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

Лабораторна робота № 1
з курсу «Чисельні методи»
з теми «Методи розв’язання
нелінійних алгебраїчних рівнянь»
Варіант № 13

Виконала:
студентка 2 курсу групи КА-02
Шапошнікова Софія Віталіївна
Перевірила:
старший викладач
Хоменко Ольга Володимирівна

Київ-2022

Тема: методи розв'язання нелінійних алгебраїчних рівнянь

Мета роботи: навчитися застосовувати чисельні методи розв'язання нелінійних алгебраїчних рівнянь, навчитися відокремлювати та уточнювати корені нелінійних рівнянь.

Хід роботи:

1. Допрограмовий етап. Відокремити корені заданих рівнянь, тобто для кожного з коренів визначити інтервал, до якого відповідний корінь належить та є на ньому єдиним.
2. Запрограмувати методи половинного ділення, хорд та дотичних та знайти за їх допомогою корені рівнянь. Критерієм закінчення мають бути нерівності для методу бісекції (a та b - кінці інтервалу) $|b - a| < \varepsilon$ та $|f(x_k)| < \varepsilon$, для методів хорд та Ньютона $|x_k - x_{k-1}| < \varepsilon$ та $|f(x_k)| < \varepsilon$, де $\varepsilon = 0.00001$.
3. Написати програму мовою Pyton, яка використовуючи функції roots та fsolve, знаходить корені заданих рівнянь та будує відповідні функції (див приклад в класрумі).

Завдання 1

Опис процесу відокремлення коренів

$$150x^7 + 249x^6 - 661x^5 - 905x^4 + 885x^3 + 917x^2 - 290x - 256 = 0$$

Розв'язок:

1) Рівняння має сім коренів

   Скористаємося теоремою Декарта про кількість коренів алгебраїчного рівняння.

   Теорема 2.5.(теорема Декарта про кількість дійсних коренів алгебраїчного рівняння)

   Число S_1 додатніх коренів алгебраїчного рівняння з врахуванням їх кратностей $P_n(x) = 0$ дорівнює числу змін знаків в послідовності коефіцієнтів a_n, a_{n-1}, a_{n-2},..., a_0 (коефіцієнти, які дорівнюють нулю не враховуються) многочлена $P_n(x)$ або менше цього числа на парне число. Число S_2 від'ємних коренів алгераїчного рівняння з врахуванням їх кратностей $P_n(x) = 0$ дорівнює числу змін знаків в послідовності a_n, a_{n-1}, a_{n-2},..., a_0 многочлена $P_n(-x)$ або менше цього числа на парне число.

Кількість змін знаку коефіцієнтів P(x) - 3, додатніх коренів 3 або 1

Кількість змін знаку коефіцієнтів P(-x) - 4, від'ємних коренів 4, 2 або 0

2) З наведеної нижче теореми:

Теорема 3.
Нехай A = max{|an−1|, ..., |a0|}, B = max{|an|, ..., |a1|},де ak, k = 0, 1, ..., n – коефіцієнти рівняння. Тоді модулі всіх коренів рівняння задовольняють нерівність:

$$\frac{1}{1+\dfrac{B}{|a_0|}} \le |x_{*i}| \le 1+\frac{A}{|a_n|},$$

Маємо:

$$\frac{1}{1+\frac{917}{256}} \le |x_{*i}| \le 1+\frac{917}{150}$$

$$\frac{1}{4,58203125} \le |x_{*i}| \le 7,11(3)$$

$$0,2182438192668 \le |x_{*i}| \le 7,11(3)$$

Додатні корені:

$$0,2182438192668 \le x_{*i} \le 7,11(3)$$

Від'ємні корені:

$$-7,11(3) \le x_{*i} \le -0,2182438192668$$

3) Скористаємося теоремою Лагранжа, наведеною нижче:

Теорема 2.4.(теорема Лагранжа про верхню межу додатніх коренів)

