.3728)^T$$

6.2 Реалізація методу Ньютона

Будемо знаходити послідовні наближення за формулами:

$$\mathbf{W}\left(\mathbf{x}^{(k)}\right)\Delta\mathbf{x}^{(k)} = -\mathbf{F}\left(\mathbf{x}^{(k)}\right)$$
$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta\mathbf{x}^{(k)}$$

Покладемо k=0.
Знайдемо матрицю Якобі

$$\mathbf{W}\left(\mathbf{x}\right) = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{pmatrix} = \begin{pmatrix} \frac{y}{\cos^2(xy+0{,}4)} - 2x & \frac{x}{\cos^2(xy+0{,}4)} \\ 1,6x & 4y \end{pmatrix}$$

$$\mathbf{W}\left(\mathbf{x}^{(0)}\right) = \begin{pmatrix} \frac{0{,}3728}{\cos^2(0{,}3728\cdot 0{,}95+0{,}4)} - 2\cdot 0,95 & \frac{0{,}95}{\cos^2(0{,}3728\cdot 0{,}95+0{,}4)} \\ 1,6\cdot 0,95 & 4\cdot 0,3728 \end{pmatrix} =$$

$$= \mathbf{W}\left(\mathbf{x}^{(0)}\right) = \begin{pmatrix} -1,198 & 1,7883 \\ 1,52 & 1,4912 \end{pmatrix}$$

$$\mathbf{F}\left(\mathbf{x}^{(0)}\right) = \begin{pmatrix} \operatorname{tg}(xy+0,4) - x^2 \\ 0,8x^2 + 2y^2 - 1 \end{pmatrix} = \begin{pmatrix} \operatorname{tg}(0,95\cdot 0,3728 + 0,4) - 0,95^2 \\ 0,8\cdot 0,95^2 + 2\cdot 0,3728^2 - 1 \end{pmatrix} = \begin{pmatrix} 0,0369 \\ -0,00004032 \end{pmatrix}$$

Запишемо систему рівнянь відносно $\Delta\mathbf{x}^{(0)}$:

$$\begin{pmatrix} -1,198 & 1,7883 \\ 1,52 & 1,4912 \end{pmatrix}\begin{pmatrix} \Delta\mathbf{x}_1^{(0)} \\ \Delta\mathbf{x}_2^{(0)} \end{pmatrix} = \begin{pmatrix} -0,0369 \\ 0,00004032 \end{pmatrix}$$

Запишемо систему рівнянь в матричному вигляді і розв'яжемо її **методом Гауса**

$$\left(\begin{array}{cc|c} -1.198 & 1.7883 & -0.0369 \\ 1.52 & 1.4912 & 0.00004032 \end{array}\right)$$

1-ий рядок ділимо на -1.198

$$\left(\begin{array}{cc|c} 1 & -1.49274 & 0.0308013 \\ 1.52 & 1.4912 & 0.00004032 \end{array}\right)$$

від 2-ого рядка віднімаємо 1-ий рядок, помножений на 1.52

$$\left(\begin{array}{cc|c} 1 & -1.49274 & 0.0308013 \\ 0 & 3.76016 & -0.046(7) \end{array}\right)$$

2-ий рядок ділимо на 3.76016 $\left(\begin{array}{cc|c} 1 & -1.49274 & 0.0308013 \\ 0 & 1 & -0.0124403 \end{array}\right)$

до 1-ого рядка додамо 2-ий рядок, помножений на 1.49274 $\left(\begin{array}{cc|c} 1 & 0 & 0.0122312 \\ 0 & 1 & -0.0124403 \end{array}\right)$

Отже, маємо:

$$\begin{pmatrix} \Delta\mathbf{x}_1^{(0)} \\ \Delta\mathbf{x}_2^{(0)} \end{pmatrix} = \begin{pmatrix} 0.0122312 \\ -0.0124403 \end{pmatrix}$$

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \Delta\mathbf{x}^{(0)} = \begin{pmatrix} 0.95 \\ -0.3728 \end{pmatrix} + \begin{pmatrix} 0.0122312 \\ -0.0124403 \end{pmatrix} = \begin{pmatrix} 0.9622312 \\ -0.3852403 \end{pmatrix}$$

$$\Delta^{(1)} = \max\{0.0122312; 0.0124403\} = 0.0124403 > 0,00001 = \epsilon$$

Покладемо k=1 і **продовжуємо ітераційний процес** програмними методами

Функція для розв'язання нелійнійних систем методом Ньютона

In [93]:
```
eps = 0.00001
appr = np.array([0.95, 0.3728])
```

In [94]:
```
func1= lambda x,y: np.tan(x*y+0.4)-x**2
func2 = lambda x,y: 0.8*x**2+2*y**2-1
```

