Розв'язання нелінійних рівнянь

Варіант 9, Калитюк Дар'я, ФІ-83

$$x^5 + x^4 \text{ -}2x^3 \text{ -}9x^2 \text{ -}3x \text{ -}2 = 0$$

In [328]:

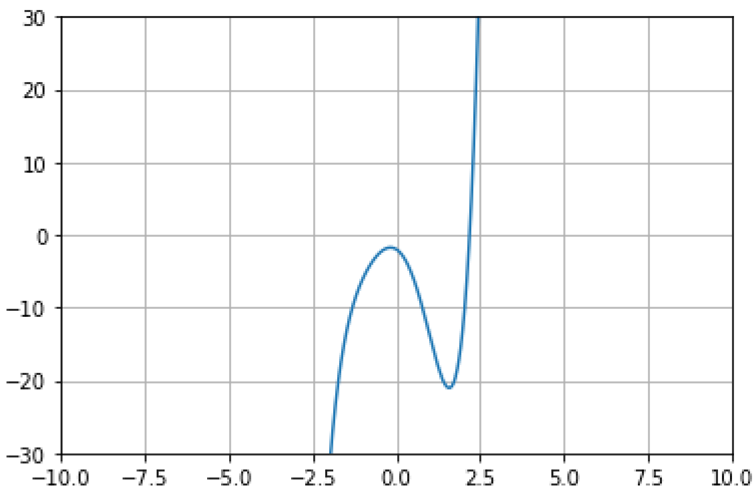
```
import scipy
from scipy import signal
import numpy as np
import math
import matplotlib.pyplot as plt
import pandas as pd
```

In [329]:

```
func = lambda x: x**5+x**4-2*x**3-9*x**2-3*x-2
plt.grid()
plt.axis([-10, 10, -30, 30])
plt.plot(x, func(x))
```

Out[329]:

```
[<matplotlib.lines.Line2D at 0x1a4a15eba88>]
```



Теорема про кільце:

In [332]:

```
a_n=[-2, -3, -9, -2, 1, 1]
A=abs(max(a_n[1:], key=abs))
B=abs(max(a_n[:(len(a_n)-1)], key=abs))
print(f'A = max(|a_i|) (i=0, 1, ..., 4) = {A}\nB = max(|a_i|) (i=1, 2, ..., 5) = {B}')
left=abs(a_n[0])/(B+abs(a_n[0])); rigth=(abs(a_n[len(a_n)-1])+A)/abs(a_n[len(a_n)-1])
print(f'Всі корені лежать у кільці: {round(left, 6)} =< |x*| =< {rigth}')
```

```
A = max(|a_i|) (i=0, 1, ..., 4) = 9
B = max(|a_i|) (i=1, 2, ..., 5) = 9
Всі корені лежать у кільці: 0.181818 =< |x*| =< 10.0
```

Теорема про верхню межу додатніх коренів:

In [256]:

```
a_neg=[]
for i in range(len(a_n)):
    if a_n[i]<0: a_neg.append(a_n[i])
B=abs(max(a_neg, key=abs))
print(f'B = max(|a_i|) (a_i<0; i=0, ..., n) = {B}')
def find(a):
    m=None
    for i in reversed(range(len(a))):
        if a[i]<0:
            m=i
            print(f'm = max(i) (a_i<0; i=0, ..., n) = {i}')
            return m
            break
    if m is None: print('Рівняння не має додатніх коренів.')

R=1+math.pow((B/a_n[len(a_n)-1]), 1/(len(a_n)-1-find(a_n)))
print(f'R = {R} - верхня межа додатніх коренів.')
```

```
B = max(|a_i|) (a_i<0; i=0, ..., n) = 9
m = max(i) (a_i<0; i=0, ..., n) = 3
R = 4.0 - верхня межа додатніх коренів.
```

