Multivariate Data Analysis and Interfacing of Experimental Systems

A Project Report Submitted in Partial Fulfilment of the Requirements
for the Degree of

MASTER OF SCIENCE

in

School of Mathematics

by

Prashant Sharma
(Roll No. IMS19175)

to

SCHOOL OF MATHEMATICS
INDIAN INSTITUTE OF SCIENCE EDUCATION AND RESEARCH
THIRUVANANTHAPURAM - 695551, INDIA

November 2023

DECLARATION

I, **Prashant Sharma** (**Roll No: IMS19175**), hereby declare that, this report entitled **"Multivariate Data Analysis and Interfacing of Experimental Systems"** submitted to Indian Institute of Science Education and Research Thiruvananthapuram towards partial requirement of **Master of Science** in **School of Mathematics** is an original work carried out by me under the supervision of **Dr. Dharmatti Sheetal and Dr. Vinayak B. Kamble** and has not formed the basis for the award of any degree or diploma, in this or any other institution or university. I have sincerely tried to uphold the academic ethics and honesty. Whenever an external information or statement or result is used then, that have been duly acknowledged and cited.

Thiruvananthapuram - 695 551 **Prashant Sharma**

November 2023

CERTIFICATE

This is to certify that the work contained in this project report entitled **"Multivariate Data Analysis and Interfacing of Experimental Systems"** submitted by **Prashant Sharma** (**Roll No: IMS19175**) to Indian Institute of Science Education and Research Thiruvananthapuram towards partial requirement of **Master of Science** in **School of Mathematics** has been carried out by him under my supervision and that it has not been submitted elsewhere for the award of any degree.

Thiruvananthapuram - 695 551

November 2023

Dr. Dharmatti Sheetal

Dr. Vinayak B. Kamble

Project Supervisors

ABSTRACT

The execution of experiments involves and creates a large amount of data that needs to be acquired, organized and assessed to find a meaningful outcome of the experiments. The main aim of the project is to understand and create a Machine learning model to identify the gases present in a system based on information provided by the sensors. To begin with we started to learn Principal Component Analysis (PCA), a well-known technique in data analytics. Various methods and techniques are studied and have been implemented in the well-known data sets. A python-based library is generated to perform PCA. It was compared with the commercially available SKlearn library. Apart from this, to collect thermal expansion data of the samples, a device was built which can measure expansion in temperature range varying from room temperature to 900 K. The device needs to be optimized for collecting reliable data which is the work under progress. In future we would explore optimizing PCA method using machine learning algorithms on the data collected from the device.

Contents

List of Figures

Chapter 1

Principal Component Analysis

Principal Component Analysis is a dimensionality reduction method, it works on the principal components and change of basis. PCA retains the data in the direction of maximum variance and reduces the features which are no corelated. This reduced feature can be used to perform unsupervised clustering and classification. In Brief the algorithm to perform PCA is to computes mean of each feature, make data mean centered, calculate the Covariance of mean centered data and then find the eigen vectors and eigen values of the covariance matrix at the end Data is projected along the selected eigen vectors. Here selected eigenvectors means that top k number of highest eigenvectors associated with highest eigenvalues becomes the principal Components[1][2].

1.1 Covariance

Definition 1.1.1. Covariance can be defined as the measure of changes between two variables, covariance matrix is a symmetric matrix.

Remark 1.1.2. for a 3-dimensional data set (x, y, z), measure of the covariance between the x and y dimensions, the y and z dimensions, and the x and z dimensions. Measuring the covariance between x and x, or y and y, or z and z would give the variance of the x, y and z dimensions respectively.

Definition 1.1.2. Variance can be defined as measure of the deviation from the mean for points in one dimension

Variance and Covariance are the measure of the “spread” of a set of points around their center of mass i.e., mean. Covariance is a measure of how much each of the dimensions vary from the mean with respect to each other, it is measured between two dimensions and variance is the covariance between one dimension and itself.

1.1.1. Covariance Matrix

Definition 1.1.3. A covariance Matrix Represents Covariance between dimensions as a matrix.

