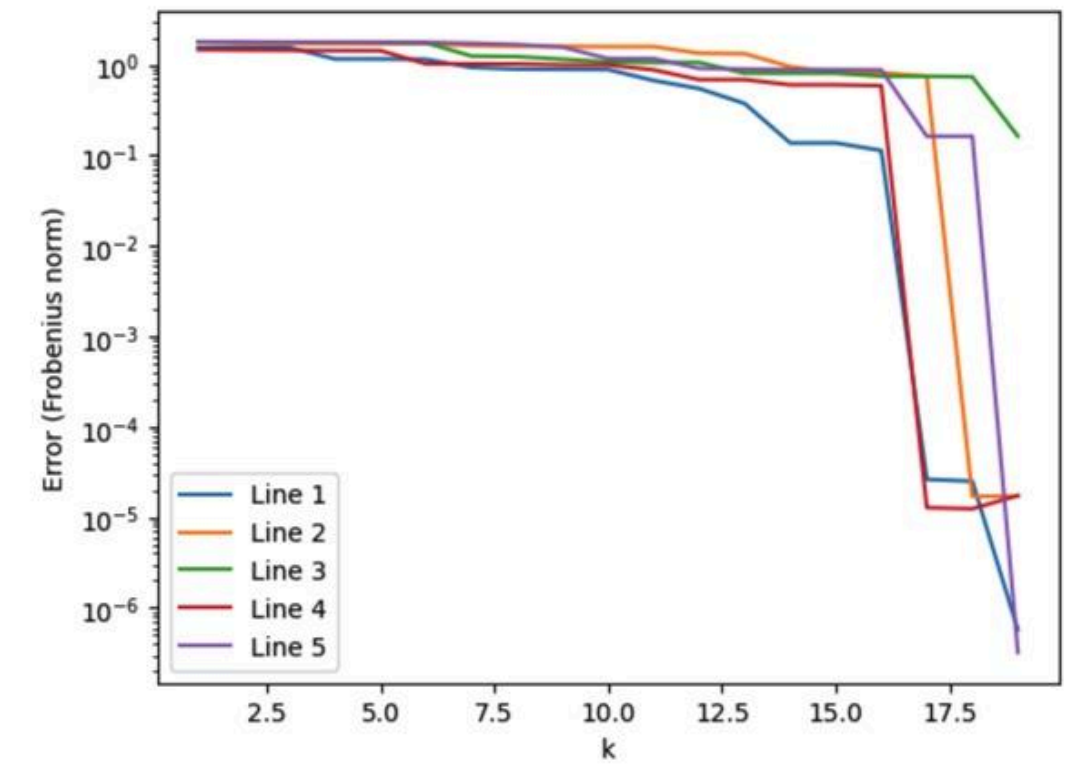
```python
import cvxpy as cp
import numpy as np
import matplotlib.pyplot as plt

def nuclear_norm_minimization(X, known_indices, tol=1e-5):
    X_star = cp.Variable(shape=X.shape)

    objective = cp.Minimize(cp.normNuc(X_star))

    constraints = [X_star[idx[0],idx[1]] == X[idx[0],idx[1]] for idx in known_indices]

    problem = cp.Problem(objective, constraints)
    problem.solve(solver=cp.SCS)

    return X_star.value
```

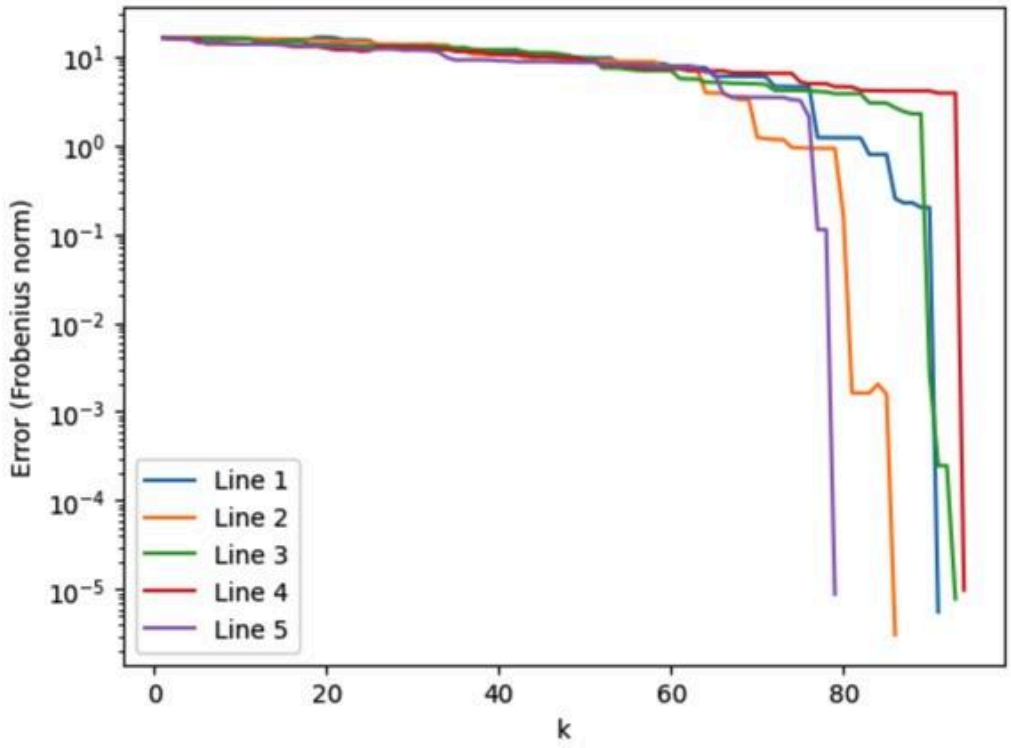
Code for minimization of Nuclear norm

```python
import random
#first we try experiments with W1
#start with k=1
def main(matrix):
  know_indices1 = []
  error1 = []
  for k in range(1, len(W1)*len(W1[0])):
      i, j = np.random.randint(len(W1)), np.random.randint(len(W1[0]))
      while (i,j) in know_indices1:
          i, j = np.random.randint(len(W1)), np.random.randint(len(W1[0]))
      know_indices1.append((i,j))
      W1_star = nuclear_norm_minimization(W1, know_indices1)
      error = np.linalg.norm(W1 - W1_star, 'fro')
      error1.append(error)
      if error < 1e-5:
          break
  print('the threshold is reached after iteration',k)
  return error1,k
```

