실습 보고서

[실습번호: 10]

[실습제목: 5 장~7 장 관련 프로그램 작성하기]

과 목 명	선형대수
교 수 명	이 선 우
학 번	20237107
작 성 자	하 태 영
제 출 일	2025.11.26

한림대학교

가. 5장

(1) 행렬식 계산

```python
import numpy as np

def getMinorMatrix(A, i, j): # 행렬 A의 i행과 j열을 제거하고 만든 행렬 생성
  n = len(A)
  M = np.zeros((n-1, n-1))
  for a in range(0, n-1):
    k = a if (a < i) else a+1
    for b in range(0, n-1):
      l = b if(b < j) else b+1
      M[a, b] = A[k, l]
  return M

def determinant(M):  # 행렬식 계산
  if len(M) == 2:    # 2x2 행렬의 행렬식
    return M[0,0]*M[1,1]-M[0,1]*M[1,0]

  detVal = 0
  for c in range(len(M)):
    detVal += ((-1)**c)*M[0,c]*determinant(getMinorMatrix(M,0,c))
  return detVal

A = np.array([[-4, 0, 2, -1, 0], [1, 3, -3, -1, 4], [2, 0, 1, 3, 0], [-2, 1, -3, -1, 5], [1, -5, 1, 0, 5]])
print("A = ")
print(A)
print("det(A) = ", determinant(A))

print()
# 7(a)
B = np.array([[1, 5, -6], [-1, -4, 4], [-2, -7, 9]])
print("7(a) = ")
print(B)
print("det(7(a)) = ", determinant(B))

print()
# 7(b)
C = np.array([[1, 5, -6], [-1, -4, 4], [-2, -7, 9]])
print("7(b) = ")
print(C)
print("det(7(b)) = ", determinant(C))

print()
# 8(a)
D = np.array([[1, 3, 0, 2], [-2, -5, 7, 4], [3, 5, 2, 1], [1, -1, 2, -3]])
print("8(a) = ")
print(D)
print("det(8(a)) = ", determinant(D))

print()
# 8(b)
F = np.array([[1, 3, 3, -4], [0, 1, 2, -5], [2, 5, 4, -3], [-3, -7, -5, 2]])
print("8(b) = ")
print(F)
print("det(8(b)) = ", determinant(F))

print()
# 9
G = np.array([[1, 3, -1, 0, -2], [0, 2, -4, -1, -6], [-2, -6, 2, 2, 9], [2, 7, -3, 8, -7], [3, 5, 5, 2, 7]])
print("9 = ")
print(F)
print("det(9)) = ", determinant(G))
```

```
A =
[[-4  0  2 -1  0]
 [ 1  3 -3 -1  4]
 [ 2  0  1  3  0]
 [-2  1 -3 -1  5]
 [ 1 -5  1  0  5]]
det(A) =  -997.0

7(a) =
[[ 1  5 -6]
 [-1 -4  4]
 [-2 -7  9]]
det(7(a)) =  3.0

7(b) =
[[ 1  5 -6]
 [-1 -4  4]
 [-2 -7  9]]
det(7(b)) =  3.0

8(a) =
[[ 1  3  0  2]
 [-2 -5  7  4]
 [ 3  5  2  1]
 [ 1 -1  2 -3]]
det(8(a)) =  0.0

8(b) =
[[ 1  3  3 -4]
 [ 0  1  2 -5]
 [ 2  5  4 -3]
 [-3 -7 -5  2]]
det(8(b)) =  0.0

9 =
[[ 1  3  3 -4]
 [ 0  1  2 -5]
 [ 2  5  4 -3]
 [-3 -7 -5  2]]
det(9)) =  -4.0
```