Consider a 3-dimensional covariance Matrix C

$$C = \begin{pmatrix} \mathrm{Cov}(X,X) & \mathrm{Cov}(X,Y) & \mathrm{Cov}(X,Z) \\ \mathrm{Cov}(Y,X) & \mathrm{Cov}(Y,Y) & \mathrm{Cov}(Y,Z) \\ \mathrm{Cov}(Z,X) & \mathrm{Cov}(Z,Y) & \mathrm{Cov}(Z,Z) \end{pmatrix}$$

Here, Diagonal is the variances of x, y and z also cov(x,y) = cov(y,x) hence matrix is symmetrical about the diagonal for three-dimensional data we get 3x3 Covariance Matrix similarly for N-dimensional data will result in NxN covariance matrix. And all diagonal entries are the variance of that column.

Suppose we have n-dimensional data with features (each dimension or each column in a data set)

For Features or dimensions $\{1,2,\dots,j\}$ and each feature has n entries $\{1,2,\dots,i\}$ then, covariance matrix C can be represented as

$$C = \begin{pmatrix} \mathrm{cov}(1,1) & \mathrm{cov}(1,2) & \cdots & \mathrm{cov}(1,j) \\ \vdots & & \ddots & \vdots \\ \mathrm{cov}(i,1) & \mathrm{cov}(i,2) & \cdots & \mathrm{cov}(i,j) \end{pmatrix}$$

Covariance can be negative or positive, A positive value of covariance indicates both dimensions increases or decreases together, opposite to this a negative covariance indicates while one increases the other decreases or vice versa. If covariance is Zero it represents both the dimensions are independent of each other.

The covariance matrix is a square matrix that provides a measure of how much each variable in dataset varies with every other variable.

For some dataset with n feature (variables), the covariance matrix is an $n \times n$ matrix.

Let's assume we have m datapoints and n features, then dataset can be represented as a matrix X with dimensions $m \times n$. (Each column of X corresponds to a different feature, and each row corresponds to a different data point.)

The Covariance between two features *i* and *j* can be calculated as follows:

$$\mathrm{Cov}(X_i, X_j) = \frac{1}{\mathrm{m}-1} \sum_{i=1}^{\mathrm{m}} (X_{ki} - \bar{X}_i)(X_{kj} - \bar{X}_j)$$

Here,

X_i and X_j are the *i*-th and *j*-th features, respectively.

$X_{ki}$and X_{kj} are the values of the *i*-th and *j*-th features in the *k*-th data point.

$\bar{X}_i$ and $\bar{X}_j$ are the means of the *i*-th and *j*-th features, respectively.

The Covariance matrix C can be written as with each entry C_{ij}(covariance between feature *I* and *j*

$$C_{ij} = \mathrm{Cov}(X_i, X_j)$$

For simplicity standardized data matrix X with each column having mean zero, can be written as

$C = \frac{1}{m-1} X^T X$ (X^T is transpose matrix of X)

Or

$$\mathrm{Cov}(X_i, X_j) = \frac{1}{\mathrm{m}-1} \sum_{i=1}^{\mathrm{m}} (X_{ki} - \bar{X}_i)(X_{kj} - \bar{X}_j)$$

This can also be used, since in this work no external library used there for the equation () is used to calculate the covariance matrix in python code (Appendix II).

1.2 Eigenvalues and Eigenvectors

Eigenvalues reflect the entire amount of variation that can be explained by a specific principal component, which is always positive. If the eigenvalues are not larger than zero, something is amiss. Higher the value of eigenvalues it explains more the variation in components[3]. Here

eigenvalues and eigenvectors of covariance matrix is calculated which gives the Principal Components. There are as many principal components as the number of features on the dataset[4][5].

Definition 1.2.1 Eigenvalues

The eigenvalues of a matrix A are obtained by solving its characteristic equation

$$A - \lambda I = 0.$$

$$\lambda^n + C_{n-1}\lambda^{n-1} + \cdots + C_0 = 0$$

Polynomial equations like this one are difficult and time-consuming to solve for large values of n. Furthermore, numerical approaches for estimating roots of high-degree polynomial equations are difficult in this case no external library was used so to reduce the rounding mistake here numerical method (in this case power iteration method) was used in my code.

the method can be used only to find the eigenvalue of A that is largest in absolute value. this eigenvalue is the dominant eigenvalue of A.

Definition 1.2.2 Dominant Eigenvalue and Dominant Eigenvector

Let $\lambda_1, \lambda_2, \cdots, and\ \lambda_n$ be the eigenvalues of $n \times n$ matrix. A. is called the dominant eigenvalue of A if $|\lambda_1| > |\lambda_i|, \quad for\ i = 1,2,\ldots,n$

the eigenvectors corresponding to λ_1 are called dominant eigenvectors of A. Not every matrix has a dominant eigenvalue.

