1주차 과제 #1

2022145079 임혜린

본 과제는 Python, VS Code를 사용하였음을 밝힙니다.

1) CSE SERVER를 사용한 기초계산

1-1. RANDOM 함수를 써서 만든 임의의 숫자 100개를 오름차순으로 정렬하세요.

코드

```
###1-1
import random
i=0
slist=[]
while i<100:
    x=random.randint(-1000, 1000)
    slist.append(x)
    i+=1
slist.sort()
print(slist)
```

결과

```
[-998, -981, -946, -906, -882, -877, -856,
-843, -806, -775, -770, -749, -713, -690, -
594, -589, -585, -535, -521, -513, -504, -4
95, -482, -463, -449, -415, -407, -361, -32
1, -317, -308, -294, -239, -210, -204, -178
, -169, -159, -128, -58, -52, -41, -7, 2, 2
0, 43, 46, 55, 123, 129, 139, 145, 150, 167
, 174, 185, 204, 206, 219, 252, 260, 271, 2
99, 333, 373, 381, 399, 416, 429, 438, 451,
 454, 460, 476, 486, 498, 516, 581, 589, 62
2, 681, 688, 692, 706, 713, 717, 726, 734,
753, 781, 802, 815, 824, 877, 929, 955, 967
, 986, 991, 999]
```

1-2. 임의의 두 4 x 4 matrix A, B를 생성하고, 두 matrix를 곱해 matrix C를 생성하세요. 그 이후에 기존의 공식을 사용하여 matrix C의 inverse matrix를 계산하세요. 그리고, C와 inverse C를 곱해 identity matrix가 나옴을 확인하세요.

코드

```
### 1-2
# matrix C 만들기
import numpy as np
A=np.array([[1,2,3,4],[2,0,1,1],[6,3,9,8],[4,2,8,5]])
B=np.array([[6,0,3,-4],[-8,0,2,5],[1,-5,0,4],[-1,-4,-6,7]])
C=A@B

#det(A)\=0, det(B)\=0 확인
print(np.linalg.det(A))
print(np.linalg.det(B))

#C-1 구하기
# AxB=C 이므로 C-1=B-1xA-1
A_1=np.linalg.inv(A)
B_1=np.linalg.inv(B)
C_1=B_1@A_1
print(C_1)
```

Numpy linalg를 사용하였다

```
#identity matrix
print(C@C_1)

#오차 판정
if np.allclose(C@C_1, np.eye(4))==True:
    print("구한 CxC-1가 indentity matrix와 유사합니다.")
```

컴퓨터 계산의 작은 오차는 오차 판정을 이용하여 판단하였다.

결과

```
0.9999999999999925
416.99999999999966
[[ 3.91366906  5.94484412 -4.26618705  2.56115108]
 [ 8.10071942 12.11990408 -8.85611511  5.34532374]
 [ 6.03597122  8.99520384 -6.8057554   4.26618705]
 [ 8.64748201 12.91366906 -9.50359712  5.79136691]]
[[ 1.00000000e+00  5.68434189e-14  3.55271368e-15  5.32907052e-14]
 [-2.48689958e-14  1.00000000e+00  1.95399252e-14 -8.88178420e-15]
 [-7.46069873e-14  1.42108547e-13  1.00000000e+00 -8.88178420e-15]
 [ 1.77635684e-14  1.98951966e-13  1.04805054e-13  1.00000000e+00]]
구한 CxC-1가 indentity matrix와 유사합니다.PS C:\Users\user\Desktop\CSE-URP>
```

2) Root-finding method

주어진 함수 $f(x) = x^5 - 9x^4 - x^3 + 17x^2 - 8x - 8$ 에 대하여 다음 문제를 풀어 제출하세요.
(Error bound = 10^{-8})

1. 이분법(bisection method)을 사용하여 주어진 3구간 [-10,-1], [-1,0] 그리고 [0,10] 에서의 $f(x)$의 해를 구하시오.

코드

```
###2
def f(x):
    return x**5-9*x**4-x**3+17*x**2-8*x-8
def ff(x):
    return 5*x**4-36*x**3-3*x**2-34*x-8

###2-1
#[-10,-1] (찾은 근: -1.3875...)
if f(-10)*f(-1)<=0:
    a=-10
    b=-1
    n11=0
while b-a>(0.1)*0.00000001:
    m=(a+b)/2
    if f(a)*f(m)<=0:
        b=m
    else:
        a=m
    n11+=1
print(f"이분법을 통해 구한 [-10,-1]의 근은 {m}입니다.")
```

```
#[-1,0] (찾은 근: -0.5104...)
if f(-1)*f(0)<=0:
    a=-1
    b=0
    n12=0
while b-a>0.00000001:
    m=(a+b)/2
    if f(a)*f(m)<=0:
        b=m
    else:
        a=m
    n12+=1
print(f"이분법을 통해 구한 [-1,0]의 근은 {m}입니다.")
```

```
#[0,10] (찾은 근: 8.9107...)
if f(0)*f(10)<=0:
    a=0
    b=10
    n13=0
while b-a>0.00000001:
    m=(a+b)/2
    if f(a)*f(m)<=0:
        b=m
    else:
        a=m
    n13+=1
print(f"이분법을 통해 구한 [0,10]의 근은 {m}입니다.")
```

결과

```
이분법을 통해 구한 [-10,-1]의 근은 -1.3875071051879786입니다.
이분법을 통해 구한 [-1,0]의 근은 -0.5104293450713158입니다.
이분법을 통해 구한 [0,10]의 근은 8.910696404054761입니다.
```

2. 뉴턴법(Newton's method)를 사용하여 주어진 3점 $x_0 = -10, -0.1, 10$ 에서 $f(x)$의 해를 구하시오.