(2) 역행렬 계산

```python
import numpy as np

def cofactor(A, i, j):  # 여인수 계산
  (n, m) = A.shape
  M = np.zeros((n-1, m-1))
  for a in range(0, n-1):
    k = a if (a < i) else a+1
    for b in range(0, m-1):
      l = b if (b < j) else b+1
      M[a, b] = A[k, l]
  return (-1)**(i+j)*np.linalg.det(M)

def getAdjointMatrix(A):
  (n, m) = A.shape
  if n != m:
      print("Error: 정방 행렬이 아닙니다.")
      return None

  adjA = np.zeros((n, m))

  # 수반 행렬 adjA[j, i] = 여인수 C[i, j]
  for i in range(0, n):         # i는 행 인덱스 (여인수 행렬의 행)
    for j in range(0, m):       # j는 열 인덱스 (여인수 행렬의 열)
      # C_ij를 adjA의 (j, i) 위치에 저장 (전치 수행)
      adjA[j, i] = cofactor(A, i, j)

  return adjA

def inverseByAdjoinMatrix(A):  # 수반행렬을 이용한 A의 역행렬 계산
  detA = np.linalg.det(A)      # A의 행렬식 계산
  (n, m) = A.shape
  adjA = np.zeros((n, m))

  for i in range(0, n):         # 수반행렬 생성
    for j in range(0, m):
      adjA[j,i] = cofactor(A, i, j)
  if detA != 0.0:
    return (1./detA) * adjA
  else:
      return 0
```

```
A = np.array([[-4, 0, 2, -1, 0], [1, 3, -3, -1, 4], [2, 0, 1, 3, 0], [-2, 1, -3, -1, 5], [1, -5, 1, 0, 5]])
print("A = ")
print(A)

Ainv = inverseByAdjoinMatrix(A)
print("A inverse=")
print(Ainv)

print()
B = np.array([[2, 1, 3], [-1, -2, 0], [2, -2, 1]])
print("25(a)")
print(B)

adjB = getAdjointMatrix(B)
print("25(a) adjoint=")
print(adjB)

print()
C = np.array([[2, -3, 1], [4, 2, 2], [1, 0, -2]])
print("25(b)")
print(C)

adjC = getAdjointMatrix(C)
print("25(b) adjoint=")
print(adjC)

print()
D = np.array([[2, 5, 5], [-1, -1, 0], [2, 4, 3]])
print("27(a) = ")
print(D)

Dinv = inverseByAdjoinMatrix(D)
print("27(a) inverse=")
print(Dinv)

print()
F = np.array([[1, 0, 1], [0, 1, 1], [2, 0, 1]])
print("27(b) = ")
print(D)

Finv = inverseByAdjoinMatrix(F)
print("27(b) inverse=")
print(Finv)

print()
G = np.array([[2, 1, 2], [3, 2, 2], [1, 2, 3]])
print("27(c) = ")
print(D)

Ginv = inverseByAdjoinMatrix(G)
print("27(c) inverse=")
print(Ginv)
```

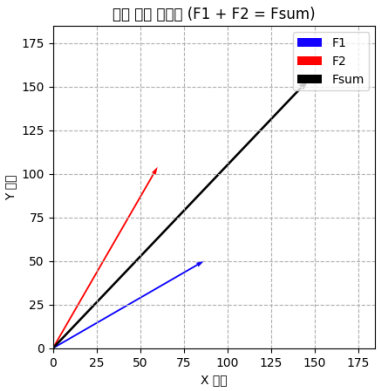
```
A =
[[-4  0  2 -1  0]
 [ 1  3 -3 -1  4]
 [ 2  0  1  3  0]
 [-2  1 -3 -1  5]
 [ 1 -5  1  0  5]]
A inverse=
[[-0.07321966  0.2106319  -0.03610832 -0.24573721  0.0772317 ]
 [ 0.16950853  0.26579739  0.09729188 -0.14343029 -0.06920762]
 [ 0.32397192  0.30090271  0.09127382 -0.35105316  0.11033099]
 [-0.05917753 -0.24072217  0.32698094  0.28084253 -0.08826479]
 [ 0.11935807  0.16349047  0.08625878 -0.02407222  0.09327984]]

25(a)
[[ 2  1  3]
 [-1 -2  0]
 [ 2 -2  1]]
25(a) adjoint=
[[-2. -7.  6.]
 [ 1. -4. -3.]
 [ 6.  6. -3.]]

25(b)
[[ 2 -3  1]
 [ 4  2  2]
 [ 1  0 -2]]
25(b) adjoint=
[[-4. -6. -8.]
 [10. -5. -0.]
 [-2. -3. 16.]]

27(a) =
[[ 2  5  5]
 [-1 -1  0]
 [ 2  4  3]]
27(a) inverse=
[[ 3. -5. -5.]
 [-3.  4.  5.]
 [ 2. -2. -3.]]

27(b) =
[[ 2  5  5]
 [-1 -1  0]
 [ 2  4  3]]
27(b) inverse=
[[-1.  0.  1.]
 [-2.  1.  1.]
 [ 2.  0. -1.]]

27(c) =
[[ 2  5  5]
 [-1 -1  0]
 [ 2  4  3]]
27(c) inverse=
[[ 0.4  0.2 -0.4]
 [-1.4  0.8  0.4]
 [ 0.8 -0.6  0.2]]
```