1.3 Power Iteration Method

The power method is used for estimating eigenvalues, like the Jacobi and Gauss-Seidel methods, is iterative. It begins by assuming that matrix A has a dominant eigenvalue and corresponding dominant eigenvectors. Then select an initial approximation x_0 to one dominant eigenvectors of A.

Initial approximation must be a nonzero vector in R_n

First iteration	$x_1 = Ax_0$
Second iteration	$x_2 = Ax_1$
Third iteration	$x_3 = Ax_2$
	$\vdots$
k-th iteration	$x_k = Ax_{k-1} = A(A^{k-1}x_0) = A^k x_0$

Theorem 1.2.1. Determining an Eigenvalue from an Eigenvector

If x is an eigenvector of a matrix A, then its corresponding eigenvalue is given by

$$\lambda = \frac{Ax \cdot x}{x \cdot x}$$

Also known as Rayleigh quotient.

Proof: Since x is an eigenvector of A, we know that

$$Ax = \lambda x$$

and we can write

$$\lambda = \frac{Ax \cdot x}{x \cdot x} = \frac{\lambda x \cdot x}{x \cdot x}$$

□

$$= \frac{A(x \cdot x)}{x \cdot x}$$

$$= \lambda$$

Same thing is used in python program to find the eigenvalues from the eigenvectors. The required condition for the algorithm to work is that the matrix should be Diagonalizable and have a Dominant eigenvalue[7].

Chapter 2

Implementation of PCA

By finding the eigenvalues and eigenvectors of the covariance matrix, we find that the eigenvectors with the largest eigenvalues correspond to the dimensions that have the strongest correlation in the dataset.

PCA is a useful statistical technique that has applications in: – fields such as face recognition and image compression – finding patterns in data of high dimension.

This study takes a deep dive into mathematics by combining the beautiful properties of Principal Component Analysis (PCA) with the accuracy of thermal expansion measurements. The creation of a custom PCA library, which is included in the MYPCA class, and the building of a Thermal Expansion Measuring Device are the essential components of this study. These mathematical works aim to clarify the differences and present novel viewpoints in the fields of experimental physics, numerical analysis, and linear algebra.

2.1 MYPCA class

The MYPCA class, a custom implementation of PCA based on the mathematical elegance of the Power Iteration method, provides the starting point for this study. In this section we go into the complexity of eigenvector and eigenvalue calculations, our goal is to demonstrate PCA's sensitivity to various numerical approaches. The MYPCA class provides a unique perspective on the mathematical principles that allow the derivation of principal components

2.1.1 PCA using Official Sklearn Library

This library was used as a reference to perform PCA, The Sklearn is a python library which has many enables to perform many machine-learning algorithm and is also used to perform PCA quickly. The library contains lot of datasets and other Machine learning algorithms. Here PCA class is used to perform PCA. Using official library its very easy to perform PCA on a given

dataset. For a given set of data with n-number of features (n-number of columns) and desired k number of features, given data and desired k features (number of Principal components) was passed in the PCA Class.

2.1.2 PCA using MYPCA class

MYPCA is a CLASS created in this project to perform PCA by passing any data and number of Principal components required just like in Official Sklearn library, but this was a local class so instead of importing it, for the development and update purpose it was kept in the same working python file. As it is a CLASS as many instances can be created for different dataset in the same python file. The difficult part of the algorithm in MYPCA was to calculate the eigenvectors and eigenvalues that is not perfect till now and other part that is contradiction to standard library (in this case Sklearn decomposition kit was used) is to standardize the data. In APPENDIX code is attached for the MYCLASS Library. For understanding Iris data from Sklearn is used here just for the explanation. Actual MYCLASS CLASS created in this project uses only numpy library. There is big difference from the actual result, while calculating eigenvalues and eigenvectors using MYCLASS.

```
In [12]: import matplotlib.pyplot as plt
         from sklearn import datasets
         import numpy as np

In [2]:  data = datasets.load_iris()
         X = data.data
         y = data.target

         X.shape

Out[2]: (150, 4)
```

