Imports

In [3]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib as plt
import os
```

Reading all the files from my folders:

In [4]:

```
directory = r"C:\Users\maorb\CSVs"

# Our columns of interest
columns = [
    'AU02_r', 'AU04_r', 'AU05_r', 'AU06_r', 'AU07_r',
    'AU09_r', 'AU10_r', 'AU12_r', 'AU14_r', 'AU15_r', 'AU17_r',
    'AU20_r', 'AU23_r', 'AU25_r', 'AU26_r', 'AU45_r'
]

# Function to read files and process them
def read_and_process_file(file_path):
    if file_path.endswith('.csv'):
        df = pd.read_csv(file_path)
    elif file_path.endswith('.xlsx'):
        df = pd.read_excel(file_path)
    else:
        return None

    df.columns = df.columns.str.strip() # Strip whitespace from column names
    return df[columns]

# Get all files that start with "Argaman"
files = [f for f in os.listdir(directory) if f.startswith("Argaman") and (f.endswit

# Read and process all files
dataframes = [read_and_process_file(os.path.join(directory, file)) for file in file

# Drop any None values in case some files were not processed
dataframes = [df for df in dataframes if df is not None]

# Find the minimum length of all dataframes
min_length = min(len(df) for df in dataframes)

# Trim all dataframes to the minimum length
dataframes = [df.iloc[:min_length, :] for df in dataframes]

# Combine the dataframes
combined_data = dataframes[0]
for df in dataframes[1:]:
    df = df.add_suffix(f'_{df}')  # Add suffix to each dataframe to avoid column na
    combined_data = combined_data.join(df)

dataframes_Trans = [df.T for df in dataframes]
```

Our Identity Matrix model

In [20]:

```
def W_i_calc3(X_i, S):
    X_S_T = np.dot(X_i, S.T)
    U_i, Sigma, V_i_T = np.linalg.svd(X_S_T, full_matrices=False)
    W_i = np.dot(U_i, V_i_T)
    return W_i

def SRM(X, tol=1e-6, max_iter=1000):
    dist_vec = []
    indices = []
    W_i_vec = []
    W_i_new_vec = []
    m = 16
    W_i = np.zeros((m,m)) # Initialize W_i as a matrix of ones
    #W_i = W_i.reshape(m, 1) # Convert W_i to a column vector
    #S = np.sum(np.dot())
    iter_count = 0
    converged = False
    k = 0

    for j, X_i in enumerate(X):
        #X_i = np.array(X_i).reshape(m, 1)  # Ensure X_i is a column vector
        # Calculate S
        S = (1 / 8) * np.dot(W_i.T, X_i)
        W_i_new = W_i_calc3(X_i, S)
        dist = np.linalg.norm(X_i - np.dot(W_i, S), 'fro')
        dist_vec.append(dist)
        indices.append(j)
        W_i = W_i_new
        W_i_new_vec.append(W_i)
    while not converged and iter_count < max_iter:
        for j,X_i in enumerate(X):
            S_sum = np.dot(W_i.T, X_i)  # Initialize the accumulator for S
            for W_i, X_i in zip(W_i_new_vec, X):
                S_sum += np.dot(W_i.T, X_i)
            S = 1 / 8 * S_sum
            W_i_new = W_i_calc3(X_i, S)
            dist = np.linalg.norm(X_i - np.dot(W_i, S), 'fro')
            dist_vec.append(dist)
            k +=1
            indices.append(k)
            W_i = W_i_new
            W_i_new_vec[j] = W_i
            if dist < tol:
                converged = True
                break
            """
            print(f"Iteration {iter_count}, Column {j}:")
            print(f"W_i: {W_i.flatten()[:10]}")  # Print first 10 values for brevit
            print(f"S: {S.flatten()[:10]}")
            print(f"Distance: {dist}")
            print(f"W_i_new: {W_i_new.flatten()[:10]}")
            print("\n")
            """
            iter_count += 1

            if iter_count >= max_iter:
                converged = True
                break