나. 6장

(1) 실습#1 : 벡터 연산

```python
import numpy as np
import matplotlib.pyplot as plt

def getVector(mag, deg): # 주어진 크기와 방향에 대응하는 벡터 생성
  # 각도를 라디안으로 변환 (deg * np.pi / 180)
  rad = np.deg2rad(deg)
  vec = np.zeros(2)
  vec[0] = mag * np.cos(rad) # x 성분
  vec[1] = mag * np.sin(rad) # y 성분
  return vec

def getMagDeg(vec):  # 벡터의 크기와 방향 계산
  # 1. 크기 (Magnitude) 계산
  # mag = np.sqrt(vec[0]**2 + vec[1]**2) # 원래 코드의 방식
  mag = np.linalg.norm(vec) # numpy의 표준 함수를 사용 (더 효율적)

  # 2. 방향 (Direction, 각도) 계산: atan2를 사용하여 사분면 오류 해결
  # atan2(y, x)를 사용하여 라디안 값 계산
  rad = np.arctan2(vec[1], vec[0])
  # 라디안을 도로 변환하여 0도에서 360도 범위로 조정
  deg = np.rad2deg(rad)
  if deg < 0:
    deg += 360 # 음수 각도 (3, 4사분면)를 양수로 변환

  return mag, deg

# 시각화를 위한 함수
def plot_vectors(F1, F2, Fsum):
    fig, ax = plt.subplots()

    # 3개의 벡터를 원점(0,0)에서 그립니다.
    # F1 (파란색)
    ax.quiver(0, 0, F1[0], F1[1], angles='xy', scale_units='xy', scale=1, color='b', width=0.005, label='F1')
    # F2 (주황색)
    ax.quiver(0, 0, F2[0], F2[1], angles='xy', scale_units='xy', scale=1, color='r', width=0.005, label='F2')
    # Fsum 합 벡터 (검정색)
    ax.quiver(0, 0, Fsum[0], Fsum[1], angles='xy', scale_units='xy', scale=1, color='k', width=0.007, label='Fsum')

    # 축 설정
    max_val = max(abs(Fsum[0]), abs(Fsum[1])) * 1.2
    ax.set_xlim(0, max_val)
    ax.set_ylim(0, max_val)
    ax.axhline(0, color='gray', linewidth=0.5)
    ax.axvline(0, color='gray', linewidth=0.5)
    ax.set_xlabel('X 성분')
    ax.set_ylabel('Y 성분')
    ax.set_title('벡터 합의 시각화 (F1 + F2 = Fsum)')
    ax.grid(True, linestyle='--')
    ax.legend()
    ax.set_aspect('equal', adjustable='box')
    plt.show()

# 주어진 벡터 정의
F1 = getVector(100, 30)  # 크기 100N, 방향 30도인 힘
F2 = getVector(120, 60)  # 크기 120N, 방향 60도인 힘

# 합 벡터 계산
Fsum = F1 + F2

# 합 벡터의 크기와 방향 계산
magn, angle = getMagDeg(Fsum)

print(f"F1 벡터: {F1}")
print(f"F2 벡터: {F2}")
print(f"Fsum 합 벡터: {Fsum}")
print("-" * 30)
print(f"**결합한 힘의 크기 (Magn) : {magn:.4f} N**")
print(f"**결합한 힘의 방향 (Angle) : {angle:.4f} 도**")

# 함수 호출
plot_vectors(F1, F2, Fsum)
```

```
F1 벡터: [86.60254038 50.        ]
F2 벡터: [ 60.         103.92304845]
Fsum 합 벡터: [146.60254038 153.92304845]
------------------------------
**결합한 힘의 크기 (Magn) : 212.5667 N**
**결합한 힘의 방향 (Angle) : 46.3954 도**
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 49457 (\N{HANGUL SYLLABLE SEONG}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 48516 (\N{HANGUL SYLLABLE BUN}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 48289 (\N{HANGUL SYLLABLE BEG}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 53552 (\N{HANGUL SYLLABLE TEO}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 54633 (\N{HANGUL SYLLABLE HAB}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 51032 (\N{HANGUL SYLLABLE YI}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 49884 (\N{HANGUL SYLLABLE SI}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 44033 (\N{HANGUL SYLLABLE GAG}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 54868 (\N{HANGUL SYLLABLE HWA}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



(2) 실습#2: 사잇각과 정사영 계산

```python
import numpy as np

# 두 벡터의 사잇각 cos(θ)와 각도(θ)를 계산하는 함수 (책 코드 오류 수정 및 cos(θ) 추가)
def angle2vectors(v, w):
  vnorm = np.linalg.norm(v)
  wnorm = np.linalg.norm(w)
  vwdot = np.dot(v.T, w)