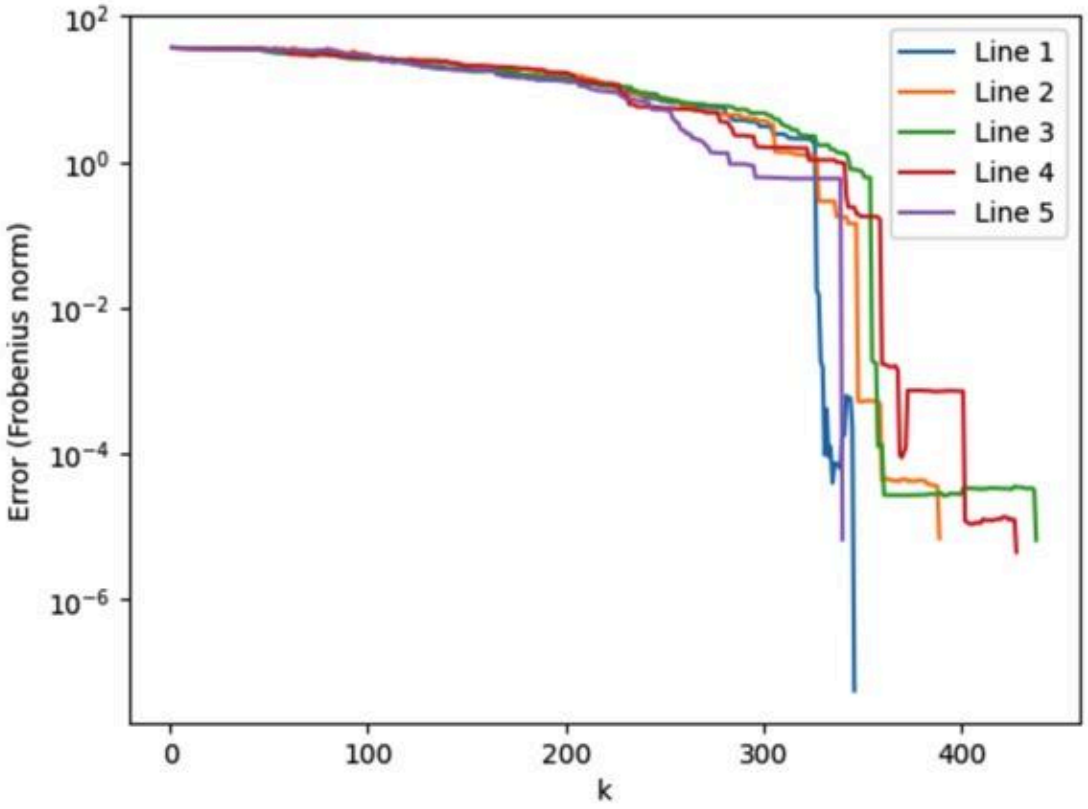
Code for increasing choices of k. This function can automatically generate a successive index sequence randomly, and increase the value of k until the error threshold is reached.



Plot for the log of Error for example matrix 1. All five attempts show that there's a sudden drop in error value when k reaches a threshold value which is around 17.



Plot for the log of Error for example matrix 2. All five attempts show that there's a sudden drop in error value after k reaches some value in the range (70,100). Also, five attempts reached the threshold after 91,86,93,94,79 iterations respectively.



Plot of the log of Error for example matrix 3. All five attempts show that there's a sudden drop in error value after k reaches some value in the range (300,450). Also, five attempts reached the threshold after 346,389,438.428, 340 iterations respectively.