Linear Algebra Homework

In [1]:
```
import numpy as np
import scipy.linalg as la
```

Problem 1: Condition Number, Accuracy, and Iterative Updating

In [2]:
```
# Defining the arrays

A = np.array([[0.780, 0.563], [0.913, 0.659]])
b = np.array([[0.217],[0.254]])
x = np.array([[1],[-1]])
xalpha = np.array([[0.999],[-1.001]])
xbeta = np.array([[0.341],[-0.087]])
```

Part A

In [15]:
```
ralpha = b - A@xalpha
rbeta = b - A@xbeta

print("r_alpha = ",ralpha)
print("r_beta = ",rbeta)
```

```
r_alpha =  [[0.001343]
 [0.001572]]
r_beta =  [[1.e-06]
 [0.e+00]]
```

The less accurate solution produced the smaller residual.

Part B

In [8]:
```
U, s, VT = la.svd(A, full_matrices = True)
print("U = ")
print(U)
print("s = ") # The singular values
print(s)
print("VT = ")
print(VT)
```

```
U =
[[-0.64955581 -0.76031391]
 [-0.76031391  0.64955581]]
s =
[1.48095206e+00 6.75241305e-07]
VT =
[[-0.81084336 -0.58526323]
 [-0.58526323  0.81084336]]
```

In [65]:
```
conditionNumber = np.max(s)/np.min(s);
print("Condition Number = ", conditionNumber)
print("Reciprocal = ", 1/conditionNumber)
```

```
Condition Number =  2193218.9997129813
Reciprocal =  4.559508193804934e-07
```

In [72]:
```
print("V^T * (xalpha - x) =")
print(VT@(xalpha - x))

print("\nV^T * (xbeta - x) =")
print(VT@(xbeta - x))
```

```
V^T * (xalpha - x) =
[[ 0.00139611]
 [-0.00022558]]

V^T * (xbeta - x) =
[[4.38606914e-07]
 [1.12598845e+00]]
```

From the above result, we see that $\vec{x_\alpha}$ has most of its error associated with the large singular value, and $\vec{x_\beta}$ has most of its error associated with the small singular value.

Part C

In [39]:

```
x_blackbox = la.solve(A,b)
difference = x_blackbox - x;
machine = 2.2e-16;

print("Difference between the black box and exact solutions:")
print(difference)

print("\nDividing that answer by machine accuracy of 2.2 x 10^(-16):")
print(difference/machine)

print("\nWell, I wouldn't exactly say that that is close to machine acc

print("\nCalculating Ax - b for the two solutions:")
print("Exact:\n",A@x - b)
print("Black Box:\n",A@x_blackbox - b)
print("\nThose answers both seem to satisfy the equation to less than m
print("The problem must then be rounding errors in calculating A and b.
```

```
Difference between the black box and exact solutions:
[[-2.30988562e-11]
 [ 3.20018456e-11]]

Dividing that answer by machine accuracy of 2.2 x 10^(-16):
[[-104994.80073246]
 [ 145462.93462688]]

Well, I wouldn't exactly say that that is close to machine accuracy...

Calculating Ax - b for the two solutions:
Exact:
 [[8.32667268e-17]
 [0.00000000e+00]]
Black Box:
 [[-2.77555756e-17]
 [ 0.00000000e+00]]

Those answers both seem to satisfy the equation to less than machine accu
racy.
The problem must then be rounding errors in calculating A and b.
```

Part D

In [47]:
```
print("Original x-beta:\n",xbeta)

print("\nr_beta:\n",rbeta)

delta_x = la.solve(A,-rbeta)

x_improved = xbeta - delta_x

print("\nImproved x-value:\n",x_improved)

print("This is an improvement over the original x-beta value.")
```

```
Original x-beta:
 [[ 0.341]
 [-0.087]]

r_beta:
 [[1.e-06]
 [0.e+00]]

Improved x-value:
 [[ 1.]
 [-1.]]
This is an improvement over the original x-beta value.
```

Now we want to try finding $\vec{r_\beta}$ analytically, which is shown below:

In [50]:
```
rbeta = b - A@xbeta # Exact same definition as before
print("r_beta:\n",rbeta)
```

```
r_beta:
 [[1.e-06]
 [0.e+00]]
```

So now we will round to the correct residual values:

In [51]:
```
rbeta_exact = np.array([[1.0e-6],[0.0]])
print("r_beta_exact:\n",rbeta_exact)
```

```
r_beta_exact:
 [[1.e-06]
 [0.e+00]]
```

Calculating the new $\delta\vec{x}$ value gives us:

In [57]:
```
delta_x_exact = la.inv(A)@b

x_improved_exact = xbeta - delta_x_exact

print("Improved x-value:\n",x_improved_exact)
```

```
Improved x-value:
 [[-0.659]
 [ 0.913]]
```

That result is an improvement over the original $\vec{x_\beta}$, though not as dramatic of an improvement as the former was.