Для знаходження нижньої межі додатніх коренів зробимо заміну x=1/y і отримаємо наступне рівняння:

$$f(y) = -2y^5 - 3y^4 - 9y^3 - 2y^2 + y + 1$$

Домножимо на -1:

$$-f(y) = 2y^5 + 3y^4 + 9y^3 + 2y^2 - y - 1$$

In [331]:

```
a_n=[-2, -3, -9, -2, 1, 1]
a_n.reverse()
a_n=[-a for a in a_n]
print(a_n)
R=1+math.pow((B/a_n[len(a_n)-1]), 1/(len(a_n)-1-find(a_n)))
print(f'R = {round(1/(R), 6)} - нижня межа додатніх коренів.')
```

```
[-1, -1, 2, 9, 3, 2]
m = max(i) (a_i<0; i=0, ..., n) = 1
R = 0.407087 - нижня межа додатніх коренів.
```

Для знаходження нижньої межі від'ємних коренів зробимо заміну x=-x і отримаємо наступне рівняння:

$$f(x) = -x^5 + x^4 + 2x^3 - 9x^2 + 3x - 2$$

Домножимо на -1:

$$-f(x) = x^5 - x^4 - 2x^3 + 9x^2 - 3x + 2$$

In [258]:

```
a_n=[2, -3, 9, -2, -1, 1]
R=1+math.pow((B/a_n[len(a_n)-1]), 1/(len(a_n)-1-find(a_n)))
print(f"R = {-R} - нижня межа від'ємних коренів.")
```

```
m = max(i) (a_i<0; i=0, ..., n) = 4
R = -10.0 - нижня межа від'ємних коренів.
```

Для знаходження верхньої межі від'ємних коренів зробимо заміну x=-1/y і отримаємо наступне рівняння:

$$f(y) = 2y^5 - 3y^4 + 9y^3 - 2y^2 - y + 1$$

In [261]:

```
a_n.reverse()
R=1+math.pow((B/a_n[len(a_n)-1]), 1/(len(a_n)-1-find(a_n)))
print(f"R = {round((-1/R), 3)} - верхня межа від'ємних коренів.")
```

```
m = max(i) (a_i<0; i=0, ..., n) = 4
R = -0.182 - верхня межа від'ємних коренів.
```

Теорема Гюа про наявність комплесних коренів:

In [262]:

```
a_n=[-2, -3, -9, -2, 1, 1]
for i in range(1, len(a_n)-1):
    if a_n[i]**2<a_n[i-1]*a_n[i+1]:
        print(f'Існує таке k, що (a_k)^2<a_(k-1)*a_(k+1), k = {i}, отже рівняння має ко
мплексні корені')
        break
    else:
        print('Рівняння не має комплексних коренів.')
```

```
Існує таке k, що (a_k)^2<a_(k-1)*a_(k+1), k = 1, отже рівняння має комплек
сні корені
```

Теорема Штурма:

In [ ]:

```
n=len(a_n)
f=np.poly1d([1, 1, -2, -9, -3, -2])
f0=f
f1=np.poly1d([5, 4, -6, -18, -3])
f2=np.poly1d(scipy.signal.deconvolve(f0, f1)[1])
f3=np.poly1d(scipy.signal.deconvolve(f1, f2)[1])
f4=np.poly1d(scipy.signal.deconvolve(f2, f3)[1])
f5=np.poly1d(scipy.signal.deconvolve(f3, f4)[1])
mas=[f2, f3, f4, f5]
```

In [303]:

```
for j in range(len(mas)):
    for i in range(len(mas[j])+1):
        mas[j][i]=-mas[j][i]
print('Загальна формула: f_(i+1)=-[f_(i-1) mod f_i]')
print(f0, ' = f0')
print(f1, ' = f1')
print(mas[0], ' = f2')
print(mas[1], ' = f3')
print(mas[2], ' = f4')
print(mas[3], ' = f5')
```

```
Загальна формула: f_(i+1)=-[f_(i-1) mod f_i]
   5     4     3     2
1 x + 1 x - 2 x - 9 x - 3 x - 2  = f0
   4     3     2
5 x + 4 x - 6 x - 18 x - 3  = f1
      3        2
0.96 x + 5.16 x + 1.68 x + 1.88  = f2
            3          2
-3.553e-15 x - 108.2 x - 12.24 x - 41.8  = f3
            2
-2.924e+16 x - 3.307e+15 x - 1.129e+16  = f4
            2
-1.421e-14 x - 3.553e-15 x - 7.105e-15  = f5
```

