# CSE 5350 Project
# Professor Sun
# Independent Component Analysis

Team members: Aracely Arevalo, Corey Warren, Ericka Guerrero, Pablo Fernandez

# *What is Independent Component Analysis (ICA)?*

- Is a computational method used for separating a multivariate signal into subcomponents.

Multivariate signal: a signal that consists of distinguishable components.

- Equation: dimension M consists of M scalar signals

$\{ x_1(n), x_2(n), \ldots\ldots\ldots, x_M(n), n = 0,1,\ldots N\}$

- Examples: images along the columns (rows)

- ICA simplifies complex data into manageable pieces

# Abstract

Focus of are Group project:

- ❑ separate wave data made of various overlain waves using ICA

Scenario:

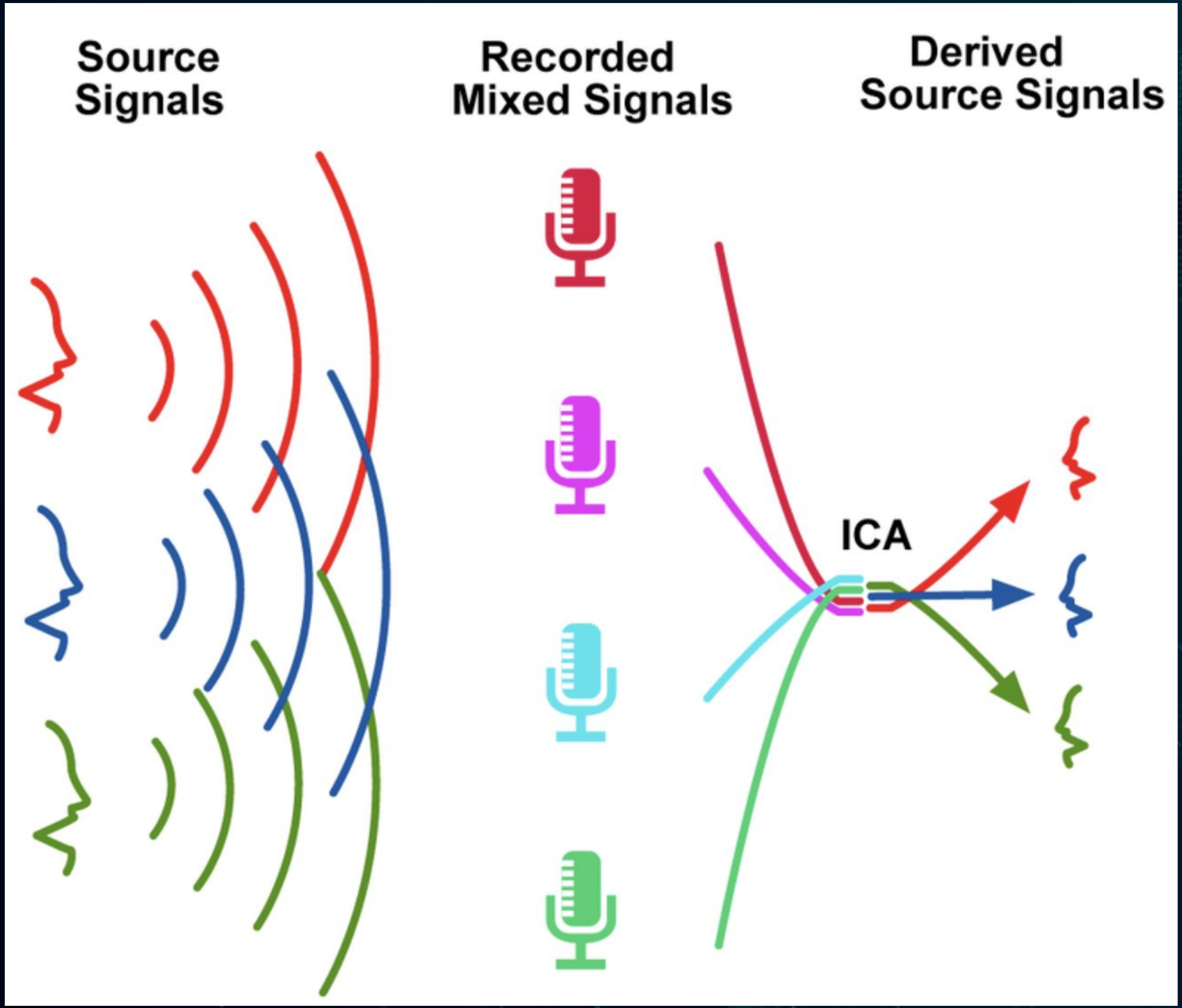
- ❑Group conversation among colleagues, differentiate individual voices