  # 1. cos(theta) 계산: 문제 23번의 요구사항
  cos_theta = vwdot[0][0] / (vnorm * wnorm)

  # 2. 사잇각 계산 (arccos 사용): 책의 arctan 오류 수정
  angle_rad = np.arccos(np.clip(cos_theta, -1.0, 1.0)) # 클리핑 적용
  angle_deg = np.rad2deg(angle_rad)

  # 문제 요구사항에 따라 cos_theta와 각도를 모두 출력할 수 있도록 반환
  return angle_deg, cos_theta

# 정사영 계산 함수 (책 코드와 동일)
def orthProj(u, x):
  xu_dot = np.dot(x.T, u)
  uu_dot = np.dot(u.T, u)
  proj_ux = (xu_dot/uu_dot)*u
  return proj_ux

print("## 23. 벡터 u와 v의 사잇각 cos θ 구하기")
print("-" * 40)

# 23. (a)
u_23a = np.array([[2], [4], [1]])
v_23a = np.array([[1], [-1], [3]])
angle_23a, cos_23a = angle2vectors(u_23a, v_23a)
print(f"23(a) u = {u_23a.flatten()}, v = {v_23a.flatten()}")
print(f"   -> cos(θ) = {cos_23a:.4f} \t (θ = {angle_23a:.4f} 도)")

# 23. (b)
u_23b = np.array([[3], [1], [-1]])
v_23b = np.array([[2], [1], [-3]])
angle_23b, cos_23b = angle2vectors(u_23b, v_23b)
print(f"23(b) u = {u_23b.flatten()}, v = {v_23b.flatten()}")
print(f"   -> cos(θ) = {cos_23b:.4f} \t (θ = {angle_23b:.4f} 도)")

print("\n## 24. 벡터 u의 벡터 v 위로의 정사영 구하기")
print("-" * 40)

# 24. (a)
u_24a = np.array([[2], [-1]])
v_24a = np.array([[1], [3]])
proj_24a = orthProj(v_24a, u_24a) # proj_v u 계산 (v 위로의 u 정사영)
print(f"24(a) u = {u_24a.flatten()}, v = {v_24a.flatten()}")
print(f"   -> proj_v u:\n{proj_24a.round(4)}")

# 24. (b)
u_24b = np.array([[2], [-2], [4]])
v_24b = np.array([[-1], [1], [2]])
proj_24b = orthProj(v_24b, u_24b)
print(f"24(b) u = {u_24b.flatten()}, v = {v_24b.flatten()}")
print(f"   -> proj_v u:\n{proj_24b.round(4)}")

# 24. (c)
u_24c = np.array([[1], [2], [1], [3]])
v_24c = np.array([[1], [-3], [3], [2]])
proj_24c = orthProj(v_24c, u_24c)
print(f"24(c) u = {u_24c.flatten()}, v = {v_24c.flatten()}")
print(f"   -> proj_v u:\n{proj_24c.round(4)}")

# 24. (d)
u_24d = np.array([[2], [-2], [4]])
v_24d = np.array([[-1], [1], [1]])
proj_24d = orthProj(v_24d, u_24d)
print(f"24(d) u = {u_24d.flatten()}, v = {v_24d.flatten()}")
print(f"   -> proj_v u:\n{proj_24d.round(4)}")
```

```
## 23. 벡터 u와 v의 사잇각 cos θ 구하기
----------------------------------------
23(a) u = [2 4 1], v = [ 1 -1  3]
   -> cos(θ) = 0.0658    (θ = 86.2275 도)
23(b) u = [ 3  1 -1], v = [ 2  1 -3]
   -> cos(θ) = 0.8058    (θ = 36.3102 도)

## 24. 벡터 u의 벡터 v 위로의 정사영 구하기
----------------------------------------
24(a) u = [ 2 -1], v = [1 3]
   -> proj_v u:
[[-0.1]
 [-0.3]]
24(b) u = [ 2 -2  4], v = [-1  1  2]
   -> proj_v u:
[[-0.6667]
 [ 0.6667]
 [ 1.3333]]
24(c) u = [1 2 1 3], v = [ 1 -3  3  2]
   -> proj_v u:
[[ 0.1739]
 [-0.5217]
 [ 0.5217]
 [ 0.3478]]
24(d) u = [ 2 -2  4], v = [-1  1  1]
   -> proj_v u:
[[-0.]
 [ 0.]
 [ 0.]]
```

(다)7장

(1) rank 계산