Fig-1 loading iris dataset from Sklearn

Mean Centering or Normalizing the data

Before performing PCA normalization of data is required, that is for each column mean is calculated and that respective mean of each column is subtracted from Each element in that column.

```
In [3]:  mean = np.mean(X, axis = 0)

         mean_X = X- mean
         print("\nmean: ", mean)
         print("\nafter subtracting mean ", mean_X.shape)
```

```
mean:  [5.84333333 3.05733333 3.758      1.19933333]

after subtracting mean  (150, 4)
```

Fig-2 Mean Centering data

It is required to perform Normalization just to make sure that all the features are at the same scale, to ensure that the first Principal Component is in the direction of maximum variance mean centering is used earlier different normalization method was used in MYCLASS.

Calculating Covariance Matrix

As mentioned before, Covariance matrix was calculated by implementing the equation of covariance as a function in the MYPCA CLASS. Another way to calculate covariance using numpy, the numpy function expects each row to present a variable, and each column to represent a different observation. Before using numpy covariance function Transpose of mean centered data is needed.

```
In [4]:  cov = np.round(np.cov(mean_X.T), 8)

         print("\n Covariance matrix: \n", cov)
```

```
Covariance matrix:
[[ 0.68569351 -0.042434    1.27431544  0.51627069]
[-0.042434    0.18997942 -0.32965638 -0.12163937]
[ 1.27431544 -0.32965638  3.11627785  1.2956094 ]
[ 0.51627069 -0.12163937  1.2956094   0.58100626]]
```

Fig-3 Covariance calculation

Calculating Eigen Vectors and Eigenvalues

Covariance matrix is used to calculate the Eigenvalues and Eigenvectors of the matrix. Each eigenvector represents a direction of variance. The eigenvector corresponding to the largest eigenvalue will give the direction of maximum variance, and also called the first principal component and similarly eigenvector correspond to the second largest eigenvalue gives the direction of second largest variance and called as second principal component and same goes for the other eigenvalues. Therefore, sorting eigenvalues along with their corresponding eigenvector was needed. Each eigenvector is orthogonal to another eigenvector.

```
In [5]: eig_val, eig_vec = np.linalg.eig(cov)

        print("eigenvalues:\n", eig_val)
        print("\n eigenvectors:\n", eig_vec)

        eigenvalues:
         [4.22824171 0.24267075 0.0782095  0.02383508]

         eigenvectors:
         [[ 0.36138659 -0.65658876 -0.58202984  0.31548724]
         [-0.08452251 -0.73016144  0.5979108  -0.31972314]
         [ 0.85667061  0.17337266  0.07623604 -0.47983899]
         [ 0.3582892   0.07548102  0.54583149  0.75365739]]
```

Fig-4 Eigenvalues & Eigenvector Calculation

Calculate the Explained Variance and N-Components Selection

Top k eigen vectors explaining most of the data is selected, to figure out how many components are needed one can look at explained variance of each feature that is computed by dividing the eigenvalues by sum of all eigen values. For very high dimensional data k-components is selected so that total of explained variance is more than 0.95.

```
In [7]:
sum_eig_val = np.sum(eig_val)

explained_var = eig_val/sum_eig_val

cumsum_var = np.cumsum(explained_var)

print("Explained variance: ", explained_var)
print("\nCumulative variance: ", cumsum_var)


n_comp = 2
n_eig_vec = eig_vec[:,:n_comp]

print("\n first ", n_comp, " of the data:\n", n_eig_vec)
```

```
Explained variance:  [0.92461872 0.05306648 0.01710261 0.00521218]

Cumulative variance:  [0.92461872 0.97768521 0.99478782 1.        ]

 first  2  of the data:
 [[ 0.36138659 -0.65658876]
 [-0.08452251 -0.73016144]
 [ 0.85667061  0.17337266]
 [ 0.3582892   0.07548102]]
```

Fig-5 Explained variance and selecting first n components

Use Eigen Vectors to Transform Data

Dot product of data with the selected k-eigenvectors is used to get the projection of data in the direction of eigenvectors.

```
In [8]:
pca_X = np.dot(mean_X, n_eig_vec)

print("after PCA data shape: ",pca_X.shape)
```

```
after PCA data shape:  (150, 2)
```

Fig-6 Transforming Data

Here's a comparision among first two features of IRIS dataset, same dataset after subtracting means and first two principal compnents.