Plan:

- ❑ Use matrices, mathematical equations, and python code

# List of Methodology

1. Center X
   - subtracting the mean
   - center the data around 0
   - subtract the mean from each dimension
   - add it back to the estimation of S at the end of the problem.
2. Whitening
   - transform data to remove any possible correlation
   - transform X into X(hat).
3. Select a value
   - select random initial value for the de-mixing matrix W.
4. Calculate
   - new value of W
5. Normalize
   - W
6. Check
   - If Algorithm has or has not converged
7. Dot Product
   - after sufficient loops take the dot product of W and x to receive independent source signals



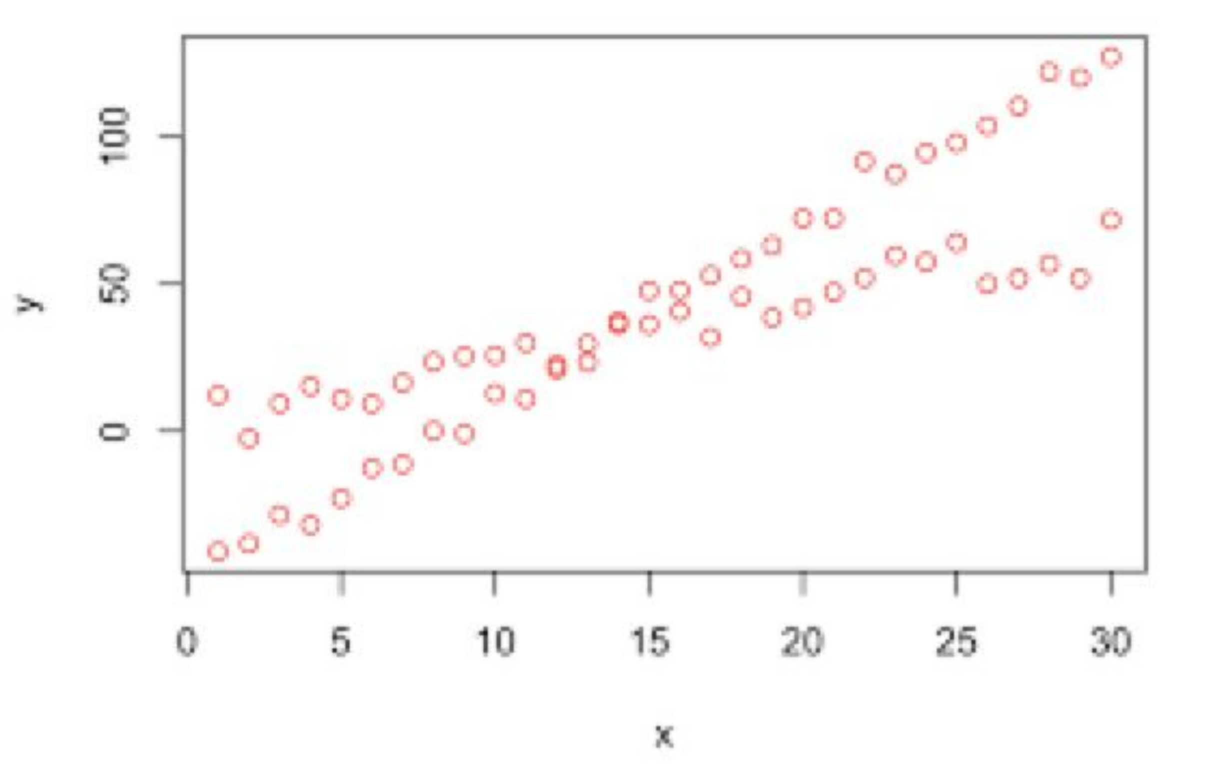

# End Goal

- separate the signals into individual components

# Experiment

## Step 1: Center X

- ❑ Select data source at the center that is around zero



# Experiment

## Step 2: Whitening

Whiten data to remove correlations.