Problem 2: The Sherman-Morrison Formula

First, we will define the matrices T and C:

In [92]:
```
T = np.mat('[2 1 0 0 0; 1 2 1 0 0; 0 1 2 1 0; 0 0 1 2 1; 0 0 0 1 2]')
C = np.mat('[2 1 0 0 1; 1 2 1 0 0; 0 1 2 1 0; 0 0 1 2 1; 1 0 0 1 2]')

print("T = \n",T)
print("C = \n",C)
```

```
T =
 [[2 1 0 0 0]
 [1 2 1 0 0]
 [0 1 2 1 0]
 [0 0 1 2 1]
 [0 0 0 1 2]]
C =
 [[2 1 0 0 1]
 [1 2 1 0 0]
 [0 1 2 1 0]
 [0 0 1 2 1]
 [1 0 0 1 2]]
```

Now we will compute their inverses:

In [89]:
```
Tinv = la.inv(T)
Cinv = la.inv(C)

print("Tinv = \n",Tinv)
print("Cinv = \n",Cinv)
```

```
Tinv = 
 [[ 0.83333333 -0.66666667  0.5        -0.33333333  0.16666667]
 [-0.66666667  1.33333333 -1.          0.66666667 -0.33333333]
 [ 0.5        -1.          1.5        -1.          0.5       ]
 [-0.33333333  0.66666667 -1.          1.33333333 -0.66666667]
 [ 0.16666667 -0.33333333  0.5        -0.66666667  0.83333333]]
Cinv = 
 [[ 1.25 -0.75  0.25  0.25 -0.75]
 [-0.75  1.25 -0.75  0.25  0.25]
 [ 0.25 -0.75  1.25 -0.75  0.25]
 [ 0.25  0.25 -0.75  1.25 -0.75]
 [-0.75  0.25  0.25 -0.75  1.25]]
```

And now we will compute the difference between the two

In [90]:
```
delta = Tinv - Cinv
print("Difference = \n",delta)
```

```
Difference = 
 [[-0.41666667  0.08333333  0.25       -0.58333333  0.91666667]
 [ 0.08333333  0.08333333 -0.25        0.41666667 -0.58333333]
 [ 0.25       -0.25        0.25       -0.25        0.25      ]
 [-0.58333333  0.41666667 -0.25        0.08333333  0.08333333]
 [ 0.91666667 -0.58333333  0.25        0.08333333 -0.41666667]]
```

Checking this with the Sherman-Morris formula, where the difference should be

$$\Delta = \frac{\vec{z} \otimes \vec{w}}{1+\lambda} \tag{1}$$

where

$$\vec{z} = A^{-1} \cdot \vec{u} \tag{2}$$

$$\vec{w} = \vec{v} \cdot A^{-1} \tag{3}$$

$$\lambda = \vec{v} \cdot \vec{z} \tag{4}$$

And vectors $\vec{u}$ and $\vec{v}$ are defined such that

$$C = T + \vec{u} \otimes \vec{v} \tag{5}$$

where the $\otimes$ symbol denotes a tensor product.

In [143]:

```
u = np.mat('[1; 0; 0; 0; 1]')
v = np.mat('[1 0 0 0 1]')

print("u = \n",u)

print("\nv = \n",v)

print("\nkron(u,v) = \n",la.kron(u,v))

print("\nT + kron(u,v) = \n",T + la.kron(u,v))
```

```
u = 
 [[1]
 [0]
 [0]
 [0]
 [1]]

v = 
 [[1 0 0 0 1]]

kron(u,v) = 
 [[1 0 0 0 1]
 [0 0 0 0 0]
 [0 0 0 0 0]
 [0 0 0 0 0]
 [1 0 0 0 1]]

T + kron(u,v) = 
 [[3 1 0 0 1]
 [1 2 1 0 0]
 [0 1 2 1 0]
 [0 0 1 2 1]
 [1 0 0 1 3]]
```

In [153]:

```
z = la.inv(T)@u
w = v@la.inv(T)
lam = v@z

delta_sherman = la.kron(z,w) / (1 + lam)

print("Difference = \n",delta_sherman)

print(Tinv - delta_sherman)
```

```
Difference =
 [[ 0.33333333 -0.33333333  0.33333333 -0.33333333  0.33333333]
 [-0.33333333  0.33333333 -0.33333333  0.33333333 -0.33333333]
 [ 0.33333333 -0.33333333  0.33333333 -0.33333333  0.33333333]
 [-0.33333333  0.33333333 -0.33333333  0.33333333 -0.33333333]
 [ 0.33333333 -0.33333333  0.33333333 -0.33333333  0.33333333]]
[[ 5.00000000e-01 -3.33333333e-01  1.66666667e-01  5.55111512e-17
  -1.66666667e-01]
 [-3.33333333e-01  1.00000000e+00 -6.66666667e-01  3.33333333e-01
   0.00000000e+00]
 [ 1.66666667e-01 -6.66666667e-01  1.16666667e+00 -6.66666667e-01
   1.66666667e-01]
 [ 0.00000000e+00  3.33333333e-01 -6.66666667e-01  1.00000000e+00
  -3.33333333e-01]
 [-1.66666667e-01 -5.55111512e-17  1.66666667e-01 -3.33333333e-01
   5.00000000e-01]]
```