In [95]:
```
yak_df1_x = lambda x,y: (y/np.cos(x*y+0.4)**2)-2*x
yak_df1_y = lambda x,y: (x/np.cos(x*y+0.4)**2)
yak_df2_x = lambda x,y: 1.6*x
yak_df2_y = lambda x,y: 4*y
```

In [132]:
```
def newtons_method_solve(x, y, appr, eps):
    find_appr = lambda l: max(abs(l[0]),abs(l[1]))
    res, appr_hist = [], []
    while True:
        yakobi_matr= np.array([[yak_df1_x(appr[0], appr[1]),
                                yak_df1_y(appr[0], appr[1])],
                           [yak_df2_x(appr[0], appr[1]),
                            yak_df2_y(appr[0], appr[1])]])
        f_appr = np.array([-func1(appr[0], appr[1]),
                           -func2(appr[0], appr[1])])
        appr_x = np.linalg.solve(yakobi_matr, f_appr)
        appr = appr + appr_x
        res.append(appr)
        appr_hist.append(find_appr(appr_x))
        if find_appr(appr_x) < eps:
            break
    return appr, res, appr_hist
```

In [133]:
```
solution, newt_iters, approximations = newtons_method_solve(x, y, appr, eps)
solution, newt_iters, approximations
```

Out[133]:
```
(array([0.96195361, 0.36035829]),
 [array([0.96222914, 0.36036171]),
  array([0.96195365, 0.36035829]),
  array([0.96195361, 0.36035829])],
 [0.012438285969650409, 0.00027548671910544053, 4.1531855722066935e-08])
```

In [129]:
```
for i in approximations:
    print("%01.8f" % round(i,8))
```

```
0.01243829
0.00027549
0.00000004
```

7 Результати

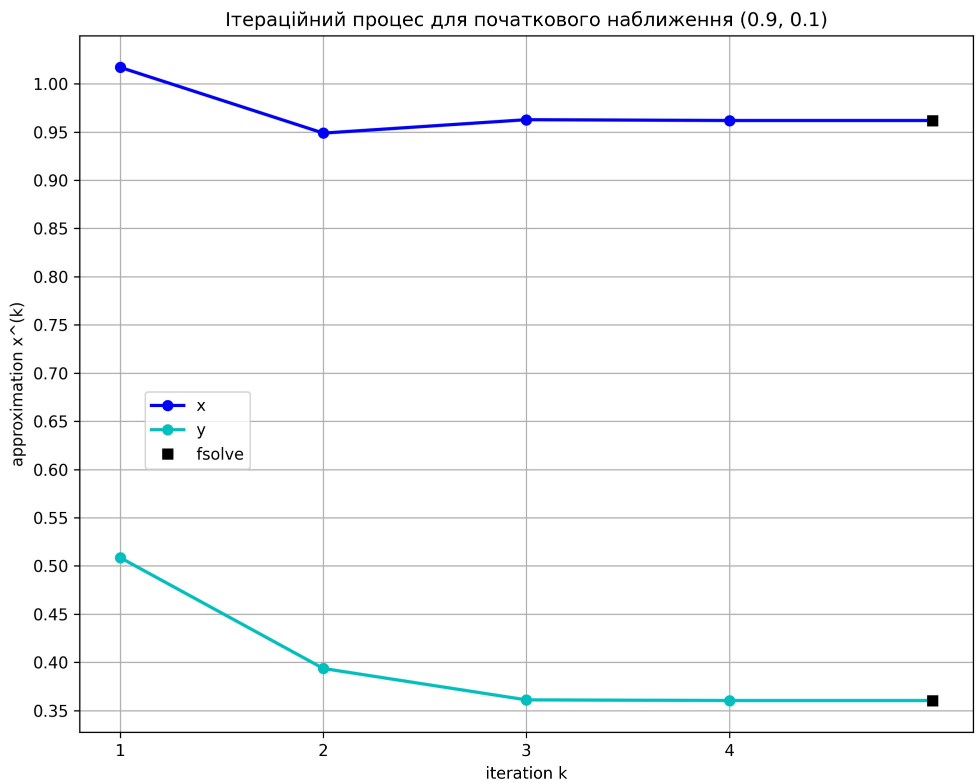
7.1 Табличне представлення:

№ ітерації	x	y	$\Delta^{(k)}$
0	0.95	0.3728	$-$
1	0.96223	0.36036	0.01243829
2	0.96195	0.36036	0.00027549
3	0.96195361	0.36035829	0.00000004

Відповідь: $x = (0.96195361, 0.36035829)^T$

7.2 Графічне представлення:

Зобразимо ітераційний процес на графіку. Для наочості речузьтатів, візьмемо інше початкове наближення:


Ітераційний процес для початкового наближення (0.9, 0.1)
approximation x^(k)
iteration k
x
y
fsolve
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
1
2
3
4

8 Перевірка

Перевірка

$$\begin{cases} \text{tg}(xy + 0,4) = x^2 \\ 0,8x^2 + 2y^2 = 1 \end{cases}$$

In [136]:
```
func1= lambda x,y: np.tan(x*y+0.4)-x**2
func2 = lambda x,y: 0.8*x**2+2*y**2-1
```

In [139]:
```
round(func1(solution[0], solution[1]),5)
```

Out[139]:
```
-0.0
```

In [140]:
```
round(func2(solution[0], solution[1]),5)
```

Out[140]:
```
0.0
```

Підставивши розв'язки в систему, отримуємо відповідні результати (округливши їх до 5 знаків після коми):

$$\begin{cases} \text{tg}(0,96195 \cdot 0,36036 + 0,4) - 0,96195^2 \approx 0 \\ 0,8 \cdot 0,96195^2 + 2 \cdot 0,36036^2 - 1 \approx 0 \end{cases}$$

9 Інші початкові наближення

Інші початкові наближення

In [141]:
```
appr_list = [[2.5, -2.5], [0.01, -50], [-4,-0.3]]
```

In [144]:
```
for pair in appr_list:
    print('Numb of iter: ', newtons_method_solve(x, y, pair, eps)[2].shape, end=' ')
    print(", result: ", newtons_method_solve(x, y, pair, eps)[0])
```

```
Numb of iter:  (8,) , result:  [ 0.38293531 -0.66433743]
Numb of iter:  (10,) , result:  [ 0.38293531 -0.66433743]
Numb of iter:  (7,) , result:  [-0.96195361 -0.36035829]
```

Бачимо, що при заданні інших початкових наближень(не близьких до розв'язку), кількість ітерацій та відповідь змінюються. Переконуємось в необхідності задавати "хороше"початкове наближення.

10 Розв'язок з допомогою функції fsolve

```
In [79]: import math
         import matplotlib.patches as mpatches
```

Метод Ньютона

Розв'яжемо систему нелінійних рівнянь

$$\begin{cases} tg(xy+0,4)=x^2 \\ 0,8x^2+2y^2=1 \end{cases}$$

```
In [65]: def F(variables) :
             (x,y)= variables
             f1 = np.tan(x*y +0.4) - x**2
             f2 = 0.8*x**2 + 2*y**2 - 1
             return [f1,f2]
```

```
In [66]: result = fsolve(F, (1, 1))
         print(result)

[0.96195361 0.36035829]
```

```
In [75]: func1= lambda x,y: np.tan(x*y+0.4)-x**2
         func2 = lambda x,y: 0.8*x**2+2*y**2-1
```

Маємо змогу порівняти результати:
Відповідь, знайдена з допомогою алгоритму методу простих ітерацій:
$x = (0.96195361, 0.36035829)^T$
Відповідь, знайдена з допомогою функції fsolve:
$x = (0.96195361, 0.36035829)^T$
Можемо помітити, що відповіді співпадають

11 Висновки

В першому завданні розв'язали систему нелінійних рівнянь з допомогою методу простих ітерацій, а в другому - з допомогою методу Ньютона. Записали результати до таблиць та прослідкували за ітераційним процесом з допомогою графіків. З'ясували, що при зміні початкових наближень, кількість ітерацій також змінюється. Переконалися, що метод Ньютона допомагає знайти розв'язок у меншу кількість ітерацій, однак потребує задання "хорошого"початкового наближення, що є недоліком в порівнянні з методом простих ітерацій. Робота дала змогу попрактикуватися в застосовуванні ітераційних чисельних методів для розв'язання нелінійних систем