Homework Assignment Number Seven Solutions

For Problems 1, 2 and 3 below, find
(a) the determinant of A
(b) the reduced row echelon form of A
(c) the reduced row echelon form of the Augmented Ab matrix.
(d) the rank of A.
(e) the rank of the augmented Ab matrix.
(f) the inverse of A if it exists
(g) a solution to $\underline{\underline{A}}\,\underline{x} = \underline{b}$ if it exists.

Problem 1. (14 points)

$$\underline{\underline{A}} = \begin{bmatrix} 1 & 2 & -1 \\ 1 & 5 & 1 \\ 1 & 4 & 2 \end{bmatrix} \text{ and } \underline{b} = \begin{bmatrix} 1 \\ 2 \\ 5 \end{bmatrix}$$

Solution:

This problem has one unique solution.

In Python:

This was my script

```
import numpy as np
from sympy import Matrix

# Define A matrix
A = np.array([[1, 2, -1],
              [1, 5, 1],
              [1, 4, 2]])

# Define b vector
b = np.array([1, 2, 5])

# Compute determinant of A
detA = np.linalg.det(A)
print(f"(a) Determinant of A: {detA}")

# Compute reduced row echelon form of A using sympy (since scipy doesn't have
RREF)
A_sym = Matrix(A)
rrefA, pivot_columns = A_sym.rref()
print(f"(b) Reduced Row Echelon Form of A:\n{rrefA}")

# Compute rank of A
rankA = np.linalg.matrix_rank(A)
print(f"(d) Rank of A: {rankA}")
```

```
# Create augmented Ab1 matrix
Ab = np.hstack([A, b.reshape(-1, 1)])

# Compute reduced row echelon form of Ab
Ab_sym = Matrix(Ab)
rrefAb, _ = Ab_sym.rref()
print(f"(c) Reduced Row Echelon Form of Ab:\n{rrefAb}")

# Compute rank of Ab
rankAb = np.linalg.matrix_rank(Ab)
print(f"(e) Rank of Ab: {rankAb}")

# Compute inverse of A
invA = np.linalg.inv(A)
print(f"(f) Inverse of A:\n{invA}")

# Solve Ax = b
x = np.dot(invA, b)
print(f"(g) Solution to Ax = b: {x}")
```

This was the output:

```
run hw07_p01.py
(a) Determinant of A: 5.000000000000001
(b) Reduced Row Echelon Form of A:
Matrix([[1, 0, 0], [0, 1, 0], [0, 0, 1]])
(d) Rank of A: 3
(c) Reduced Row Echelon Form of Ab1:
Matrix([[1, 0, 0, 5], [0, 1, 0, -1], [0, 0, 1, 2]])
(e) Rank of Ab: 3
(f) Inverse of A:
[[ 1.2 -1.6  1.4]
 [-0.2  0.6 -0.4]
 [-0.2 -0.4  0.6]]
(g) Solution to Ax = b: [ 5. -1.  2.]
```

In Matlab:

This was my MatLab script

```
clear all;
A = [1 2 -1; 1 5 1; 1 4 2]
b = [1; 2; 5]
detA = det(A)
rrefA = rref(A)
rankA = rank(A)
Ab = [A,b]
rrefAb = rref(Ab)
rankAb = rank(Ab)
invA = inv(A)
x = invA*b
bcheck = A*x
bdiff = b - bcheck
```

This was the output:

```
>> hw9p01
```

```
A =
     1     2    -1
     1     5     1
     1     4     2

b =
     1
     2
     5

detA =     5

rrefA =
     1     0     0
     0     1     0
     0     0     1

rankA =     3

Ab =

     1     2    -1     1
     1     5     1     2
     1     4     2     5

rrefAb =
     1     0     0     5
     0     1     0    -1
     0     0     1     2

rankAb =     3

invA =
    1.2000   -1.6000    1.4000
   -0.2000    0.6000   -0.4000
   -0.2000   -0.4000    0.6000

x =
    5.0000
   -1.0000
    2.0000

bcheck =
    1.0000
    2.0000
    5.0000

bdiff =   1.0e-15 *
    0.8882
    0.8882
    0.8882
```

Problem 2. (14 points)

$$\underline{\underline{A}} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & -1 \\ 1 & 3 & 2 & -2 \\ 1 & 4 & 3 & -3 \end{bmatrix} \text{ and } \underline{b} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 2 \end{bmatrix}$$

Solution:

This problem has an infinite number of solutions.

