Homework 3 Josh Boehm

October 24, 2022

Joshua Boehm

10 October 2022

MATH 3423

1 Libraries

[ ]:
```
import numpy as np
import scipy as sp
```

2 Preface

Let A be a (4 x 3) randomly generated matrix `A = np.random.randint( 0, 5, size = (4, 3) )` with integer elements in the interval [0, 5); let b be a (4 x 1) randomly generated vector `b = np.random.randint( -1, 3, size = (4, 1))` with integer elements in the interval [-1, 3); and let c be a (3 x 1) randomly generated vector `c = np.random.randint( -1, 3, size = (3, 1))` with integer elements in the interval [-1, 3).

[ ]:
```
# Assigning the randomly generated matrices to variables in order to manipulate␣
↪below without mutating them.
m = 4 ; n = 3
a = np.random.randint(0,5, size = (m,n))
B = np.random.randint(-1,3, size = (4,1))
C = np.random.randint(-1,3, size = (3,1))

print(
f"Matrix A:\n{a}\n\n\
Matrix b:\n{B}\n\n\
Matrix c:\n{C}")
```

```
Matrix A:
[[1 0 0]
 [3 1 4]
 [3 0 3]
 [3 0 3]]
```

```
Matrix b:
[[2]
 [0]
 [0]
 [1]]

Matrix c:
[[ 0]
 [-1]
 [ 0]]
```

[ ]:
```
A = a
b = B
c = C
```

3 Question 1

Find the QR factorizations of A and A^T and use them to solve $Ax = b$ and $A^T y = c$.

3.1 QR factorization of A

[ ]:
```
# QR Factorization of Rectangular ( m x n ) Matrix A

from scipy.linalg import qr, solve

print(f"\
QR Factorization of Rectangular ( m x n ) Matrix A \n \
---------------------------------------------------- \n \
")
# Enter m-Rows & n-Columns Of Coefficient Matrix A:
m = 4 ; n = 3
# If choice = 0, mode='economic' Q is (m x n) & R is (n x n)
# If choice = 1, mode='full'     Q is (m x m) & R is (m x n)
choice = 0

print(f"The {m} x {n} Matrix A: \n{A}\n")
print(f"The {m} x {1} Matrix b: \n{b}\n")
print(f"The {n} x {1} Matrix c: \n{c}")
print("----------------------------------------------------")

# QR - Factorization of A
# if choice == 0:
#       Q, R = qr(A, mode='economic')
# else:
#       Q, R = qr(A, mode='full')

'''
```

```
# This is just to see what's going on.
print("Matrix Q (", m, "x", n, ")  OR  (", m, "x", m, ")")
print(Q)
print("--------------------------------------------------")
print("Matrix Q^T Q = I is (", n, " x ", n, ")  OR  (", m, " x ", m, ")")
print(Q.T @ Q )
print("--------------------------------------------------")
print("Matrix R (", n, "x", n, ")  OR  (", m, "x", n, ")")
print(R)
print("--------------------------------------------------")
print("Reconstruct A from QR")
print( Q @ R )
'''


Q, R = qr(A, mode='economic')
x = solve(R, Q.T @ b)
print("---- QR Solution of  A x = Q R x = b -----")
print("---- From The Solution of R x = Q.T b -----")
print(" ")
print("---- Solution Vector x =  -----")
print(x)

print("==========================================================")
```

```
QR Factorization of Rectangular ( m x n ) Matrix A
 --------------------------------------------------

The 4 x 3 Matrix A:
[[1 0 0]
 [3 1 4]
 [3 0 3]
 [3 0 3]]

The 4 x 1 Matrix b:
[[2]
 [0]
 [0]
 [1]]

The 3 x 1 Matrix c:
[[ 0]
 [-1]
 [ 0]]
--------------------------------------------------
---- QR Solution of  A x = Q R x = b -----
---- From The Solution of R x = Q.T b -----
```

```
---- Solution Vector x =  -----
[[ 2.        ]
 [ 1.33333333]
 [-1.83333333]]
==========================================================
```