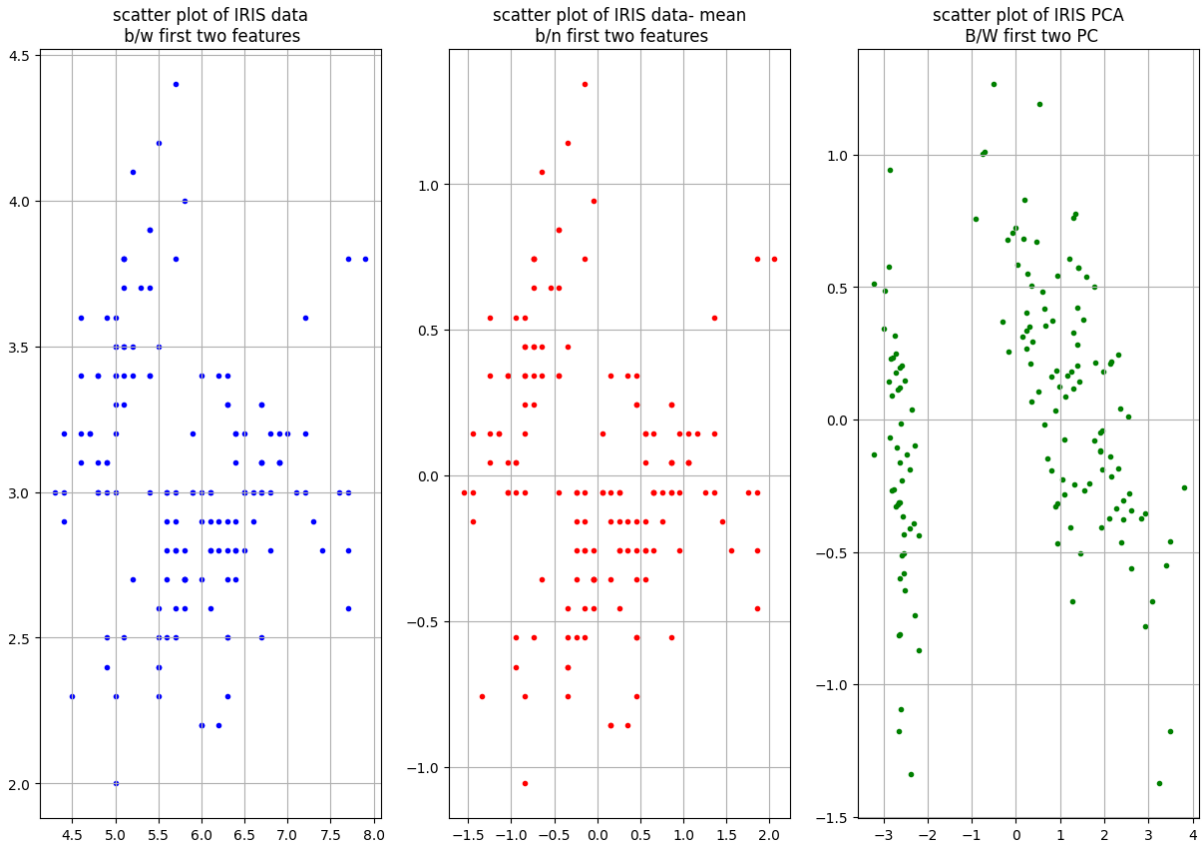


Fig-7 most left: raw data first two features, middle: first two features after mean centering, right: principle component-1 v/s principle component-2

Reconstructing Original data by Inverting PCA

Taking dot product of transpose of eigenvectors with transformed data original data is constructed.

Earlier mean was subtracted from the data while inverting PCA to original data addition of mean is required. As all the eigenvectors are orthogonal to each other the dot product of eigen vectors with itself gives the identity matrix

reconstruction after PCA

```
In [10]:  rev_X = np.dot(pca_X, n_eig_vec.T) + mean

          print(rev_X.shape)
```

```
(150, 4)
```