In Python:

This was my script

```
import numpy as np
from sympy import Matrix

# Define A matrix
A = np.array([[1, 1, 1, 1],
              [1, 2, 1, -1],
              [1, 3, 2, -2],
              [1, 4, 3, -3]])

# Define b vector
b = np.array([1, 0, 1, 2])

# Compute determinant of A
detA = np.linalg.det(A)
print(f"(a) Determinant of A: {detA}")

# Compute reduced row echelon form of A using sympy (since scipy doesn't have
RREF)
A_sym = Matrix(A)
rrefA, pivot_columns = A_sym.rref()
print(f"(b) Reduced Row Echelon Form of A:\n{rrefA}")

# Compute rank of A
rankA = np.linalg.matrix_rank(A)
print(f"(d) Rank of A: {rankA}")

# Create augmented Ab1 matrix
Ab = np.hstack([A, b.reshape(-1, 1)])

# Compute reduced row echelon form of Ab
Ab_sym = Matrix(Ab)
rrefAb, _ = Ab_sym.rref()
print(f"(c) Reduced Row Echelon Form of Ab:\n{rrefAb}")

# Compute rank of Ab
rankAb = np.linalg.matrix_rank(Ab)
print(f"(e) Rank of Ab: {rankAb}")

# size of original A matrix
```

```
n = 4;

# get subset of rref Ab
# convert rrefAb from symbolic to numeric matrix
rrefAbnum = np.array(rrefAb.evalf(), dtype=np.float64)
#print(f"Reduced Row Echelon Form of Abnum:\n{rrefAbnum}")
Anew = rrefAbnum[0:n-1,0:n-1]
print(f"Subset of Reduced Row Echelon Form of Ab:\n{Anew}")

# pick an arbitrary value for x4
x4 = 1.0

# form new b vector fron rrefAbnum
bnew = np.array([[rrefAbnum[0,n]-rrefAbnum[0,n-1]*x4],[rrefAbnum[1,n]-
rrefAbnum[1,n-1]*x4],[rrefAbnum[2,n]-rrefAbnum[2,n-1]*x4]])
print(f"bnew:\n{bnew}")

# Compute inverse of Anew
invAnew = np.linalg.inv(Anew)
print(f"Inverse of Anew:\n{invAnew}")

# Solve Anewxnew = bnew
xnew = invAnew@bnew
print(f"Solution to Anewxnew = bnew: \n{xnew}")

# provide solution to original problem
x = np.vstack([xnew, x4])
print(f"Solution to Ax = b: \n{x}")
```

This was the output:

```
run hw07_p02.py
(a) Determinant of A: 3.7007434154171906e-16
(b) Reduced Row Echelon Form of A:
Matrix([[1, 0, 0, 2], [0, 1, 0, -2], [0, 0, 1, 1], [0, 0, 0, 0]])
(d) Rank of A: 3
(c) Reduced Row Echelon Form of Ab:
Matrix([[1, 0, 0, 2, 0], [0, 1, 0, -2, -1], [0, 0, 1, 1, 2], [0, 0, 0, 0,
0]])
(e) Rank of Ab: 3
Subset of Reduced Row Echelon Form of Ab:
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
bnew:
[[-2.]
 [ 1.]
 [ 1.]]
Inverse of Anew:
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
Solution to Anewxnew = bnew:
[[-2.]
 [ 1.]
 [ 1.]]
Solution to Ax = b:
```

```
[[-2.]
 [ 1.]
 [ 1.]
 [ 1.]]
```

In Matlab:

This was my MatLab script

```
clear all;
A = [1 1 1 1; 1 2 1 -1; 1 3 2 -2; 1 4 3 -3]
b = [1; 0; 1; 2]
detA = det(A)
rrefA = rref(A)
rankA = rank(A)
Ab = [A,b]
rrefAb = rref(Ab)
rankAb = rank(Ab)
A3 =rrefAb(1:3,1:3)
x4 = 1
b3 = rrefAb(1:3,5) - rrefAb(1:3,4)*x4
invA3 = inv(A3)
x3 = invA3*b3
x = [x3; x4]
bcheck = A*x
bdiff = b - bcheck
```