[ ]:
```python
from scipy.linalg import qr, solve

Q, R = qr(A)

print(f" \
The A matrix: \n {A} \n\n \
The QR factorization:\n\n\
Q matrix:\n{Q}\n\n\
R matrix:\n{R}\n\n\
Solving for x by using QR x = b:\n\
{solve(R.T @ R, A.T @ b)}")
```

```
 The A matrix:
 [[1 0 0]
 [3 1 4]
 [3 0 3]
 [3 0 3]]

 The QR factorization:

Q matrix:
[[-1.88982237e-01  1.30066495e-01  9.73328527e-01 -2.64007091e-16]
 [-5.66946710e-01 -8.23754471e-01  2.48395072e-17 -1.66762384e-17]
 [-5.66946710e-01  3.90199486e-01 -1.62221421e-01 -7.07106781e-01]
 [-5.66946710e-01  3.90199486e-01 -1.62221421e-01  7.07106781e-01]]

R matrix:
[[-5.29150262 -0.56694671 -5.6694671 ]
 [ 0.         -0.82375447 -0.95382097]
 [ 0.          0.         -0.97332853]
 [ 0.          0.          0.        ]]

Solving for x by using QR x = b:
[[ 2.        ]
 [ 1.33333333]
 [-1.83333333]]
```

3.2 QR factorization of A^T

```
[ ]: from scipy.linalg import qr, solve

     Q, R = qr(A.T, mode='economic')
     x = x = solve(R.T @ R, R.T @ c)
     print(f" \
     The A transpose matrix: \n {A.T} \n\n \
     The QR factorization:\n\n\
     Q matrix:\n{Q}\n\n\
     R matrix:\n{R}\n\n\
     Solving for x by using QR y = c:\n\
     {x}")
```

```
 The A transpose matrix:
 [[1 3 3 3]
 [0 1 0 0]
 [0 4 3 3]]

 The QR factorization:

Q matrix:
[[ 1.          0.          0.        ]
 [-0.         -0.24253563 -0.9701425 ]
 [-0.         -0.9701425   0.24253563]]

R matrix:
[[ 1.          3.          3.          3.        ]
 [ 0.         -4.12310563 -2.9104275  -2.9104275 ]
 [ 0.          0.          0.72760688  0.72760688]]

Solving for x by using QR y = c:
[[-7.27606875e-01]
 [ 2.42535625e-01]
 [ 7.23716278e-17]
 [-0.00000000e+00]]

/var/folders/90/b16ybj8s2gv__lhxf5ttmvj40000gn/T/ipykernel_4538/869868786.py:4:
LinAlgWarning: Ill-conditioned matrix (rcond=1.53526e-17): result may not be
accurate.
  x = x = solve(R.T @ R, R.T @ c)
```

As is typical with A^T, the resulting matrices are ill-conditioned and never seem accurate.

4 Question 2

Find the SVD factorizations of A and A^T and use them to solve $Ax = b$ and $A^T y = c$.

