In [2]:
```
import numpy as np
from scipy.io import loadmat
```

Question 1-a

$$V = \begin{bmatrix} 1 & X & X \\ X & 2 & 4 \\ -1 & 2 & X \\ X & -2 & X \end{bmatrix}$$

1 rank matrix

$$V = \begin{bmatrix} 1 & -2 & -4 \\ -1 & 2 & 4 \\ -1 & 2 & 4 \\ 1 & -2 & -4 \end{bmatrix}$$

Question 1-b

We need to know at least 4 elements to fill in the rank 1, so we can miss at most 8 and still complete, so we can miss at minimum 9 and not be able to complete it.

Question 2

In [19]:
```
Xtrue = loadmat("incomplete.mat")["Xtrue"]
Y1 = loadmat("incomplete.mat")["Y1"]
Y2 = loadmat("incomplete.mat")["Y2"]
Y3 = loadmat("incomplete.mat")["Y3"]

def ItSingValThresh(Y, r=1):
    """
    Iterative Singular Value Thresholding function for Matrix Completion
    """
    tol = 10**(-3)  # difference between iterates at termination
    max_its = 100;
    n,p = Y.shape
    X = np.array(Y) #make a copy so operations do not mutate the original
    X[np.isnan(X)] = 0 # Fill in missing entries with zeros

    err = 10**6
    itt = 0
```

```
    while err > tol and itt < max_its:
        U,s,VT = np.linalg.svd(X, full_matrices=False)
        V, S = VT.T, np.diag(s)
        Xnew = U[:, :r] @ np.diag(s[:r]) @ V.T[:r, :] ### Complete this line
        for i in range(n):
            for j in range(p):
                if ~np.isnan(Y[i,j]):  #replace Xnew with known entries
                    Xnew[i,j] = Y[i,j]
        err = np.linalg.norm(X-Xnew,'fro')
        X = Xnew
        itt += 1
    return X

recovered_Y1 = ItSingValThresh(Y1, r=2)
recovered_Y2 = ItSingValThresh(Y2, r=2)
recovered_Y3 = ItSingValThresh(Y3, r=2)

num_nans_Y1 = np.sum(np.isnan(Y1))
num_nans_Y2 = np.sum(np.isnan(Y2))
num_nans_Y3 = np.sum(np.isnan(Y3))

print(f"Number of NaNs in Y1: {num_nans_Y1}")
print(f"Number of NaNs in Y2: {num_nans_Y2}")
print(f"Number of NaNs in Y3: {num_nans_Y3}")

print("Difference between recovered Y1 and Xtrue:", np.linalg.norm(recovered
print("Difference between recovered Y2 and Xtrue:", np.linalg.norm(recovered
print("Difference between recovered Y3 and Xtrue:", np.linalg.norm(recovered
```

```
Number of NaNs in Y1: 136
Number of NaNs in Y2: 76
Number of NaNs in Y3: 16
Difference between recovered Y1 and Xtrue: 87.24667705099742
Difference between recovered Y2 and Xtrue: 0.00473559952740616
Difference between recovered Y3 and Xtrue: 0.0007153218655176282
```

We can see that less NaN's means less error

Question 2-b

In [20]:
```
recovered_Y1_rank3 = ItSingValThresh(Y1, r=3)
recovered_Y2_rank3 = ItSingValThresh(Y2, r=3)
recovered_Y3_rank3 = ItSingValThresh(Y3, r=3)

print("Difference between recovered Y1 and Xtrue (rank=3):", np.linalg.norm(
print("Difference between recovered Y2 and Xtrue (rank=3):", np.linalg.norm(
print("Difference between recovered Y3 and Xtrue (rank=3):", np.linalg.norm(
```

```
Difference between recovered Y1 and Xtrue (rank=3): 128.77804846772108
Difference between recovered Y2 and Xtrue (rank=3): 48.97940976510773
Difference between recovered Y3 and Xtrue (rank=3): 20.785069891601783
```

The error is worse when we use the wrong rank