In [311]:

```
a=0.41
b=4
c=-0.182
d=-10
eps=0.00001
```

In [312]:

```
pd.DataFrame({'f': ['f0', 'f1', 'f2', 'f3', 'f4', 'f5'], f'f({a})':[f'{f0(a)}', f'{f1(a)}', f'{f2(a)}', f'{f3(a)}', f'{f4(a)}', f'{f5(a)}'], f'f({b})':[f'{f0(b)}', f'{f1(b)}', f'{f2(b)}', f'{f3(b)}', f'{f4(b)}', f'{f5(b)}'], f'f({c})':[f'{f0(c)}', f'{f1(c)}', f'{f2(c)}', f'{f3(c)}', f'{f4(c)}', f'{f5(c)}'], f'f({d})':[f'{f0(d)}', f'{f1(d)}', f'{f2(d)}', f'{f3(d)}', f'{f4(d)}', f'{f5(d)}']})
```

Out[312]:

	f	f(0.41)	f(4)	f(-0.182)	
0	f0	-4.8408987699	994	-1.739161354910432	
1	f1	-10.971627949999998	1365	0.05862772488000001	
2	f2	3.5023601600000003	152.6	1.73937241472	-458.919
3	f3	-65.00404947916672	-1822.0052083333353	-43.15339114583337	-10739.71
4	f4	-1.7565132780716904e+16	-4.923349186390264e+17	-1.16607357770489e+16	-2.902042194
5	f5	-1.0950884643534663e-14	-2.4868995751603507e-13	-6.929553819645662e-15	-1.3926637

In [313]:

```
print(f'Бачимо, що у точках {c} і {d} подслідовності змінюють знак два рази, отже на ць
ому проміжку рівняння не має коренів, але у точці {a} послідовність змінює знак 2 рази,
а у точці {b} - 1 раз. Отже на відрізку [{a}, {b}] буде лише один дійсний корінь(що так
ож видно з графіку функції).')
```

```
Бачимо, що у точках -0.182 і -10 подслідовності змінюють знак два рази, от
же на цьому проміжку рівняння не має коренів, але у точці 0.41 послідовніс
ть змінює знак 2 рази, а у точці 4 - 1 раз. Отже на відрізку [0.41, 4] буд
е лише один дійсний корінь(що також видно з графіку функції).
```

In [314]:

```
def bisection(x, y):
    z=(x+y)/2
    i=0
    while abs(x-y)>eps:
        i=i+1
        z=(x+y)/2
        if func(x)*func(z)<=0:
            y=z
        else: x=z
        print(f'{i}) a = {x}, b = {y}')
    return (x+y)/2

print('Метод бісекції:')
x_bisection=bisection(a, b)
print(f'Корінь рівняння з проміжку [{a}, {b}] дорівнює: ', x_bisection)
```

```
Метод бісекції:
1) a = 0.41, b = 2.205
2) a = 1.3075, b = 2.205
3) a = 1.75625, b = 2.205
4) a = 1.980625, b = 2.205
5) a = 2.0928125, b = 2.205
6) a = 2.14890625, b = 2.205
7) a = 2.176953125, b = 2.205
8) a = 2.176953125, b = 2.1909765625
9) a = 2.176953125, b = 2.18396484375
10) a = 2.180458984375, b = 2.18396484375
11) a = 2.1822119140625, b = 2.18396484375
12) a = 2.1822119140625, b = 2.18308837890625
13) a = 2.1822119140625, b = 2.182650146484375
14) a = 2.1824310302734373, b = 2.182650146484375
15) a = 2.1824310302734373, b = 2.182540588378906
16) a = 2.1824858093261716, b = 2.182540588378906
17) a = 2.1824858093261716, b = 2.182513198852539
18) a = 2.182499504089355, b = 2.182513198852539
19) a = 2.182506351470947, b = 2.182513198852539
Корінь рівняння з проміжку [0.41, 4] дорівнює:  2.1825097751617433
```