This was the output:

```
>> hw9p02

A =
     1     1     1     1
     1     2     1    -1
     1     3     2    -2
     1     4     3    -3

b =
     1
     0
     1
     2

detA =   3.3307e-16

rrefA =
     1     0     0     2
     0     1     0    -2
     0     0     1     1
     0     0     0     0

rankA =     3

Ab =
     1     1     1     1     1
     1     2     1    -1     0
```

```
     1     3     2    -2     1
     1     4     3    -3     2

rrefAb =
     1     0     0     2     0
     0     1     0    -2    -1
     0     0     1     1     2
     0     0     0     0     0

rankAb =     3

A3 =
     1     0     0
     0     1     0
     0     0     1

x4 =     1

b3 =
    -2
     1
     1

invA3 =
     1     0     0
     0     1     0
     0     0     1

x3 =
    -2
     1
     1

x =
    -2
     1
     1
     1

bcheck =
     1
     0
     1
     2

bdiff =
     0
     0
     0
     0
```

Problem 3. (14 points)

$$\underline{\underline{A}} = \begin{bmatrix} 1 & 2 & -1 \\ 1 & 5 & 1 \\ 2 & 7 & 0 \end{bmatrix} \text{ and } \underline{b} = \begin{bmatrix} 1 \\ 2 \\ 5 \end{bmatrix}$$

Solution:

This problem has no solutions.

In Python:

This was my script

```
import numpy as np
from sympy import Matrix

# Define A matrix
A = np.array([[1, 2, -1],
              [1, 5, 1],
              [2, 7, 0]])

# Define b vector
b = np.array([1, 2, 5])

# Compute determinant of A
detA = np.linalg.det(A)
print(f"(a) Determinant of A: {detA}")

# Compute reduced row echelon form of A using sympy (since scipy doesn't have
RREF)
A_sym = Matrix(A)
rrefA, pivot_columns = A_sym.rref()
print(f"(b) Reduced Row Echelon Form of A:\n{rrefA}")

# Compute rank of A
rankA = np.linalg.matrix_rank(A)
print(f"(d) Rank of A: {rankA}")

# Create augmented Ab1 matrix
Ab = np.hstack([A, b.reshape(-1, 1)])

# Compute reduced row echelon form of Ab
Ab_sym = Matrix(Ab)
rrefAb, _ = Ab_sym.rref()
print(f"(c) Reduced Row Echelon Form of Ab:\n{rrefAb}")

# Compute rank of Ab
rankAb = np.linalg.matrix_rank(Ab)
print(f"(e) Rank of Ab: {rankAb}")

# # Compute inverse of A
# invA = np.linalg.inv(A)
# print(f"(f) Inverse of A:\n{invA}")
```

```
# # Solve Ax = b
# x = np.dot(invA, b)
# print(f"(g) Solution to Ax = b: {x}")
```

This was the output:

```
run hw07_p03.py
(a) Determinant of A: 0.0
(b) Reduced Row Echelon Form of A:
Matrix([[1, 0, -7/3], [0, 1, 2/3], [0, 0, 0]])
(d) Rank of A: 2
(c) Reduced Row Echelon Form of Ab:
Matrix([[1, 0, -7/3, 0], [0, 1, 2/3, 0], [0, 0, 0, 1]])
(e) Rank of Ab: 3
```

(f) The inverse does not exist.
(g) There is no solution to the problem.

In Matlab:

This was my MatLab script

```
clear all;
A = [1 2 -1; 1 5 1; 2 7 0]
b = [1; 2; 5]
detA = det(A)
rrefA = rref(A)
rankA = rank(A)
Ab = [A,b]
rrefAb = rref(Ab)
rankAb = rank(Ab)
```

This was the output:

```
>> hw9p03

A =
     1     2    -1
     1     5     1
     2     7     0

b =
     1
     2
     5

detA =     0

rrefA =
    1.0000         0   -2.3333
         0    1.0000    0.6667
         0         0         0

rankA =     2

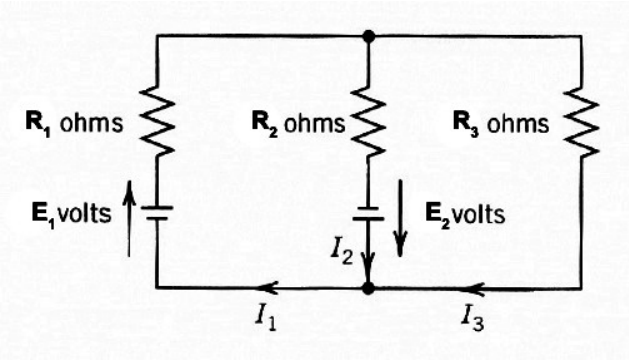
Ab =
```

```
     1     2    -1     1
     1     5     1     2
     2     7     0     5

rrefAb =
    1.0000         0   -2.3333         0
         0    1.0000    0.6667         0
         0         0         0    1.0000

rankAb =     3
```

Problem 4. (8 points)
Consider the following circuit.