Fig-8 reconstructing dataset

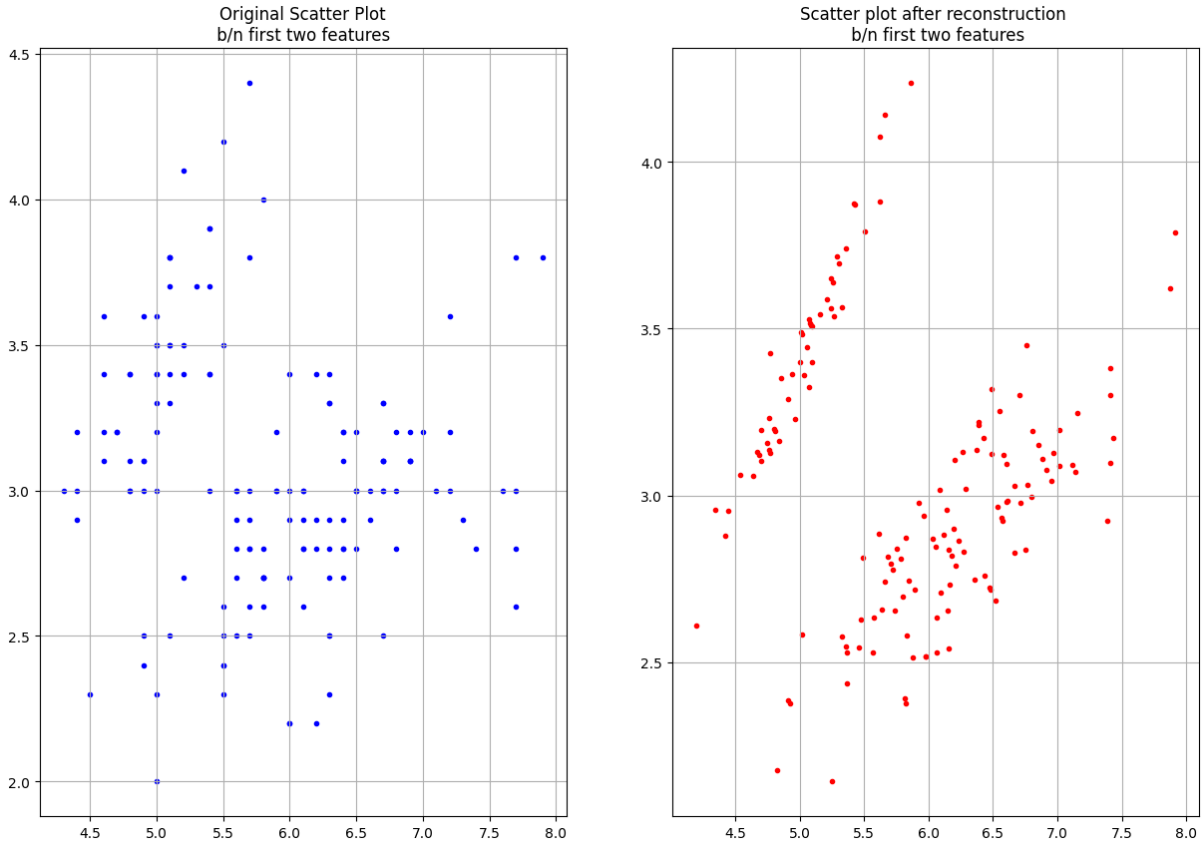


Fig-9 left: raw data feaeture-1 v/s feature-2, right: feaeture-1 v/s feature-2 after reconstruction using first two principle components

Quick Summary

Mean centered data, $X = X - mean\ of\ each\ feature$

After PCA, $X_{pca} = X * V$ (V is eigenvector matrix)

Reconstruction of the original data,

$$X_{reverse} = X_{pca} * V^T + mean$$

$$= (X * V) * V^T + mean$$

$$= X * (V * V^T) + mean \ = X * I + mean$$

$$X_{reverse} = X + mean$$

Chapter 3

Building Thermal Expansion Measurement System and its interfacing

3.1. The principle of measurement system

The coefficient of linear Thermal expansion (α) denotes the change in length of the given material when heated. It is a material property and the same may be written as,

$$\alpha = \frac{(L_f - L_0)/L_0}{(T_f - T_0)} \quad (1)$$

The quantity is of high significance for studying the thermal properties of materials for fundamental as well as applied areas.

There are a couple of methods reported for measurement of TEC. Those can be broadly classified as-

- Interferometry - light interreference based
- Dilatometry
  - Capacitance based
  - LVDT based

In case of Interferometric methods, the sample surface is shone with a monochromatic light beam and the interference fringes are observed from the rays reflected from the surface. Although this method has high precision, it is applicable to only limited range of α values and depends on the sample surface optical properties.