- ❑ Transform X to X(hat)
- ❑ Whitened Signal of covariance matrix = identity matrix
- ❑ Decompose of Covariance Matrix = eigenvalue of x-hat
- ❑ D = Diagonal matrix of eigenvalues

$$I_1 = [1],\ I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},\ \cdots,\ I_n = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}.$$

$$\tilde{x} = ED^{-1/2}E^T x$$

$$D = \begin{pmatrix} \lambda_1 & 0 & 0 & \dots \\ 0 & \lambda_2 & 0 & \dots \\ 0 & 0 & \lambda_3 & \dots \\ \dots & \dots & \dots & \dots \end{pmatrix}$$

$$for\ 1\ to\ the\ number\ of\ components:$$

$$repeat\ until\ w_p^T w_{p+1} \approx 1:$$

$$w_p = \frac{1}{n}\sum_i^n X g(W^T X) - \frac{1}{n}\sum_i^n g'(W^T X) W$$

$$w_p = w_p - \sum_{j=1}^{p-1} (w_p^T w_j) w_j$$

$$w_p = \frac{w_p}{||w_p||}$$

$$W = [w1, w2 ...]$$

$$g(u) = \tanh(u)$$

$$g'(u) = 1 - \tanh^2(u)$$

# Experiment

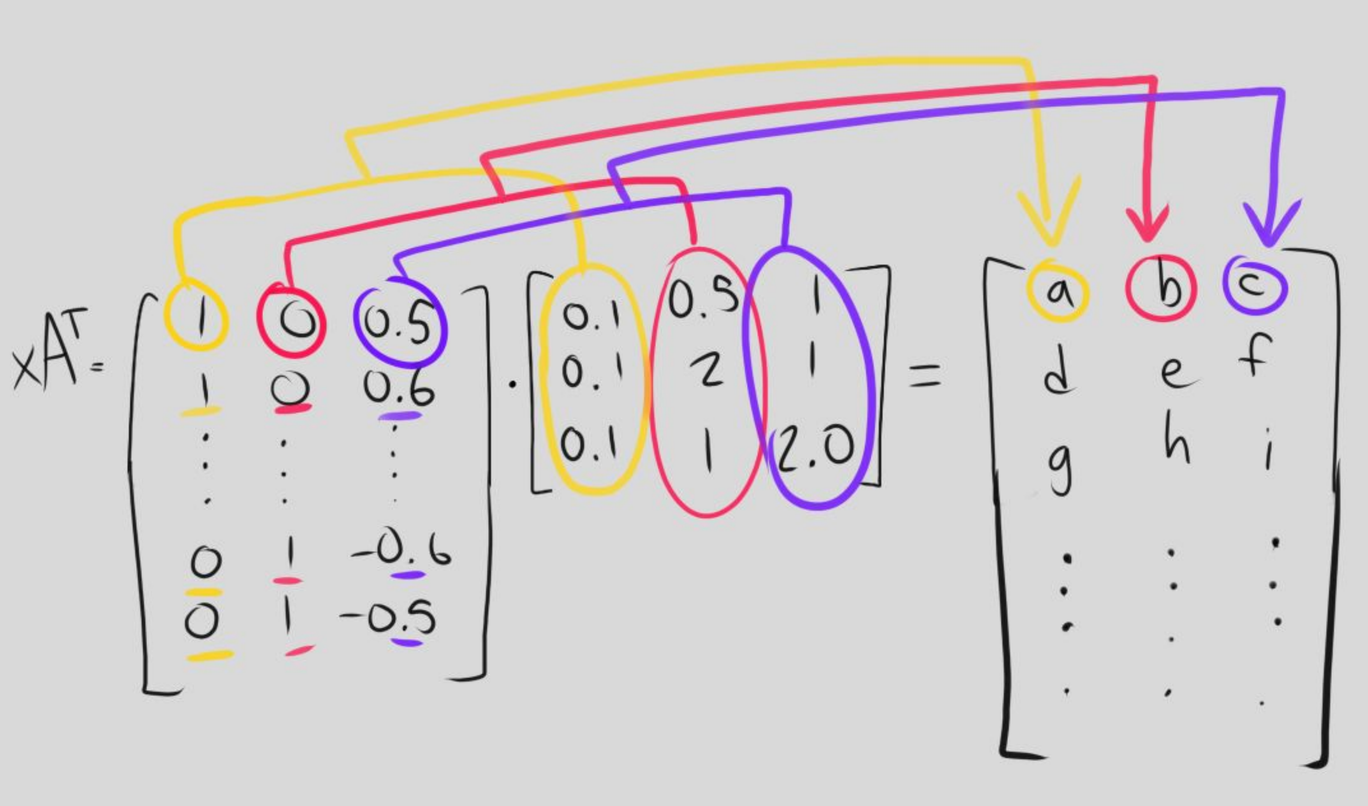
## Step 3-6: Select value, Calculate, Normalize, & Check

- ❑Determine W
- ❑Calculate new value for W
- ❑Normalize W
- ❑Check for convergence