Circuit analysis is performed by using Kirchoff's Current & Voltage laws. Kirchoff's current law states that the sum of the currents entering and leaving any node must sum to zero. So for the bottom node, Kirchoff's current law provides the following equation.

$$-i_1 + i_2 + i_3 = 0$$

where *i* denotes current. Kirchoff's voltage law states that in a loop the sum of the voltages is zero. The application of Kirchoff's voltage law to the right and left loops respectively yields

$$R_1 i_1 + R_2 i_2 = E_1 + E_2$$

$$-R_2 i_2 + R_3 i_3 = -E_2$$

where *R* denotes a resistance and *E* denotes an applied voltage. Consider an example where the following values are assigned: $R_1 = 15, R_2 = 15, R_3 = 25, E_1 = 50, E_2 = 80$

(a) Write this set of equations in matrix notation, $\underline{A}\underline{x} = \underline{b}$. Identify all three quantities, $\underline{A}$, $\underline{x}$ and $\underline{b}$.
(b) Calculate the determinant of $\underline{A}$.
(c) How many solutions are there to this problem?
(d) Calculate the currents.

Solution:

(a) Write this set of equations in matrix notation, $\underline{\underline{A}}\underline{x} = \underline{b}$. Identify all three quantities, $\underline{\underline{A}}$, $\underline{x}$ and $\underline{b}$.

The three independent equations can be written in matrix form as:

$$\begin{bmatrix} -1 & 1 & 1 \\ R_1 & R_2 & 0 \\ 0 & -R_2 & R_3 \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ i_3 \end{bmatrix} = \begin{bmatrix} 0 \\ E_1 + E_2 \\ -E_2 \end{bmatrix}$$

(b) Calculate the determinant of $\underline{\underline{A}}$.
See the script below.

(c) How many solutions are there to this problem?
Below we show that the determinant is non-zero, so there is one solution.

(d) Calculate the currents.

In Python:

This was my script

```
import numpy as np

R1 = 15;
R2 = 15;
R3 = 25;
E1 = 50;
E2 = 80;

# Define A matrix
A = np.array([[-1, 1, 1],
              [R1, R2, 0],
              [0, -R2, R3]])

# Define b vector
b = np.array([0, E1+E2, -E2])

# Compute determinant of A
detA = np.linalg.det(A)
print(f"Determinant of A: {detA}")

# Compute inverse of A
invA = np.linalg.inv(A)
print(f"Inverse of A:\n{invA}")

# Solve Ax = b
x = np.dot(invA, b)
print(f"Solution to Ax = b: {x}")
```

This was the output:

```
run hw07_p04.py
Determinant of A: -975.0000000000001
Inverse of A:
```

```
[[-0.38461538  0.04102564  0.01538462]
 [ 0.38461538  0.02564103 -0.01538462]
 [ 0.23076923  0.01538462  0.03076923]]
Solution to Ax = b: [ 4.1025641   4.56410256 -0.46153846]
```

In Matlab:

This was my MatLab script

```
clear all;
n = 3;
A = zeros(n,n);
x = zeros(n,1);
b = zeros(n,1);
%
R1 = 15;
R2 = 15;
R3 = 25;
E1 = 50;
E2 = 80;
%
A = [-1 1 1
    R1 R2 0
    0  -R2 R3]

b = [0; E1+E2; -E2]

detA = det(A)
rankA = rank(A)
rankAb =rank([A,b])
invA = inv(A);
x = invA*b
```

(b) Calculate the determinant of $\underline{A}$.
(c) Calculate the rank of $\underline{A}$.
(d) Calculate the rank of $\underline{A}|\underline{b}$
(e) How many solutions are there to this problem?
(f) Calculate the currents.