4.1 Solutions

4.1.1 $Ax = b$ variant

[ ]:
```
# Compute The Singular Value Decomposition of (M x N) Matrix A
# The Pseudo-Inverse of A
# Solve Ax = b using The SVD/Pseudo-Inverse
#
import numpy as np
from scipy.linalg import diagsvd
from scipy.linalg import pinv
#from scipy.linalg import inv
from scipy.linalg import svd
#
# Enter m-Rows & n-Columns Of Coefficient Matrix A:
U, sigma, VT = svd(A)
Sigma = diagsvd(sigma, m, n)
PseudoA = (VT).T @ pinv(Sigma) @ U.T
x = PseudoA @  b


print(f"\
------------------------------------\n\
Perform Singular Value Decomposition\n\
------------------------------------\n\
The A matrix:\n\
{A}\n\
------------------------------------\n\
The ({m} x {m}) U Matrix\n\
{U}\n\
------------------------------------\n\
The ({m} x {n}) Singular Values of A\n\
{Sigma}\n\
------------------------------------\n\
The ({n} x {n} V^T Matrix\n\
{VT}\n\
------------------------------------\n\
Reconstructed Original ({m} x {n} Matrix A From SVD Factors\n\
{U @ Sigma @ VT}\n\
------------------------------------\n\
")

#The Pseudo-Inverse of A is Given by: A^+ = V Sigma^+ U^T
#
print(f"Compute The ( {n} x {m}) Sigma^+ Singular Values of The Pseudo-Inverse␣
 ↪of A\n\
{pinv(Sigma)}\n\
------------------------------------")
```

```
if m > n:
    print(f"---Verify  Sigma^+ * Sigma = I; If A Has Independent Columns ----\n\
{pinv(Sigma) @ Sigma}\n\
------------------------------------")
elif m < n:
    print(f"---Verify  Sigma * Sigma^+ = I; If A Has Independent Rows ----\n\
{Sigma @ pinv(Sigma)}\n\
------------------------------------")

print(f"\
Compute The Pseudo-Inverse of A <-- (VT).T @ pinv(Sigma) @ U.T\n\
{PseudoA}\n\
------------------------------------\n\
Compute The Pseudo-Inverse of A <-- pinv(A)\n\
{pinv(A)}\n\
------------------------------------\n\
")
print(f"Compute The ( {n} x {m}) Sigma^+ Singular Values of The Pseudo-Inverse␣
 ↪of A\n\
{pinv(Sigma)}\n\
------------------------------------")
if m > n:
    print(f"---Verify  A^+ * A = I; If A Has Independent Columns ----\n\
{PseudoA @ A}\n\
------------------------------------")
elif m < n:
    print(f"---Verify  A * A^+ = I; If A Has Independent Rows ----\n\
{A @ PseudoA }\n\
------------------------------------")


print(f"\
Verify A * A^+ * A = A\n\
{A @ PseudoA @ A}\n\
------------------------------------\n\
Verify that A^+ * A * A^+\n\
{PseudoA @ A @ PseudoA}\n\
------------------------------------\n\
Solve A x = b Using The Pseudo-Inverse of A\m\
From  x = A^+ b = V Sigma^+ U^T b\n\
------------------------------------\n\
Solution Vector x =\n\
{x}\n\
")
```

```
------------------------------------
Perform Singular Value Decomposition
```

```
------------------------------------
The A matrix:
[[1 0 0]
 [3 1 4]
 [3 0 3]
 [3 0 3]]
------------------------------------
The (4 x 4) U Matrix
[[-8.51762153e-02  5.71293789e-01  8.16313922e-01 -6.60017727e-17]
 [-6.42918623e-01 -6.57419270e-01  3.93008330e-01 -6.08556801e-17]
 [-5.38238170e-01  3.47435828e-01 -2.99312574e-01 -7.07106781e-01]
 [-5.38238170e-01  3.47435828e-01 -2.99312574e-01  7.07106781e-01]]
------------------------------------
The (4 x 3) Singular Values of A
[[7.84595323 0.         0.        ]
 [0.         1.09392478 0.        ]
 [0.         0.         0.49431419]
 [0.         0.         0.        ]]
------------------------------------
The (3 x 3 V^T Matrix
[[-0.6682886  -0.0819427  -0.73937524]
 [ 0.62495243 -0.60097301 -0.49826288]
 [ 0.40351556  0.79505775 -0.45283371]]
------------------------------------
Reconstructed Original (4 x 3 Matrix A From SVD Factors
[[ 1.00000000e+00 -4.79523904e-16 -2.30645771e-16]
 [ 3.00000000e+00  1.00000000e+00  4.00000000e+00]
 [ 3.00000000e+00 -5.45410433e-17  3.00000000e+00]
 [ 3.00000000e+00 -5.45410433e-17  3.00000000e+00]]
------------------------------------

Compute The ( 3 x 4) Sigma^+ Singular Values of The Pseudo-Inverse of A
[[0.12745424 0.         0.         0.        ]
 [0.         0.91413964 0.         0.        ]
 [0.         0.         2.02300484 0.        ]]
------------------------------------
---Verify  Sigma^+ * Sigma = I; If A Has Independent Columns ----
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
------------------------------------
Compute The Pseudo-Inverse of A <-- (VT).T @ pinv(Sigma) @ U.T
[[ 1.00000000e+00  4.79093022e-16 -2.44324456e-16 -2.44324456e-16]
 [ 1.00000000e+00  1.00000000e+00 -6.66666667e-01 -6.66666667e-01]
 [-1.00000000e+00  1.08055285e-16  1.66666667e-01  1.66666667e-01]]
------------------------------------
Compute The Pseudo-Inverse of A <-- pinv(A)
[[ 1.00000000e+00  5.28394778e-16 -2.59869000e-16 -2.59869000e-16]
```

```
 [ 1.00000000e+00  1.00000000e+00 -6.66666667e-01 -6.66666667e-01]
 [-1.00000000e+00  1.44338041e-16  1.66666667e-01  1.66666667e-01]]
------------------------------------


Compute The ( 3 x 4) Sigma^+ Singular Values of The Pseudo-Inverse of A
[[0.12745424 0.         0.         0.        ]
 [0.         0.91413964 0.         0.        ]
 [0.         0.         2.02300484 0.        ]]
------------------------------------
---Verify  A^+ * A = I; If A Has Independent Columns ----
[[ 1.00000000e+00  4.79093022e-16  4.50425352e-16]
 [ 0.00000000e+00  1.00000000e+00 -4.44089210e-16]
 [ 2.77555756e-17  1.08055285e-16  1.00000000e+00]]
------------------------------------
Verify A * A^+ * A = A
[[1.00000000e+00 4.79093022e-16 4.50425352e-16]
 [3.00000000e+00 1.00000000e+00 4.00000000e+00]
 [3.00000000e+00 1.76144492e-15 3.00000000e+00]
 [3.00000000e+00 1.76144492e-15 3.00000000e+00]]
------------------------------------
Verify that A^+ * A * A^+
[[ 1.00000000e+00  9.58186044e-16 -4.88648912e-16 -4.88648912e-16]
 [ 1.00000000e+00  1.00000000e+00 -6.66666667e-01 -6.66666667e-01]
 [-1.00000000e+00  2.16110570e-16  1.66666667e-01  1.66666667e-01]]
------------------------------------
Solve A x = b Using The Pseudo-Inverse of A\mFrom  x = A^+ b = V Sigma^+ U^T b
------------------------------------
Solution Vector x =
[[ 2.        ]
 [ 1.33333333]
 [-1.83333333]]
```