Нехай $a_n > 0$ і a_i – перший від'ємний коефіцієнт в послідовності a_n, a_{n-1}, a_{n-2},..., a_0; C – найбільша з абсолютних величин від'ємних коефіцієнтів. Тоді за верхню межу додатніх коренів рівняння (2.1) може бути прийняте число

$$R = 1 + \sqrt[n-i]{\frac{C}{a_n}}.$$

$$150\,x^7 + 249\,x^6 - 661\,x^5 - 905\,x^4 + 885\,x^3 + 917\,x^2 - 290\,x - 256 = 0$$

$$R = 1 + \sqrt[n-1]{\frac{C}{a_n}} = 1 + \sqrt[6]{\frac{905}{150}} = 1 + \sqrt[6]{6,0(3)} \approx 2,34925$$

Звідси: $x_{*i} \leq 2,34925$

Аналогічно оцінимо нижню межу. Покладемо: $x_{*i} = -x$ та помножимо отримане рівняння на -1, щоб виконувалося $a_7 > 0$

Тоді маємо:

$$150\,x^7 - 249\,x^6 - 661\,x^5 + 905\,x^4 + 885\,x^3 - 917\,x^2 - 290\,x + 256 = 0$$

$$R = 1 + \sqrt[n-1]{\frac{C}{a_n}} = 1 + \sqrt[6]{\frac{917}{150}} \approx 2,35222$$

Звідси: $x_{*i} \geq -2,35222$

4) Перевіримо виконання необхідної умови дійсності коренів (за теоремою Гюа)

Теорема 2.6.(теорема Гюа про необхідні умови дійсності всіх коренів алгебраїчного рівняння)

Якщо алгебраїчне рівняння (2.1) має всі дійсні корені, то квадрат кожного некрайнього коефіцієнта більше добутку двох його сусідніх коефіцієнтів.

$$(a_k)^2 > a_{k+1} \cdot a_{k-1}$$

$$249^2 > 150 \cdot (-661) \Rightarrow 62001 > -99150$$
$$(-661)^2 > 249 \cdot (-905) \Rightarrow 436921 > -225345$$
$$(-905)^2 > (-661) \cdot 885 \Rightarrow 819025 > -584985$$
$$885^2 > (-905) \cdot 917 \Rightarrow 783225 > -829885$$
$$917^2 > 885 \cdot (-290) \Rightarrow 840889 > -256650$$
$$(-290)^2 > 917 \cdot (-256) \Rightarrow 84100 > -234752$$

Отже, умови виконуються

5) Застосуємо теорему Штурма.

Теорема 2.9. (Штурма)

Нехай $f(x) = P(x)$– поліном без кратних коренів. Утворимо послідовність многочленів:

$$f_0 = f(x),$$
$$f_1 = f'(x),$$
$$f_2 = -[f_0 \ mod \ f_1],$$

$$f_{i+1} = -[f_{i-1} \ mod \ f_i], \ \ i = 1, \ldots, n-1,$$

тобто, починаючи з f_2, кожний наступний многочлен є залишком від ділення двох попередніх многочленів, взятим з протилежним знаком. Кількість дійсних коренів рівняння $f(x) = 0$ на довільному відрізку $[a; b]$ дорівнює різниці між кількістю змін знаку у цій послідовності при $x = a$ та кількістю змін знаку при $x = b$.

Утворимо послідовність многочленів:

$$f_0 = 150x^7 + 249x^6 - 661x^5 - 905x^4 + 885x^3 + 917x^2 - 290x - 256$$

$$f_1 = f'(x) = 1050x^6 + 1494x^5 - 3305x^4 - 3620x^3 + 2655x^2 + 1834x - 290$$

$$f_2 = -[f_0 \bmod f_1] \approx 293.4702x^5 + 275.89184x^4 - \\ - 628.35102x^3 - 565.0551x^2 + 310.70286x + 246.17551$$

$$f_3 = -[f_1 \bmod f_2] \approx 1533.3690976719x^4 + 512.99046374719x^3 - 2519.3289199309x^2 -$$

$$- 416.55642189398x + 715.20395720727$$

$$f_4 = -[f_2 \bmod f_3] \approx 205.63225172224x^3 + 193.35108919753x^2 - 222.09772498145x -$$

$$- 163.28637723501$$

$$f_5 = -[f_3 \bmod f_4] \approx 10.149034519412x^2 - 202.12529671642x - 22.328013548315$$

$$f_6 = -[f_4 \bmod f_5] \approx 20.261262421458x + 422.56874383283$$

$$f_7 = -[f_5 \bmod f_6] \approx 8607.7460042387$$

Побудуємо таблицю:

	-2.25	-2	-1.75	-1.5	-0.7	-0.6	0.7	1.25	1.45
f_0	-	+	+	-	+	-	+	-	+
f_1	+	+	-	-	+	-	+	-	+
f_2	-	-	-	+	-	+	+	-	+
f_3	+	+	+	-	-	+	-	-	+
f_4	-	-	-	-	+	-	-	+	+
f_5	+	+	+	+	+	-	-	+	+
f_6	-	-	-	-	-	-	+	+	+
f_7	+	+	+	+	+	+	+	+	+
кількість змін знаку	7	6	6	5	4	3	2	1	0

На відрізку [-2.25;-2] маємо 1 корінь, оскільки різниця між кількістю змін знаку

дорівнює 1.

На відрізку [-2;-1.75] маємо 1 корінь, оскільки різниця між кількістю змін знаку дорівнює 1.

На відрізку [-1.75;-1.5] жодного кореня, оскільки різниця між кількістю змін знаку дорівнює 0.

На відрізку [-1.5;-0.7] маємо 1 корінь, оскільки різниця між кількістю змін знаку дорівнює 1.

На відрізку [-0.7;-0.6] маємо 1 корінь, оскільки різниця між кількістю змін знаку дорівнює 1.

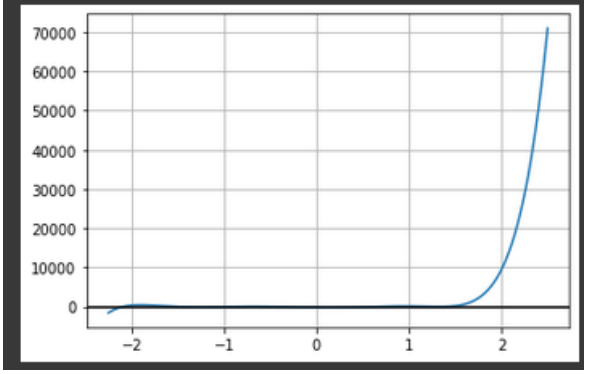
На відрізку [-0.6;0.7] маємо 1 корінь, оскільки різниця між кількістю змін знаку дорівнює 1.

На відрізку [0.7;1.25] маємо 1 корінь, оскільки різниця між кількістю змін знаку дорівнює 1.

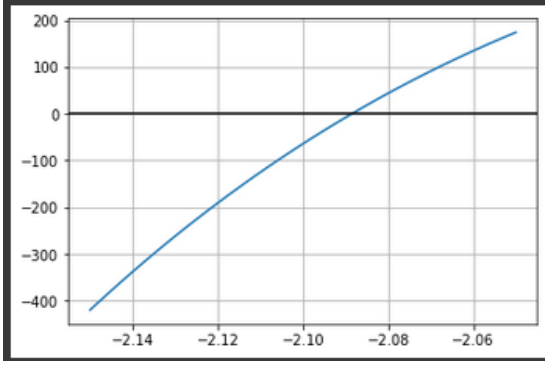
На відрізку [1.25;1.45] маємо 1 корінь, оскільки різниця між кількістю змін знаку дорівнює 1.

Графічний спосіб відокремлення коренів наведений нижче:

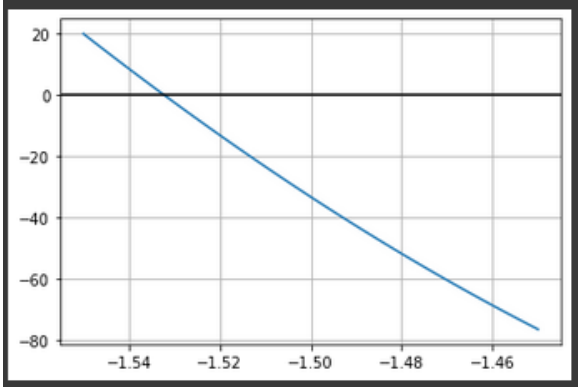
```
t=np.linspace(-2.25,2.5,10000)
plt.plot(t,eq(t))
plt.axhline(0, color='black')
plt.grid()
```