This script generated the following results.

```
>> xm3p02_f18

A =
    -1     1     1
    15    15     0
     0   -15    25

b =
     0
   130
   -80
```

```
detA = -975.0000

rankA =3

rankAb =  3

x =
    4.1026
    4.5641
   -0.4615
```

$i_1 = 4,.1026 i_2 = 4.5641, i_3 = -0.4615$

Problem 5. (8 points)

For an ideal mixture, the volume of the mixture, V_{mix}, is given by the sum of the pure component molar volumes, V_i, weighted by the mole fraction, x_i. Similarly, the enthalpy of the mixture, H_{mix}, is given by the sum of the pure component enthalpies, H_i, weighted by the mole fraction. The cost of the mixture, C_{mix}, is also given by the sum of the pure component costs, C_i, weighted by the mole fraction. As a reminder, the sum of the mole fractions is unity.

$$V_{mix} = \sum_{i=1}^{n_c} x_i V_i \qquad H_{mix} = \sum_{i=1}^{n_c} x_i H_i \qquad C_{mix} = \sum_{i=1}^{n_c} x_i C_i \qquad 1 = \sum_{i=1}^{n_c} x_i$$

where n_c is the number of components. Now consider a four component ideal mixture ($n_c = 4$) with the following pure component properties and mixture properties.

component	A	B	C	D	mixture
molar volume (liter/mol)	11	14	17	8	12.8
enthalpy (kJ/mol)	49	74	45	63	57.1
cost ($/mol)	40.18	89.67	38.22	55.99	56.216

(a) Is this system of algebraic equations linear or nonlinear? (2 pts)
(b) Determine the composition of this mixture. Show reasoning and method. (6 pts)

Solution:

This is a set of linear algebraic equations, which can be written as follows.

$$x_A V_A + x_B V_B + x_C V_C + x_D V_D = V_{mix}$$
$$x_A H_A + x_B H_B + x_C H_C + x_D H_D = H_{mix}$$
$$x_A C_A + x_B C_B + x_C C_C + x_D C_D = C_{mix}$$
$$x_A + x_B + x_C + x_D = 1$$

This set of linear algebraic equations can be written in matrix form as

$$\underline{\underline{A}}\,\underline{x} = \underline{b}$$

where $\underline{\underline{A}} = \begin{bmatrix} V_A & V_B & V_C & V_D \\ H_A & H_B & H_C & H_D \\ C_A & C_B & C_C & C_D \\ 1 & 1 & 1 & 1 \end{bmatrix}$, $\underline{b} = \begin{bmatrix} V_{mix} \\ H_{mix} \\ C_{mix} \\ 1 \end{bmatrix}$ and $\underline{x} = \begin{bmatrix} x_A \\ x_B \\ x_C \\ x_D \end{bmatrix}$.

In Python:

This was my script

```
import numpy as np

# Define A matrix
A = np.array([[11, 14, 17, 8],
              [49, 74, 45, 63],
              [40.18,  89.67,  38.22, 55.99],
              [1, 1, 1, 1]])

# Define b vector
b = np.array([12.8, 57.1, 56.216, 1.0])

# Compute determinant of A
detA = np.linalg.det(A)
print(f"Determinant of A: {detA}")

# Compute inverse of A
invA = np.linalg.inv(A)
print(f"Inverse of A:\n{invA}")

# Solve Ax = b
x = np.dot(invA, b)
print(f"Solution to Ax = b: {x}")
```

This was the output:

```
run hw07_p05.py
Determinant of A: -1231.3799999999965
Inverse of A:
[[-0.33358508 -0.33274862  0.16810408 14.2196966 ]
 [-0.02907307 -0.07226039  0.05847098  1.51119882]
 [ 0.24168819  0.1590898  -0.09501535 -6.63625363]
 [ 0.12096997  0.24591921 -0.13155971 -8.09464178]]
Solution to Ax = b: [0.4 0.3 0.2 0.1]
```

In Matlab:

This was my MatLab script

```
clear all;
A = [11     14      17      8
     49     74      45      63
```

```
     40.18  89.67   38.22   55.99
     1      1       1       1];
b = [12.8
   57.1
   56.216
    1.0];
detA = det(A)
invA = inv(A)
x = inv(A)*b
```

At the command line prompt, I executed the script with the command

```
>> xm03p01_f17
```

which generated the following output

```
detA =  -1.2314e+03

invA =
   -0.3336   -0.3327    0.1681   14.2197
   -0.0291   -0.0723    0.0585    1.5112
    0.2417    0.1591   -0.0950   -6.6363
    0.1210    0.2459   -0.1316   -8.0946

x =
    0.4000
    0.3000
    0.2000
    0.1000
```

Therefore the composition of the mixture is given by

$$\underline{x} = \begin{bmatrix} x_A \\ x_B \\ x_C \\ x_D \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.3 \\ 0.2 \\ 0.1 \end{bmatrix}$$