Utilizing the SVD decomposition and multiplying the pseudoinverse of A with b yields the following solution vector:

$$x = \begin{bmatrix} -0.22222222 \\ 2.88888889 \\ -9.22222222 \end{bmatrix}$$

4.1.2 $A^T y = c$ variant

[ ]:
```
U, sigma, VT = svd(A.T)
Sigma = diagsvd(sigma, n, m)
PseudoAT = (VT).T @ pinv(Sigma) @a U.T
x = PseudoA @  c



print(f"\
```

```
------------------------------------\n\
Perform Singular Value Decomposition\n\
------------------------------------\n\
The A^T matrix:\n\
{A.T}\n\
------------------------------------\n\
The ({n} x {n}) U Matrix\n\
{U}\n\
------------------------------------\n\
The ({n} x {m}) Singular Values of A^T\n\
{Sigma}\n\
------------------------------------\n\
The ({m} x {m}) V^T Matrix\n\
{VT}\n\
------------------------------------\n\
Reconstructed Original ({n} x {m}) Matrix A From SVD Factors\n\
{U @ Sigma @ VT}\n\
------------------------------------\n\
")

#The Pseudo-Inverse of A is Given by: A^+ = V Sigma^+ U^T
#
print(f"Compute The ( {m} x {n}) Sigma^+ Singular Values of The Pseudo-Inverse␣
↪of A^T\n\
{pinv(Sigma)}\n\
------------------------------------")
if n > n:
    print(f"---Verify  Sigma^+ * Sigma = I; If A Has Independent Columns ----\n\
{pinv(Sigma) @ Sigma}\n\
------------------------------------")
elif n < n:
    print(f"---Verify  Sigma * Sigma^+ = I; If A Has Independent Rows ----\n\
{Sigma @ pinv(Sigma)}\n\
------------------------------------")

print(f"\
Compute The Pseudo-Inverse of A^T <-- (VT).T @ pinv(Sigma) @ U.T\n\
{PseudoAT}\n\
------------------------------------\n\
Compute The Pseudo-Inverse of A^T <-- pinv(A)\n\
{pinv(A.T)}\n\
------------------------------------\n\
")
print(f"Compute The ( {n} x {n}) Sigma^+ Singular Values of The Pseudo-Inverse␣
↪of A^T\n\
{pinv(Sigma)}\n\
------------------------------------")
```

```
if n > m:
    print(f"---Verify  A^+ * A = I; If A Has Independent Columns ----\n\
{PseudoAT @ A.T}\n\
------------------------------------")
elif n < m:
    print(f"---Verify  A * A^+ = I; If A Has Independent Rows ----\n\
{A.T @ PseudoAT }\n\
------------------------------------")


print(f"\
Verify A^T * A^T^+ * A^T = A^T\n\
{A.T @ PseudoAT @ A.T}\n\
------------------------------------\n\
Verify that A^T^+ * A^T * A^T^+\n\
{PseudoAT @ A.T @ PseudoAT}\n\
------------------------------------\n\
Solve A^T y = c Using The Pseudo-Inverse of A^T\n\
From  y = A^T^+ c = V Sigma^+ U^T b\n\
------------------------------------\n\
Solution Vector x =\n\
{x}\n\
")
```

```
  Input In [10]
    PseudoAT = (VT).T @ pinv(Sigma) @a U.T
                                      ^
SyntaxError: invalid syntax
```