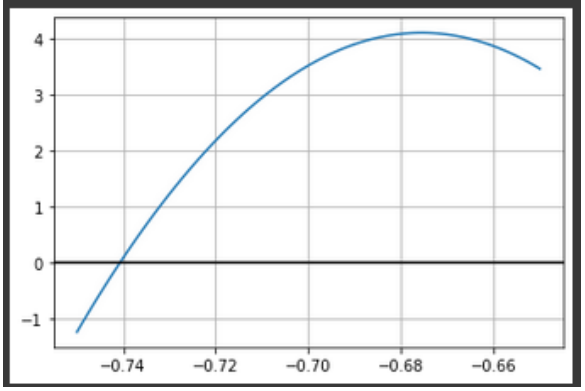

```
t=np.linspace(-2.15,-2.05,100)
plt.plot(t,eq(t))
plt.axhline(0, color='black')
plt.grid()
```



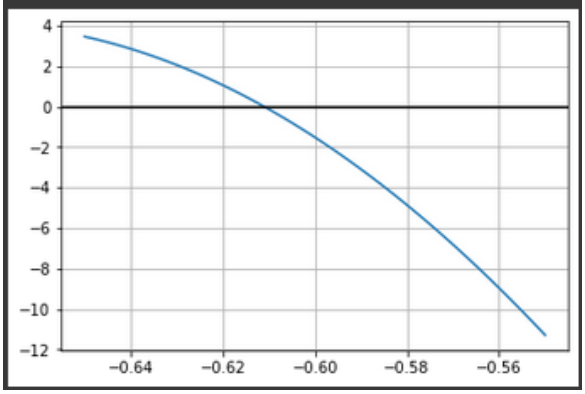
```
t=np.linspace(-1.55,-1.45,100)
plt.plot(t,eq(t))
plt.axhline(0, color='black')
plt.grid()
```



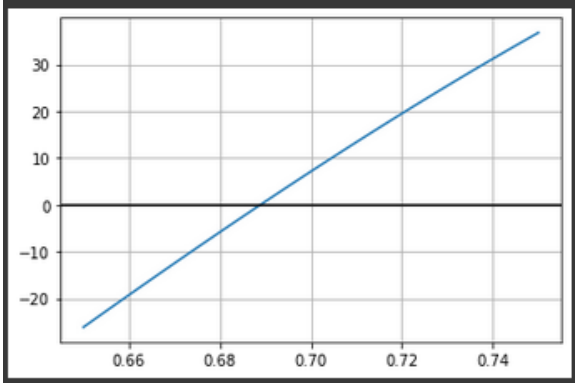

```
t=np.linspace(-0.75,-0.65,100)
plt.plot(t,eq(t))
plt.axhline(0, color='black')
plt.grid()
```



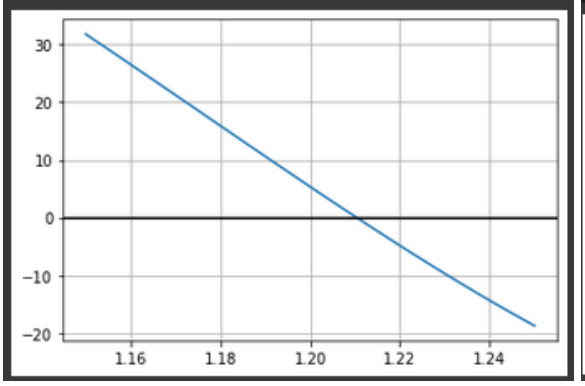

```
t=np.linspace(-0.65,-0.55,100)
plt.plot(t,eq(t))
plt.axhline(0, color='black')
plt.grid()
```