In [315]:

```
def horda(x, y):
    i=0
    z=(x*func(y)-y*func(x))/(func(y)-func(x))
    while abs(func(z))>eps:
        i=i+1
        z=(x*func(y)-y*func(x))/(func(y)-func(x))
        if func(x)*func(z)<=0:
            y=z
        else: x=z
        print(f'{i}) a = {x}, b = {y}')
    return (x*func(y)-y*func(x))/(func(y)-func(x))

print('Метод хорд:')
x_horda=horda(a, b)
print(f'Корінь рівняння з проміжку [{a}, {b}] дорівнює: ', x_horda)
```

```
Метод хорд:
1) a = 0.4273989937790327, b = 4
2) a = 0.44540349737631857, b = 4
3) a = 0.46404907638526394, b = 4
4) a = 0.48337349930024137, b = 4
5) a = 0.5034168341347133, b = 4
6) a = 0.5242215326659143, b = 4
7) a = 0.5458324961089405, b = 4
8) a = 0.5682971142706951, b = 4
9) a = 0.5916652680778225, b = 4
10) a = 0.6159892827438678, b = 4
11) a = 0.6413238156709448, b = 4
12) a = 0.6677256594099593, b = 4
13) a = 0.6952534355920997, b = 4
14) a = 0.7239671506960844, b = 4
15) a = 0.7539275789065208, b = 4
16) a = 0.7851954313414404, b = 4
17) a = 0.8178302649519255, b = 4
18) a = 0.8518890790510111, b = 4
19) a = 0.8874245436950938, b = 4
20) a = 0.924482803466723, b = 4
21) a = 0.9631008046125898, b = 4
22) a = 1.0033031056353106, b = 4
23) a = 1.0450981545981126, b = 4
24) a = 1.0884740542610065, b = 4
25) a = 1.13339389230847, b = 4
26) a = 1.1797907908970593, b = 4
27) a = 1.2275629275867563, b = 4
28) a = 1.2765688939815534, b = 4
29) a = 1.3266238779537793, b = 4
30) a = 1.3774972605247064, b = 4
31) a = 1.4289122809192527, b = 4
32) a = 1.4805484085544187, b = 4
33) a = 1.5320469352003125, b = 4
34) a = 1.5830200428873988, b = 4
35) a = 1.6330632176576616, b = 4
36) a = 1.681770406981758, b = 4
37) a = 1.7287508390709347, b = 4
38) a = 1.7736460405806407, b = 4
39) a = 1.8161454086964628, b = 4
40) a = 1.855998781281158, b = 4
41) a = 1.8930248074684122, b = 4
42) a = 1.9271144822770654, b = 4
43) a = 1.9582298513934022, b = 4
44) a = 1.9863984805198638, b = 4
45) a = 2.0117047083098853, b = 4
46) a = 2.0342789075556857, b = 4
47) a = 2.0542859701393383, b = 4
48) a = 2.071914056854153, b = 4
49) a = 2.0873643844438745, b = 4
50) a = 2.1008425282098497, b = 4
51) a = 2.1125514515190873, b = 4
52) a = 2.1226862633971657, b = 4
53) a = 2.1314305617551055, b = 4
54) a = 2.1389541380362256, b = 4
55) a = 2.145411786860419, b = 4
56) a = 2.150942966950643, b = 4
57) a = 2.1556720837180663, b = 4
58) a = 2.1597091985918175, b = 4
59) a = 2.163151007901276, b = 4
60) a = 2.166081970083323, b = 4
```

```
61) a = 2.1685754916316573, b = 4
62) a = 2.1706951085093804, b = 4
63) a = 2.172495620658846, b = 4
64) a = 2.174024153251442, b = 4
65) a = 2.175321130153689, b = 4
66) a = 2.1764211535306575, b = 4
67) a = 2.177353789301849, b = 4
68) a = 2.1781442619503957, b = 4
69) a = 2.1788140644931886, b = 4
70) a = 2.179381490667614, b = 4
71) a = 2.179862096903529, b = 4
72) a = 2.1802691016695506, b = 4
73) a = 2.180613729488497, b = 4
74) a = 2.180905506435177, b = 4
75) a = 2.181152513349671, b = 4
76) a = 2.181361602381198, b = 4
77) a = 2.181538581861429, b = 4
78) a = 2.1816883739164403, b = 4
79) a = 2.181815148677903, b = 4
80) a = 2.1819224384538307, b = 4
81) a = 2.182013234769744, b = 4
82) a = 2.1820900707919133, b = 4
83) a = 2.182155091292882, b = 4
84) a = 2.182210112012283, b = 4
85) a = 2.182256669998894, b = 4
86) a = 2.1822960662888544, b = 4
87) a = 2.182329402075782, b = 4
88) a = 2.182357609357378, b = 4
89) a = 2.1823814768963907, b = 4
90) a = 2.182401672208322, b = 4
91) a = 2.182418760181089, b = 4
92) a = 2.1824332188404973, b = 4
93) a = 2.1824454526975594, b = 4
94) a = 2.182455804047515, b = 4
95) a = 2.182464562534121, b = 4
96) a = 2.1824719732450153, b = 4
97) a = 2.1824782435633567, b = 4
98) a = 2.182483548966554, b = 4
99) a = 2.1824880379336804, b = 4
100) a = 2.1824918360984413, b = 4
101) a = 2.182495049763575, b = 4
102) a = 2.1824977688748013, b = 4
103) a = 2.1825000695373777, b = 4
104) a = 2.1825020161455733, b = 4
105) a = 2.182503663184573, b = 4
106) a = 2.1825050567551805, b = 4
107) a = 2.18250623586395, b = 4
108) a = 2.1825072335148277, b = 4
109) a = 2.182508077632831, b = 4
110) a = 2.1825087918456054, b = 4
111) a = 2.1825093961447233, b = 4
112) a = 2.182509907445225, b = 4
113) a = 2.182510340059059, b = 4
114) a = 2.182510706095668, b = 4
115) a = 2.182511015800934, b = 4
116) a = 2.182511277843964, b = 4
117) a = 2.18251149955974, b = 4
118) a = 2.182511687154433, b = 4
119) a = 2.1825118458791173, b = 4
120) a = 2.1825119801767547, b = 4
121) a = 2.182512093806576, b = 4
```

```
122) a = 2.1825121899492146, b = 4
123) a = 2.1825122712959204, b = 4
124) a = 2.182512340123708, b = 4
125) a = 2.1825123983591843, b = 4
126) a = 2.182512447632462, b = 4
127) a = 2.1825124893227863, b = 4
128) a = 2.182512524597141, b = 4
129) a = 2.182512554442917, b = 4
130) a = 2.1825125796955507, b = 4
131) a = 2.1825126010619074, b = 4
Корінь рівняння з проміжку [0.41, 4] дорівнює:  2.182512619140069
```