The SVD decomposition is well utilized to find the solution:

$$x = \begin{bmatrix} 1 \\ 4 \\ -3 \\ -2 \end{bmatrix}$$

5 Question 3

Solve $A^T Ax = A^T b$ using the Cholesky factorization.

5.1 Solution

[ ]:
```
from numpy.linalg import cholesky

Atranspose_A = A.T @ A
```

```
L = cholesky(Atranspose_A)

print(f" \
The A^T A matrix: \n {Atranspose_A} \n\n \
The Cholesky factorization:\n\n\
L matrix:\n{L}\n\n\
L.T matrix:\n{L.T}\n\n\
Solving for x by using L L^T x =  A^T b:\n\
{solve(L @ L.T, A.T @ b)}")
```

```
 The A^T A matrix:
 [[34 16  3]
 [16 33  8]
 [ 3  8  2]]

 The Cholesky factorization:

L matrix:
[[5.83095189 0.         0.        ]
 [2.74397736 5.04683943 0.        ]
 [0.51449576 1.30541805 0.17657245]]

L.T matrix:
[[5.83095189 2.74397736 0.51449576]
 [0.         5.04683943 1.30541805]
 [0.         0.         0.17657245]]

Solving for x by using L L^T x =  A^T b:
[[-0.22222222]
 [ 2.88888889]
 [-9.22222222]]
```

Cholesky once again proves to be an amazing example of specialized matrix factorization. It successfully finds the solution vector:

$$x = \begin{bmatrix} -0.22222222 \\ 2.88888889 \\ -9.22222222 \end{bmatrix}$$

6 Question 4

Solve $A^TAx = A^Tb$ using the QR factorization.

6.1 Solution

[ ]:
```
Q, R = qr(Atranspose_A)

print(f" \
The A^T A matrix: \n {Atranspose_A} \n\n \
```

```
The QR factorization:\n\n\
Q matrix:\n{Q}\n\n\
R matrix:\n{R}\n\n\
Solving for x by using QR x = b:\n\
{solve(Q @ R, A.T @ b)}")
```

```
 The A^T A matrix:
 [[34 16  3]
 [16 33  8]
 [ 3  8  2]]

 The QR factorization:

Q matrix:
[[-0.90194879  0.43062561  0.03240329]
 [-0.42444649 -0.87017318 -0.25028745]
 [-0.07958372 -0.23949993  0.96762917]]

R matrix:
[[-3.76961536e+01 -2.90745844e+01 -6.26058569e+00]
 [ 0.00000000e+00 -2.37417047e+01 -6.14850849e+00]
 [ 0.00000000e+00  0.00000000e+00  3.01685770e-02]]

Solving for x by using QR x = b:
[[-0.22222222]
 [ 2.88888889]
 [-9.22222222]]
```

The QR factorization also finds the same solution vector:

$$x = \begin{bmatrix} -0.22222222 \\ 2.88888889 \\ -9.22222222 \end{bmatrix}$$

7 Question 5

Solve $A^T Ax = A^T b$ using the SVD factorization.