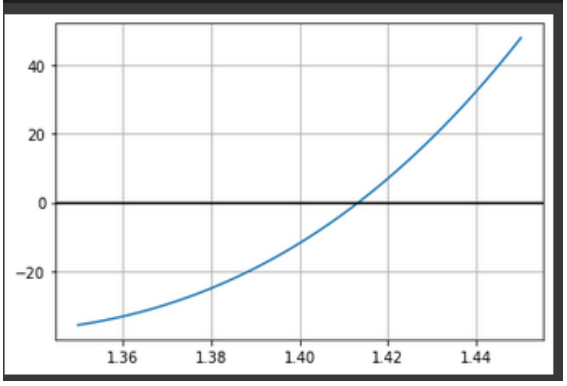

```
t=np.linspace(0.65,0.75,100)
plt.plot(t,eq(t))
plt.axhline(0, color='black')
plt.grid()
```




```
t=np.linspace(1.15,1.25,100)
plt.plot(t,eq(t))
plt.axhline(0, color='black')
plt.grid()
```




```
t=np.linspace(1.35,1.45,100)
plt.plot(t,eq(t))
plt.axhline(0, color='black')
plt.grid()
```



На цьому процес відокремлення завершено

Код програм:

1) Метод бісекції (половинного ділення)

```
def bisection_method(a,b):

 eps = 0.00001

 i = 0

 while True:

   i = i+1

   c = (a+b) * 0.5

   res = eq(c)

   if eq(a)/abs(eq(a))*res < 0: b = c

   elif eq(b)/abs(eq(b))*res < 0:a = c

   else: return c

   print('Iteration: ', i)

   print('a = {0:.6f},b = {1:.6f} : eq(a)= {2:.6f}, eq(b)= {3:.6f}'.format(a, b,
eq(a),eq(b)))

   if abs(b-a)<eps and abs(res)<eps:return c
```

2) Метод хорд

```
def chord_method(a, b):

 eps = 0.00001

 i = 0

 if (ddx(a+b)/2)*eq(a)>0:
```

```
    n=a
    eqn=eq(a)
    x0=b
  elif (ddx(a+b)/2)*eq(b)>0:
    n=b
    eqn=eq(b)
    x0=a
  while True:
    i=i+1
    x=x0-(eq(x0)/(eq(x0)-eqn))*(x0-n)
    print('Iteration: ', i)
    print('x_0 = {0:.6f},x = {1:.6f} : eq(x_0)= {2:.6f}, eq(x)=
{3:.6f}'.format(x0, x, eq(x0),eq(x)))
    if(abs(x-x0)<eps and abs(eq(x))<eps): break
    else:x0=x
  return x
```

3) Метод Ньютона

```
def newtons_method(a, b):
  eps = 0.00001
  i = 0
  x0=float()
  if ddx(a)*eq(a)>0:x0=a
```

```
elif ddx(b)*eq(b)>0:x0=b

while True:

  i=i+1

  x=x0-eq(x0)/dx(x0)

  print('Iteration: ', i)

  print('x = {0:.6f}, eq(x) = {1:.6f}'.format(x,eq(x)))

  if(abs(x-x0)<eps and abs(eq(x))<eps): break

  else:x0=x

return x
```

4) fsolve

Завдання 2

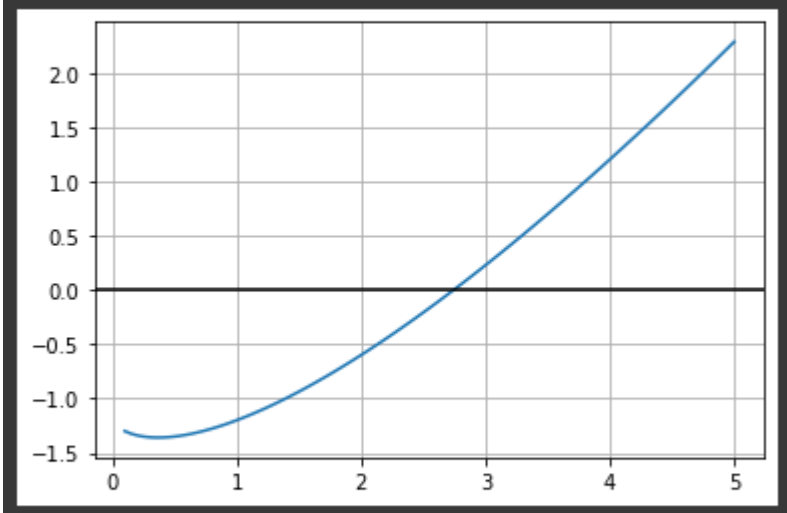
Потрібно відокремити корені рівняння $xlgx - 1.2 = 0$. В цьому випадку корисно попередньо знайти область визначення $f(x)$ та проміжки зростання\спадання функції.

