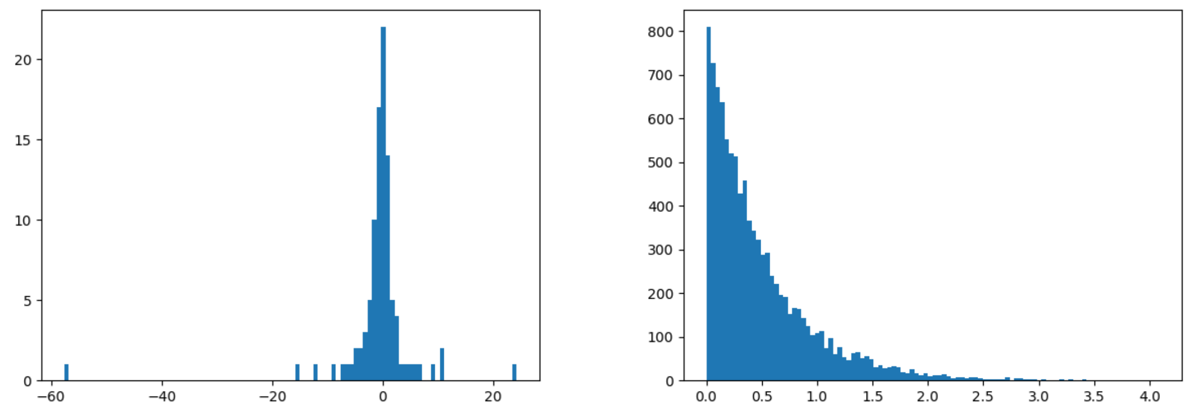
1 Inverse Transform Sampling [20 marks]

Inverse transform sampling provides a way to generate random numbers that follow a specific probability distribution, even if there isn't a direct method available for sampling from that distribution. Go through this page to understand the necessary ideas.

Complete the function `inv_transform`, which generates samples from a given probability distribution using uniform random samples from $[0, 1]$. The arguments to the function are:

- `distribution` - One out of `exponential` or `cauchy`.
- `num_samples` - The number of random samples to be generated from the given distribution.
- `kwargs` - A dictionary containing parameters for the given distribution. Check out `q1_cauchy.json` and `q1_exponential.json` for the exact format of `kwargs`.

The generated random numbers should be **rounded to** 4 **decimal places**. You also need to create a histogram of the generated samples, which will look as shown in Figure 1.



(a) Cauchy Distribution (b) Exponential Distribution

Figure 1: Generated Histograms

2 Principal Component Analysis (PCA) [60 marks]

PCA is a popular statistical dimensionality reduction technique in Machine Learning. It uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. Necessary formulas for conversion are given here.

You will be given a text file (`pca_data.csv`). This file contains N lines, each with D comma-separated values. Read the file (using `pandas.read_csv`) and generate an $N \times D$ matrix.

Steps for PCA

1. Standardize the $N \times D$ matrix (**Do not divide by standard deviation**).
2. Calculate the covariance matrix for the D dimensions in the matrix.
3. Calculate the eigenvalues and eigenvectors for the covariance matrix.
4. Sort eigenvalues (in decreasing order) and their corresponding eigenvectors.
5. Pick $K = 2$ ($K < D$) eigenvalues and form a matrix of eigenvectors.
6. Transform the original matrix and store the $N \times K$ transformed matrix in the same directory with the name `transform.csv`.
7. Plot the projected data (using a scatter plot from `matplotlib`) and save the plot to the current directory as `out.png`. While plotting, ensure the x and y axes have the same aspect, and show the values from $[-15, 15]$.

Output

1. Return the sorted eigenvalues (**rounded up to** 4 **decimal places**).
2. Save the transformed matrix in `transform.csv`.
3. Save a scatter plot in `out.png` (example shown in Figure 2).

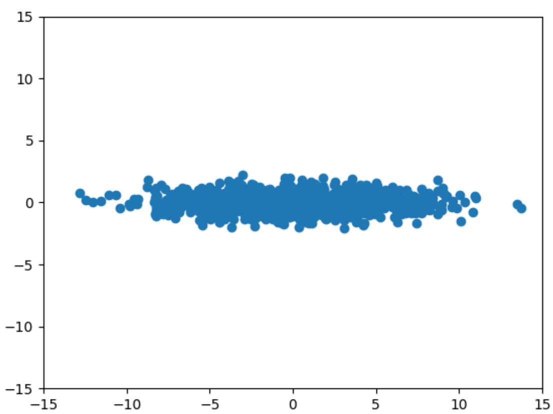


Figure 2: Scatter Plot of Transformed $N \times K$ Matrix

3 Curve Fitting [20 marks]

A magnet slides on a rough non-magnetic metallic inclined plane, which makes an angle with the horizontal. The equation models the displacement of the magnet with time

$$S(t) = v\left[t - \frac{(1 - e^{-kt})}{k}\right]$$

Given experimental data of $S(t)$ of a magnet with time t in file `data.csv`, your task is to find the value of constants v and k by fitting a curve.

t	$S(t)$
0.016	0.001
0.049	0.003
0.070	0.006
0.090	0.010
0.120	0.017
0.174	0.029
0.230	0.046
0.270	0.058
0.320	0.074
0.370	0.091

Table 1: Experimental Data

1. Read the data from `data.csv`.
2. Fit a curve over the given data using `scipy.optimize.curve_fit`.
3. Calculate the values of v and k (rounded up to 4 decimal places).
4. Plot the experimental data and the fitted curve, and save in `fit_curve.png`.

Code your solution in `q1.py`, in the given TODO blocks. An example fitted curve for the data above is shown in Figure 3 (replace the ? mark with the actual value of v and k).

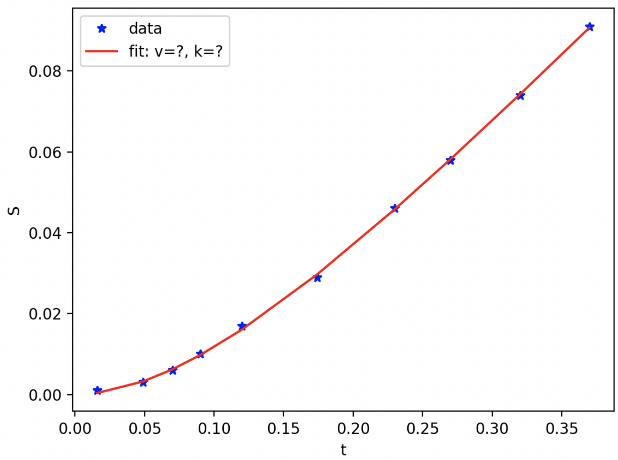


Figure 3: Fitted Curve for the given Experimental Data