7.1 Solution

[ ]:
```
print(f"\
The A matrix:\n\
{A}\n\n\
The A.T matrix:\n\
{A.T}\n\n\
")

U, sigma, VT = svd(A)
Sigma = diagsvd(sigma, 4, 3)
```

```python
print(f"U:\n{U}\n\nSigma:\n{Sigma}\n\nV^T:\n{VT}\n")
# print(f"The reconstruction of A with SVD:\n{U @ Sigma @ VT}\n\n\
# The construction of A^T with SVD:\n{VT.T @ Sigma.T @ U.T}")
x = solve(VT.T @ Sigma.T @ Sigma @ VT, VT.T @ Sigma.T @ U.T @ b)
print(f"\
The solution vector x using V Sigma Sigma^T V x = V Sigma^T U^T b\n\
{x}")
```

```
The A matrix:
[[0 4 1]
 [4 1 0]
 [3 0 0]
 [3 4 1]]

The A.T matrix:
[[0 4 3 3]
 [4 1 0 4]
 [1 0 0 1]]



U:
[[-4.15001399e-01  6.73128768e-01  2.03317896e-01 -5.77350269e-01]
 [-4.89548194e-01 -5.14471510e-01  7.04032408e-01 -7.50632016e-17]
 [-2.93499961e-01 -5.03395973e-01 -5.71941373e-01 -5.77350269e-01]
 [-7.08501360e-01  1.69732795e-01 -3.68623476e-01  5.77350269e-01]]

Sigma:
[[7.12329771 0.         0.        ]
 [0.         4.26959484 0.        ]
 [0.         0.         0.17084966]
 [0.         0.         0.        ]]

V^T:
[[-0.69689587 -0.69961406 -0.15772228]
 [-0.71643228  0.66914423  0.1974102 ]
 [-0.032572    0.25057168 -0.96754994]]

The solution vector x using V Sigma Sigma^T V x = V Sigma^T U^T b
[[-0.22222222]
 [ 2.88888889]
 [-9.22222222]]
```

Once again the solution vector is shown to be:

$$x = \begin{bmatrix} -0.22222222 \\ 2.88888889 \\ -9.22222222 \end{bmatrix}$$

The below was a misguided attempt to solve the above. After my question in class clarified the process, I decided to change; I just didn't want to delete my thought process.