# Experiment

## Step 7: Dot Product

- Take the Dot Product of W and X
- Results in independent signal sources




$$S = Wx$$

# Code

- Code written in Python
- Output

  >three wave signals
- Artificially mixes the wave signals together, to be later re-separated.

```
#Create the wave signals
s1 = np.sin(2 * time) #sinusoidal
s2 = np.sign(np.sin(3 * time)) # square signal
s3 = signal.sawtooth(2 * np.pi * time) #saw tooth signal

print("s1 = \n",s1)
print()
print("s2 = \n",s2)
print()
print("s3 = \n",s3)
print()
# Compute dot product of matrix A and the signals
# this will give us a combination of all three
# Then, use ICA to separate the mixed signal into the three
# original source signals

#np.c_ is a special type of np.r_
#np.r_ Translates slice objects to concatenation along the first axis.
#np.c_ is a variation of that adds column vectors to each other, in first-to-last order.
#therefore, np.c_ [s1, s2, s3] means add the signals as columns to the matrix X!
X = np.c_[s1, s2, s3]

print("X = ", X)
print()

#this A matrix is mostly arbitrary. Its only purpose is to mix up the X matrix data.
A = np.array(([[0.1, 0.1, 0.1], [1, 2, 1.0], [1, 1, 2.0]]))

#np.dot(X, A.T) means take the dot product of X and A.T (the transpose of A)
X = np.dot (X, A.T)
print("X*A.T = ", X)
print()

#X.T means get the transpose matrix of X
X = X.T
print("X = X.T => ", X)
print()
```

# Code

- Whitening

- Calculation of W

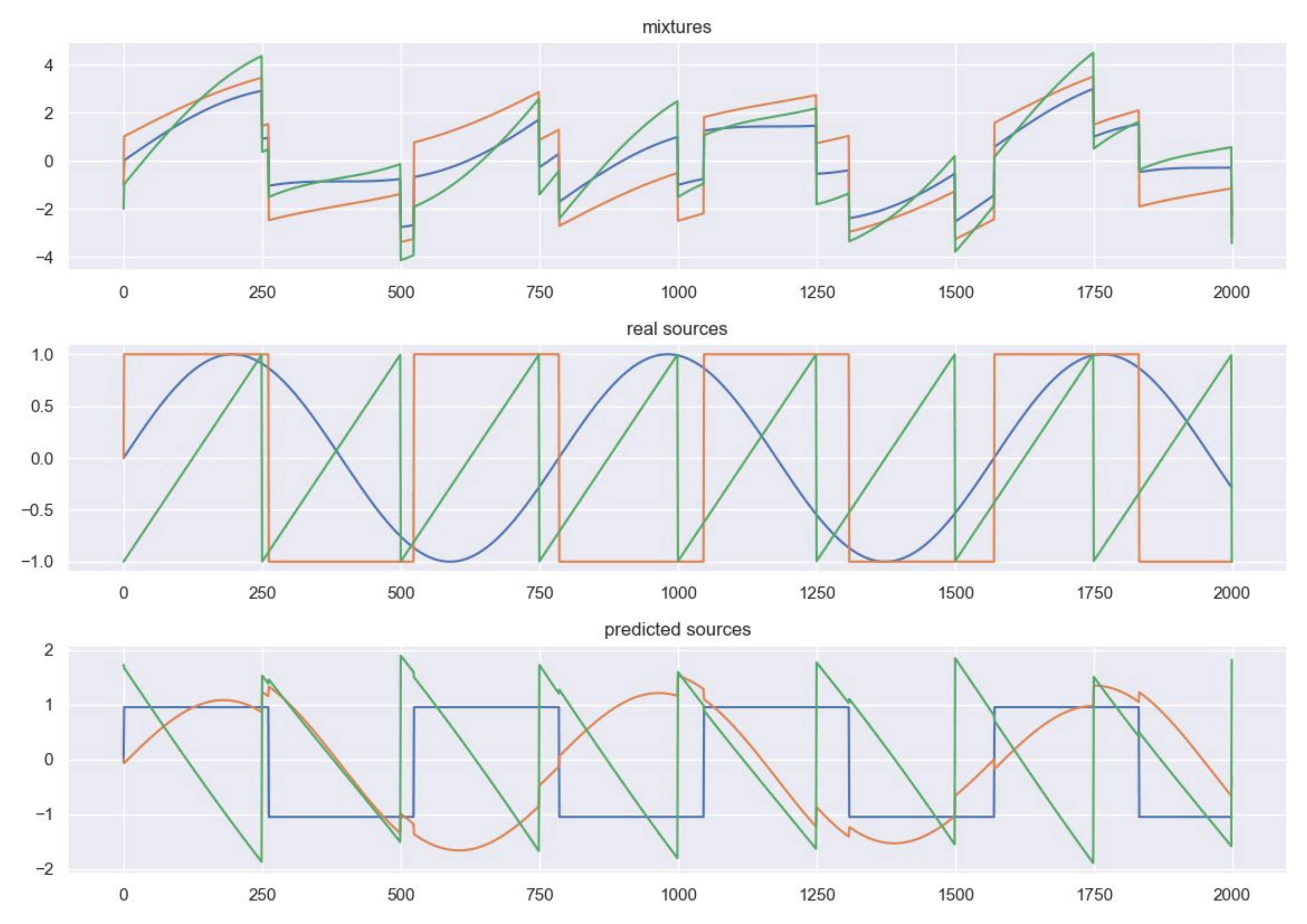
```python
def whitening(X):
    cov = np.cov(X)

    #return to d, E these values:
    #d will receive the eigenvalues in ascending order, each repeated according to its multiplicity
    #multiplicity of an eigenvalue is the number of times it appears as a root of
    #the characteristic polynomial (ex.: the polynomial whose roots are the eigenvalues of a matrix)
    #polynomial being an expression such as: P(x) = 2x^2 - 6x + 12

    #meanwhile, E receives

    d, E = np.linalg.eigh(cov)

    #D shall be a diagonal matrix of eigenvalues (every lambda is an eigenvalue of the covariance matrix)
    D = np.diag(d)

    D_inv = np.sqrt(np.linalg.inv(D))

    X_whiten = np.dot(E, np.dot(D_inv, np.dot(E.T, X)))

    return X_whiten

# update the de-mixing matrix W

def calculate_new_w(w, X):
    w_new = (X * g(np.dot(w.T, X))).mean(axis=1) - g_der(np.dot(w.T, X)).mean() * w

    w_new /= np.sqrt((w_new ** 2).sum() )

    return w_new
```