Область визначення $f(x) = 2lgx - x + 1$ проміжок $(0; \infty)$. Знайдемо інтервали монотонності. $f'(x) = lg(x) + \frac{1}{ln10}$,

$f(x)$ зростає на $(0; 10^{-\frac{1}{ln10}})$, спадає на $(10^{-\frac{1}{ln10}}, \infty)$.

$x = 10^{-\frac{1}{ln10}} \approx 0,367879.$

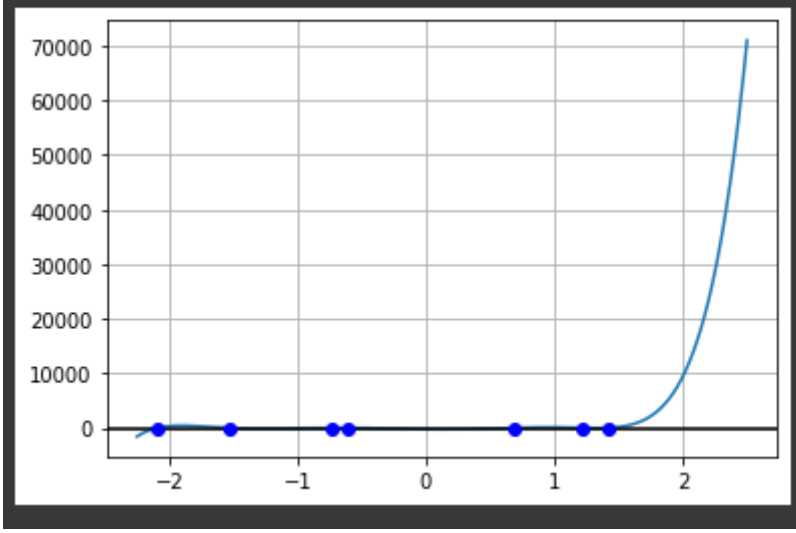
```
x=np.linspace(0.1,5,100)
plt.plot(x,x*np.log10(x)-1.2)
plt.axhline(0, color='black')
plt.grid()
```




```
def eq2(x): return np.log10(x)*x-1.2
x = fsolve(eq2,1.5)
print(x, eq(x))
```

Знайдені корені рівняння зобразимо графічно:

```
# show the roots from bisection method
y = [eq(x) for x in bisect]
t=np.linspace(-2.25,2.5,10000)
plt.plot(t,eq(t))
plt.axhline(0, color='black')
plt.plot(bisect, y, 'bo')
plt.grid()
```



```
# Let's check, which of those methods is technically faster

time_dict = {}

intervals = [(-2.15, -2.05), (-1.55, -1.45), (-0.75,-0.65), (-0.65,-0.55),
(0.65,0.75), (1.15,1.25), (1.35,1.45)]

start1 = time.time()

for i,j in intervals:

 bisection_method(i,j)

end1 = time.time()

start2 = time.time()

for i,j in intervals:

 chord_method(i,j)

end2 = time.time()

start3 = time.time()

for i,j in intervals:

 newtons_method(i,j)

end3 = time.time()

time_dict['bisection_method'] = end1 - start1

time_dict['chord_method'] = end2 - start2

time_dict['newtons_method'] = end3 - start3
```

```
print(time_dict)
```

```
{'bisection_method': 0.05203413963317871, 'chord_method': 0.020914554595947266, 'newtons_method': 0.006373167037963867}
```

Бачимо, що метод Ньютона виявився найшвидшим, а метод хорд - найповільнішим.

Висновки

Метод Ньютона (метод Дотичних) – найефективніший, адже він забезпечує збіжність за мінімальне число ітерацій. В свою чергу перевага методу хорд є в тому, що він простіший і не потребує використання похідних першого та другого порядку, але збіжність відбувається повільніше ніж у методі Ньютона