    # We can find the argmin W_i of the function ||X - W_i @ S||^2 by finding the c
    A = (np.linalg.norm(X_i - np.dot(W_i, S), 'fro'))**2  # We'll find the argmin W
    return W_i, W_i_new_vec, S, A, dist_vec, indices
```

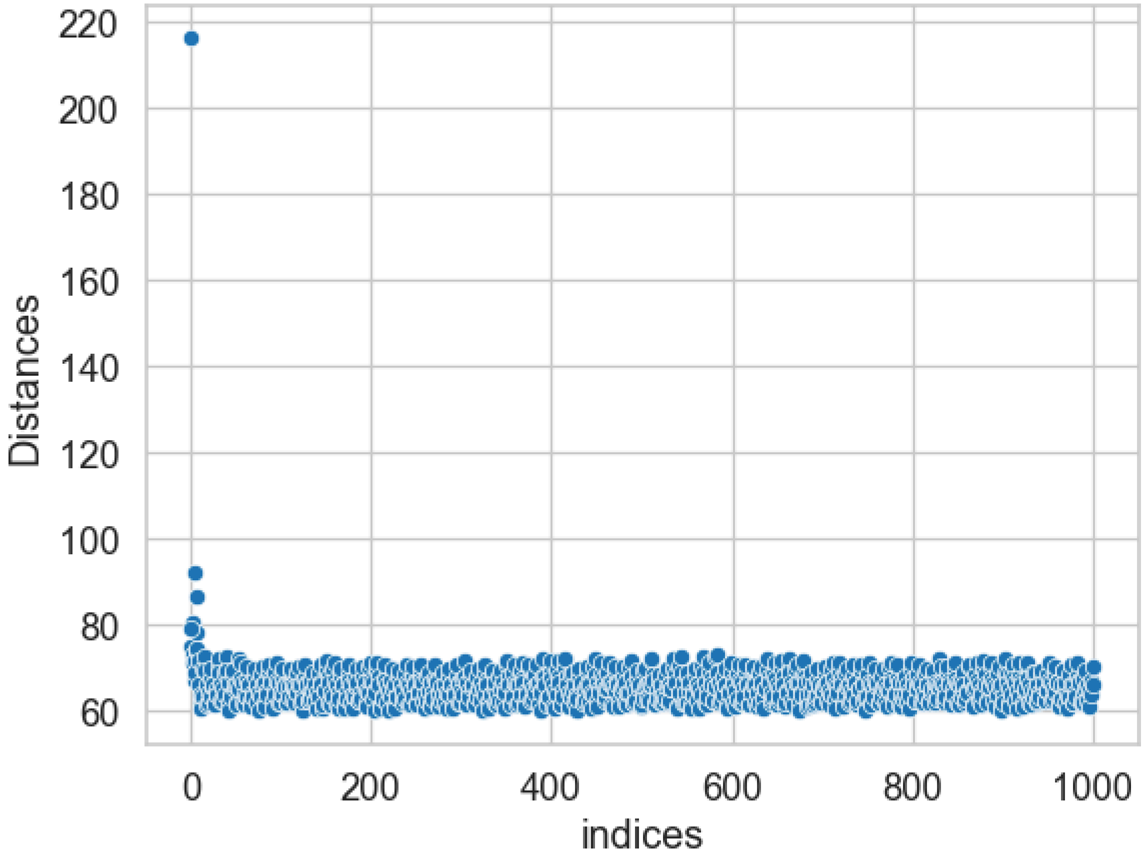
```
In [8]: datsa = SRM(dataframes_Trans)
```

```
In [21]: W_i, W_i_new_vec, S, A, dist_vec, indices = SRM(dataframes_Trans)
```

Let's plot the distances

```
In [19]: import matplotlib.pyplot as plt
         # Create a scatter plot using seaborn
         dists = pd.DataFrame({'indices': indices, 'Distances': dist_vec})
         sns.scatterplot(dists, x= 'indices', y='Distances', )

         # Set the labels and title using seaborn's functionality
         sns.set_context("notebook", font_scale=1.2)
         sns.set_style("whitegrid")
         #sns.label(x = "Indices")
```


Distances
indices