# Results

```
PS C:\Users\corey\Desktop\CSE 5350 Final> python ./CSE_5350_Final_Project_v5_x.py
Running... Please wait...

s1 =
 [ 0.          0.00800392  0.01600732 ... -0.27253687 -0.28022907
 -0.28790332]

s2 =
 [ 0.  1.  1. ... -1. -1. -1.]

s3 =
 [-1.        -0.991996 -0.983992 ...  0.983992  0.991996 -1.      ]

X =  [[ 0.          0.          -1.        ]
 [ 0.00800392  1.          -0.991996  ]
 [ 0.01600732  1.          -0.983992  ]
 ...
 [-0.27253687 -1.           0.983992  ]
 [-0.28022907 -1.           0.991996  ]
 [-0.28790332 -1.          -1.        ]]

X*A.T =  [[-1.00000000e-01 -1.00000000e+00 -2.00000000e+00]
 [ 1.60079185e-03  1.01600792e+00 -9.75988079e-01]
 [ 3.20153243e-03  1.03201532e+00 -9.51976672e-01]
 ...
 [-2.88544871e-02 -1.28854487e+00  6.95447125e-01]
 [-2.88233070e-02 -1.28823307e+00  7.03762928e-01]
 [-2.28790332e-01 -3.28790332e+00 -3.28790332e+00]]

X = X.T =>  [[-1.00000000e-01  1.60079185e-03  3.20153243e-03 ... -2.88544871e-02
  -2.88233070e-02 -2.28790332e-01]
 [-1.00000000e+00  1.01600792e+00  1.03201532e+00 ... -1.28854487e+00
  -1.28823307e+00 -3.28790332e+00]
 [-2.00000000e+00 -9.75988079e-01 -9.51976672e-01 ...  6.95447125e-01
   7.03762928e-01 -3.28790332e+00]]
```

# Results

- Graphs were plotted using Python Matplotlib library
- Three wave signals are generated
  - Sine wave
  - Square wave
  - Sawtooth wave
- Solved by ICA function
  - The three signals are re-separated



# Conclusions

- ❑ x Matrix was our signal matrix
- ❑ After mixing and then de-mixing, Signals switched spots
- ❑ ICA algorithm successful, but produced signals in an unpredictable order
- ❑ Based on initial mixing matrix, S.
- ❑ ***Why did this happen?***
  - "Even when the sources are not independent, ICA finds a space where they are maximally independent." This could hint that ICA's primary focus is not on preserving signal order, or column order in the resulting de-mixed signal matrix, but rather, it focuses on maximizing the clarity and independence of each individual signal, regardless of the order they were initially in.

- ❑ ***Another algorithm would be needed after ICA to label all independent signals in the correct manner.***

# Thank You!