```
import numpy as np

# 행렬 A를 출력하는 함수
def pprint(msg, A):
  print("---", msg, "---")
  (n, m) = A.shape
  for i in range(0, n):
    line = ""
    for j in range(0, m):
      line += "{0:.2f}".format(A[i, j]) + "\t"
    print(line)
  print("")

A = np.eye(4)
print()
pprint("A", A)
print("rank(A) =", np.linalg.matrix_rank(A))  # 행렬 A의 계수 계산

B = np.zeros((3,3))
print()
pprint("B", B)
print("rank(B) =", np.linalg.matrix_rank(B))  # 행렬 B의 계수 계산

C = np.array([[2, 5, -3, -4, 8],
              [4, 7, -4, -3, 9],
              [6, 9, -5, 2, 4],
              [0, -9, 6, 5, -6]]);
print()
pprint("C", C)
print("rank(C) =", np.linalg.matrix_rank(C))  # 행렬 C의 계수 계산


CT = np.transpose(C)
print()
pprint("C^T", CT)
print("rank(C^T) =", np.linalg.matrix_rank(CT))  # 행렬 C^T의 계수 계산

D = np.array([[1, 2, 4, 4],
              [3, 4, 8, 0]]);

print()
pprint("29(a)", D)
print("29(a) rank =", np.linalg.matrix_rank(D))


F = np.array([[1, 2, 3],
              [2, 3, 5],
              [3, 4, 7],
              [4, 5, 9]]);

print()
pprint("29(b)", F)
print("29(b) rank =", np.linalg.matrix_rank(D))

G = np.array([[1, 2, 1, 5],
              [2, 4, -3, 0],
              [-3, 1, 2, -1],
              [1, 2, -1, 1]]);
print()
pprint("30(a)", G)
print("30(a) rank =", np.linalg.matrix_rank(G))
print("30(a) nullity =", G.shape[1] - np.linalg.matrix_rank(G))

H = np.array([[1, -2, 1],
              [1, -1, 3],
              [1, 1, 7]]);
print()
pprint("30(b)", H)
print("30(b) rank =", np.linalg.matrix_rank(H))
print("30(b) nullity =", H.shape[1] - np.linalg.matrix_rank(H))

J = np.array([[2, 5, -3, -4, 8],
              [4, 7, -4, -3, 9],
              [6, 9, -5, 2, 4],
              [0, -9, 6, 5, -6]]);
print()
pprint("35", J)
print("35 rank =", np.linalg.matrix_rank(J))
print("35 nullity =", J.shape[1] - np.linalg.matrix_rank(J))
```

```
--- A ---
1.00    0.00    0.00    0.00
0.00    1.00    0.00    0.00
0.00    0.00    1.00    0.00
0.00    0.00    0.00    1.00

rank(A) = 4

--- B ---
0.00    0.00    0.00
0.00    0.00    0.00
0.00    0.00    0.00

rank(B) = 0

--- C ---
2.00    5.00    -3.00   -4.00   8.00
4.00    7.00    -4.00   -3.00   9.00
6.00    9.00    -5.00   2.00    4.00
0.00    -9.00   6.00    5.00    -6.00

rank(C) = 3

--- C^T ---
2.00    4.00    6.00    0.00
5.00    7.00    9.00    -9.00
-3.00   -4.00   -5.00   6.00
-4.00   -3.00   2.00    5.00
8.00    9.00    4.00    -6.00

rank(C^T) = 3
```

```
--- 29(a) ---
1.00    2.00    4.00    4.00
3.00    4.00    8.00    0.00

29(a) rank = 2

--- 29(b) ---
1.00    2.00    3.00
2.00    3.00    5.00
3.00    4.00    7.00
4.00    5.00    9.00

29(b) rank = 2

--- 30(a) ---
1.00    2.00    1.00    5.00
2.00    4.00    -3.00   0.00
-3.00   1.00    2.00    -1.00
1.00    2.00    -1.00   1.00

30(a) rank = 3
30(a) nullity = 1

--- 30(b) ---
1.00    -2.00   1.00
1.00    -1.00   3.00
1.00    1.00    7.00

30(b) rank = 2
30(b) nullity = 1

--- 35 ---
2.00    5.00    -3.00   -4.00   8.00
4.00    7.00    -4.00   -3.00   9.00
6.00    9.00    -5.00   2.00    4.00
0.00    -9.00   6.00    5.00    -6.00

35 rank = 3
35 nullity = 2
```