Part 5: Unique Solutions and Least Squares for an Overdetermined Matrix

In [ ]:
```
import pandas as pd
import numpy as np
```

A)

Displaying the singular values for matrix A

In [ ]:
```
A = np.array([[6, 5, 3],
              [4, 1, 5],
              [6, 3, 6],
              [5, 3, 6],
              [6, 6, 3]])

u, s, v = np.linalg.svd(A)

s
```

Out[ ]:
```
array([17.92667464,  4.69961929,  0.74021274])
```

B)

Finding the eigenvalues of A.

In [ ]:
```
# a_square = np.matmul(A.T, A)
a_square = A.T @ A
a_square
```

```
Out[ ]: array([[149, 103, 122],
               [103,  80,  74],
               [122,  74, 115]])
```

```python
In [ ]: eigenvalues, eigenvectors = np.linalg.eig(a_square)
        eigenvalues
```

```
Out[ ]: array([321.36566366,   0.54791491,  22.08642144])
```

The singular value σ_i is related to the eigenvalue λ_i by:

$$\sigma_i = \sqrt{\lambda_i}$$

C)

Because there are only 3 singular values in A), $rank(A) = 3$

```python
In [ ]: np.linalg.matrix_rank(A)
```

```
Out[ ]: 3
```

D)

Yes, even though the column space and row space do not have the same dimensionality, m > n and each column vector is linearly independent. This means that our input space is $\mathbb{R}^3$ and maps onto $\mathbb{R}^5$ - however they do not span all of $\mathbb{R}^5$. The three column vectors are linearly independent meaning $Span(v_1, v_2, v_3) = 3$, where v_i is a column vector in A. Because the $Span(v_1, v_2, v_3) = dim(Col(A))$, if there is a solution in the solution space, it must be unique.

E)

Finding solution or approximation for

$$b = \begin{bmatrix} -1 \\ 2 \\ 0 \\ 3 \\ 1 \end{bmatrix}$$

In [ ]:
```
b = np.array([-1, 2, 0, 3, 1])
b
```

Out[ ]:
```
array([-1,  2,  0,  3,  1])
```

In [ ]:
```
A, b
```

Out[ ]:
```
(array([[6, 5, 3],
        [4, 1, 5],
        [6, 3, 6],
        [5, 3, 6],
        [6, 6, 3]]),
 array([-1,  2,  0,  3,  1]))
```

Demonstration that $Ax = b$ does not have a solution

In [ ]:
```
np.linalg.solve(A, b)
```

```
---------------------------------------------------------------------------
LinAlgError                               Traceback (most recent call last)
Cell In[8], line 1
----> 1 np.linalg.solve(A, b)

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-packages\Python312
\site-packages\numpy\linalg\linalg.py:396, in solve(a, b)
    394 a, _ = _makearray(a)
    395 _assert_stacked_2d(a)
--> 396 _assert_stacked_square(a)
    397 b, wrap = _makearray(b)
    398 t, result_t = _commonType(a, b)

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-packages\Python312
\site-packages\numpy\linalg\linalg.py:213, in _assert_stacked_square(*arrays)
    211 m, n = a.shape[-2:]
    212 if m != n:
--> 213     raise LinAlgError('Last 2 dimensions of the array must be square')

LinAlgError: Last 2 dimensions of the array must be square
```

Finding the value of x that minimizes $||Ax - b||$.

In [ ]:

```
x, _, _, s = np.linalg.lstsq(A, b, rcond=None)
x
```

Out[ ]:

```
array([-2.06119825,  1.37336076,  1.54641296])
```