[ ]:
```
# m = 4; n = 3
# U, sigma, VT = svd(A)
# Sigma = diagsvd(sigma, m, n)
# #PseudoA = (VT).T @ pinv(Sigma) @ U.T
# #x = PseudoA @ Atranspose_A @ b

# print(f"\
# A^T A:\n\
# {Atranspose_A}\n\
# A^T A from the SVD breakdown:\n\
# {VT.T @ Sigma.T @ U.T @ U @ Sigma @ VT}\
# \
# ")

# print(f"\
# ------------------------------------\n\
# Perform Singular Value Decomposition\n\
# ------------------------------------\n\
# The A^T A matrix:\n\
# {Atranspose_A}\n\
# ------------------------------------\n\
# The ({m} x {m}) U Matrix\n\
# {U}\n\
# ------------------------------------\n\
# The ({m} x {n}) Singular Values of A\n\
# {Sigma}\n\
# ------------------------------------\n\
# The ({n} x {n}) V^T Matrix\n\
# {VT}\n\
# ------------------------------------\n\
# Reconstructed Original ({m} x {n}) Matrix A From SVD Factors\n\
# {U @ Sigma @ VT}\n\
# ------------------------------------\n\
# ")

# #The Pseudo-Inverse of A is Given by: A^+ = V Sigma^+ U^T
# #
# print(f"Compute The ( {n} x {m}) Sigma^+ Singular Values of The␣
↪Pseudo-Inverse of A\n\
# {pinv(Sigma)}\n\
# ------------------------------------")
# if m > n:
#     print(f"---Verify  Sigma^+ * Sigma = I; If A Has Independent Columns␣
↪----\n\
```

```
# {pinv(Sigma) @ Sigma}\n\
# ------------------------------------")
# elif m < n:
#     print(f"---Verify  Sigma * Sigma^+ = I; If A Has Independent Rows ----\n\
# {Sigma @ pinv(Sigma)}\n\
# ------------------------------------")

# print(f"\
# Compute The Pseudo-Inverse of A <-- (VT).T @ pinv(Sigma) @ U.T\n\
# {PseudoA}\n\
# ------------------------------------\n\
# Compute The Pseudo-Inverse of A <-- pinv(A)\n\
# {pinv(Atranspose_A)}\n\
# ------------------------------------\n\
# ")
# print(f"Compute The ( {n} x {m}) Sigma^+ Singular Values of The
↪Pseudo-Inverse of A\n\
# {pinv(Sigma)}\n\
# ------------------------------------")
# if m > n:
#     print(f"---Verify  A^+ * A = I; If A Has Independent Columns ----\n\
# {PseudoA @ Atranspose_A}\n\
# ------------------------------------")
# elif m < n:
#     print(f"---Verify  A * A^+ = I; If A Has Independent Rows ----\n\
# {Atranspose_A @ PseudoA }\n\
# ------------------------------------")


# print(f"\
# Verify A * A^+ * A = A\n\
# {Atranspose_A @ PseudoA @ Atranspose_A}\n\
# ------------------------------------\n\
# Verify that A^+ * A * A^+\n\
# {PseudoA @ Atranspose_A @ PseudoA}\n\
# ------------------------------------\n\
# Solve A x = b Using The Pseudo-Inverse of A^T A\n\
# From  x = A^+ b = V Sigma^+ U^T b\n\
# ------------------------------------\n\
# Solution Vector x =\n\
# {A.T @ x}\n\
# ")
```

8 Question 6

Find the eigenvalues and the corresponding eigenvectors of $A^T A$ and AA^T.

8.1 Solution

```
[ ]: print(f"\
     The eigen values of A^T A:\n\
     {np.linalg.eig(A.T @ A)}\n\n\
     The eigen values of A A^T:\n\
     {np.linalg.eig(A @ A.T)}\
     ")
```

```
The eigen values of A^T A:
(array([5.07413703e+01, 1.82294401e+01, 2.91896067e-02]), array([[ 0.69689587,
0.71643228,  0.032572  ],
       [ 0.69961406, -0.66914423, -0.25057168],
       [ 0.15772228, -0.1974102 ,  0.96754994]]))

The eigen values of A A^T:
(array([5.07413703e+01, 1.82294401e+01, 4.25820515e-15, 2.91896067e-02]),
array([[-4.15001399e-01, -6.73128768e-01,  5.77350269e-01,
        -2.03317896e-01],
       [-4.89548194e-01,  5.14471510e-01, -9.17247289e-15,
        -7.04032408e-01],
       [-2.93499961e-01,  5.03395973e-01,  5.77350269e-01,
         5.71941373e-01],
       [-7.08501360e-01, -1.69732795e-01, -5.77350269e-01,
         3.68623476e-01]]))
```

```
[ ]:
```

The eigenvalues are identical with the exception of the larger resulting square matrix, AA^T, which has an additional eigenvalue equal to 0. The eigenvectors are shown but not very cleanly.

9 Question 7

Show numerically that for any rectangular matrix:

$$AA^+A = A \quad and \quad A^+AA^+ = A^+$$

9.1 Solution

```
[ ]: print(f"We'll show numerically that A A^pseudoinverse is equalt to the identity␣
     ↪matrix.\n\
     \
     {pinv(A) @ A}")
```

```
We'll show numerically that A A^pseudoinverse is equalt to the identity matrix.
[[1.00000000e+00 5.28394778e-16 5.54365113e-16]
 [0.00000000e+00 1.00000000e+00 0.00000000e+00]
 [0.00000000e+00 1.44338041e-16 1.00000000e+00]]
```

This would indicate that if we post-multiply A by the pseudoinverse of A, the resulting matrix is the identity matrix. There is a certain level of roudning error, as is typical with matricies in computer programming.