In [316]:

```
d_func=lambda x: 5*x**4+4*x**3-6*x**2-18*x-3
def newton(x, y):
    i=0
    x0=(x+y)/2
    x1=x0-func(x0)/d_func(x0)
    while(abs(func(x1))>eps):
        i=i+1
        print(f'{i}) a = {x0}, b = {x1}')
        x0=(x1+x0)/2
        x1=x0-func(x0)/d_func(x0)
    return x1

print('Метод Ньютона:')
x_newton=newton(a, b)
print(f'Корінь рівняння з проміжку [{a}, {b}] дорівнює: ', x_newton)
```

```
Метод Ньютона:
1) a = 2.205, b = 2.1831526800019936
2) a = 2.194076340000997, b = 2.182684919976828
3) a = 2.1883806299889126, b = 2.182557470873077
4) a = 2.185469050430995, b = 2.1825241319020803
5) a = 2.1839965911665375, b = 2.1825156008753672
6) a = 2.183256096020952, b = 2.182513442797346
7) a = 2.1828847694091493, b = 2.182512900061704
Корінь рівняння з проміжку [0.41, 4] дорівнює:  2.1825127639725306
```

In [317]:

```
res_b=func(x_bisection)
res_h=func(x_horda)
res_n=func(x_newton)
print(f'Бісекція: f({x_bisection}) = {res_b}\nХорди: f({x_horda}) = {res_h}\nНьютон: f(
{x_newton}) = {res_n}')
if min([res_b, res_h, res_n], key=abs)==res_b:
    print(f'Найбільш точне значення отримано методом бісекції.')
elif min([res_b, res_h, res_n], key=abs)==res_h:
    print(f'Найбільш точне значення отримано методом хорд.')
else:
    print(f'Найбільш точне значення отримано методом Ньютона.')
```

```
Бісекція: f(2.1825097751617433) = -0.00024773388207055547
Хорди: f(2.182512619140069) = -8.365521783737506e-06
Ньютон: f(2.1825127639725306) = 3.824600943502787e-06
Найбільш точне значення отримано методом Ньютона.
```