코드

```
###2-2
def ff(x):
    return 5*x**4-36*x**3-3*x**2-34*x-8

# x0=-10 (찾은 근: -1.3875...)
x0=-10
n21=0
while abs(f(x0))>0.00000001:
    x0=x0-f(x0)/ff(x0)
    n21+=1
print(f"뉴턴법을 통해 구한 x0=-10에서의 근은{x0}입니다.")
#print(n21)
```

```
# x0=-0.1 (찾은 근: -1.3875...)
x0=-0.1
n22=0
while abs(f(x0))>0.00000001:
    x0=x0-f(x0)/ff(x0)
    n22+=1
print(f"뉴턴법을 통해 구한 x0=-0.1에서의 근은{x0}입니다.")
#print(n22)

# x0=-0.1인 경우 -10에서와 마찬가지로 -1.3875...의 근을 찾게 됨.

#x0=10 (찾은 근: 8.9107...)
x0=10
n23=0
while abs(f(x0))>0.00000001:
    x0=x0-f(x0)/ff(x0)
    n23+=1
print(f"뉴턴법을 통해 구한 x0=10에서의 근은{x0}입니다.")
#print(n23)
```

x0=-0.1의 경우 가까운 근인 -0.5104...가 아닌 -1.3875를 찾아 내는 것을 확인하였다.

코드에 허점이 있는 것 같으나 어떻게 고쳐야 할 지 알아내지 못하였다.

결과

```
뉴턴법을 통해 구한 x0=-10에서의 근은-1.387507105761322입니다.
뉴턴법을 통해 구한 x0=-0.1에서의 근은-1.387507105731511입니다.
뉴턴법을 통해 구한 x0=10에서의 근은8.91069640295999입니다.
```

3. 뉴턴법을 사용하였을 때, 초기 조건을 $x_0 = 0$ 에 대해 계산할 경우, 계산이 어떠한 양상을 보이는지 서술하시오.

코드

```
###2-3 (찾은 근: -1.3875...)
x0=0
n3=0
while abs(f(x0))>0.00000001:
    x0=x0-f(x0)/ff(x0)
    n3+=1
print(f"뉴턴법을 통해 구한 x0=0에서의 근은{x0}입니다.")
#print(n3)
# x0=-0.5104가 아닌 x0=-1.3875의 근을 찾음.
# 0이 -1.3875로 수렴하는 영역에 속함을 알 수 있음.
```

-0.5014...가 아닌 -1.3875의 근을 찾아 내었다.

X0=-0.1을 실행하였을 때와 비슷한 양상이다.

뉴턴법의 한계인 듯하다.

결과

```
뉴턴법을 통해 구한 x0=0에서의 근은-1.3875071054062653입니다.
```

4. 시컨트법(secant method)를 사용하여 주어진 초기조건에서 $f(x)$의 해를 구하시오.

(1) $x_1 = -10, x_2 = -9.9$ / (2) $x_1 = -0.1, x_2 = -0.2$ / (3) $x_1 = 10, x_2 = 9.9$

코드

```
###2-4
### x1=-10, x2=-9.9 (찾은 근: -1.3875...)
xn2=-10
xn1=-9.9
n41=0
while abs(f(xn1))>=0.00000001:
    x=xn1-f(xn1)*(xn1-xn2)/(f(xn1)-f(xn2))
    xn2=xn1
    xn1=x
    n41+=1
print(f"시컨트법을 통해 구한 x1=-10, x2=-9.9에서의 근은{xn1}입니다.")
#print(n41)

### x1=-0.1, x2=-0.2 (찾은 근: -0.5104...)
xn2=-0.1
xn1=-0.2
n42=0
while abs(f(xn1))>=0.00000001:
    x=xn1-f(xn1)*(xn1-xn2)/(f(xn1)-f(xn2))
    xn2=xn1
    xn1=x
    n42+=1
print(f"시컨트법을 통해 구한 x1=-0.1, x2=-0.2에서의 근은{xn1}입니다.")
#print(n42)
### x1=10, x2=9.9 (찾은 근: 8.9107...)
xn2=10
xn1=9.9
n43=0
while abs(f(xn1))>=0.00000001:
    x=xn1-f(xn1)*(xn1-xn2)/(f(xn1)-f(xn2))
    xn2=xn1
    xn1=x
    n43+=1
print(f"시컨트법을 통해 구한 x1=10, x2=9.9에서의 근은{xn1}입니다.")
#print(n43)
```

결과

```
시컨트법을 통해 구한 x1=-10, x2=-9.9에서의 근은-1.387507105582635입니다.
시컨트법을 통해 구한 x1=-0.1, x2=-0.2에서의 근은-0.5104293428171596입니다.
시컨트법을 통해 구한 x1=10, x2=9.9에서의 근은8.910696402960 3입니다.
```

5. 이분법, 뉴턴법, 시컨트법에 대한 계산을 진행하고, 몇 번의 반복계산 후에 해를 구하였는지 서술하시오.

코드

```
###2-5
print(f"""이분법을 이용한 경우 각각 {n11}, {n12}, {n13}번
뉴턴법을 이용한 경우 각각 {n21}, {n22}, {n23}번
시컨트법을 이용한 경우 각각 {n41}, {n42}, {n43}번
의 반복 계산 후 해를 구하였습니다.""")
```

결과

```
이분법을 이용한 경우 각각 34, 27, 30번
뉴턴법을 이용한 경우 각각 56, 48, 13번
시컨트법을 이용한 경우 각각 18, 5, 7번
의 반복 계산 후 해를 구하였습니다.
```