On the other hand, dilatometric methods, the change in length of the material is measured by means of a linear translation rod/arm assembly that changes the signal on the capacitance bridge or an LVDT respectively. Here we have developed a system based on this method using a Linear Variable Differential Transformer (LVDT) as linear transducer.

3.2. Instrumentation

The entire system is home built and the schematic diagram of the system is shown below in Fig 10.

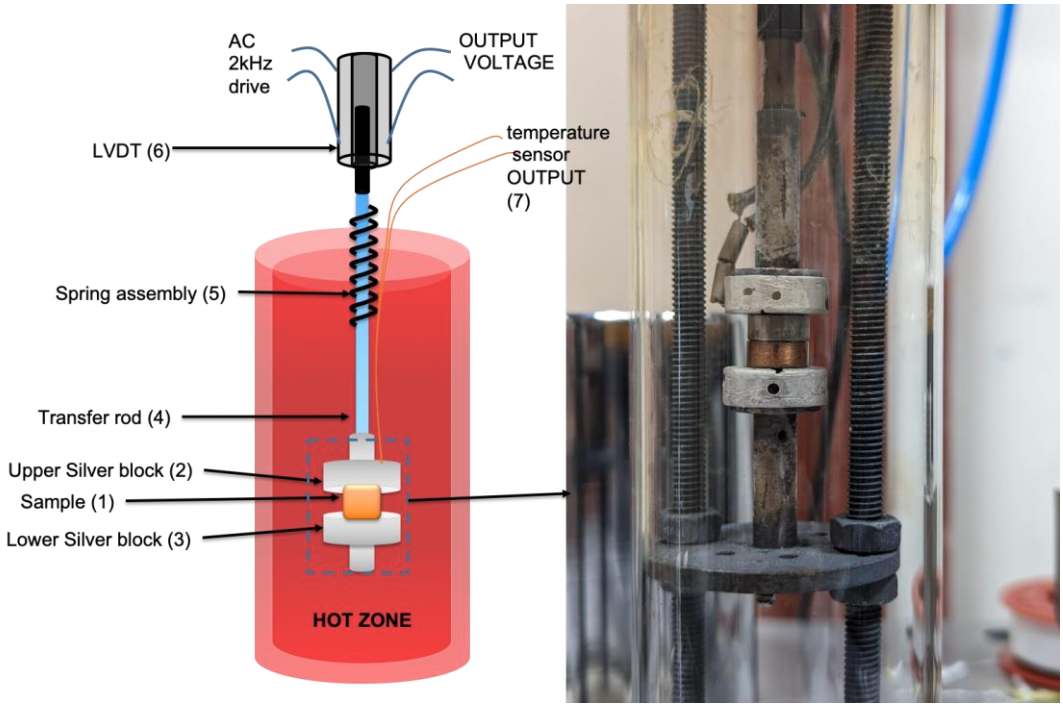


Figure 10: The schematic diagram of the home built Thermal expansion measurement system and its digital photograph.

The sample (1) to be measured is sandwiched between two silver metal blocks (2 and 3) for to hold in place, as well as silver helps in better temperature equilibration. The lower silver block is fixed while the upper silver block is mounted on a transfer rod/arm (4) which is made up of steel (not modified to Quartz). For this top silver block to always press against the sample, it is equipped with a spring assembly (5) that always pushes it against the sample. This ensures that any change is sample length is directly translated to the other end of the rod where the transducer is place. As mentioned above, LVDT is used here as a transducer as shown in the figure.

The enlarged view of sample holder assembly is also shown in Fig 1. The entire sample holder goes inside a vacuum chamber made of the quartz tube with suitable couplers and flanges. The Quartz tube is inserted into the furnace (hot zone) to raise the sample temperature uniformly. A k-type thermocouple is inserted in upper silver block to measure the sample temperature. The electronic components are place away from the hot zone.

If the sample is placed, it is expected to show a certain signal from the LVDT arm. If its temperature is raised and it expands the change is length of the sample caused the quartz rod to push the core of LVDT further inside producing a change in signal.

3.2.1 Design and Working of LVDT

An AC power source of 1.2 V at 2 kHz sine wave was used to power the LVDT, and a Keithley 2700 Multimeter cum data acquisition system was used to read its analog outputs.

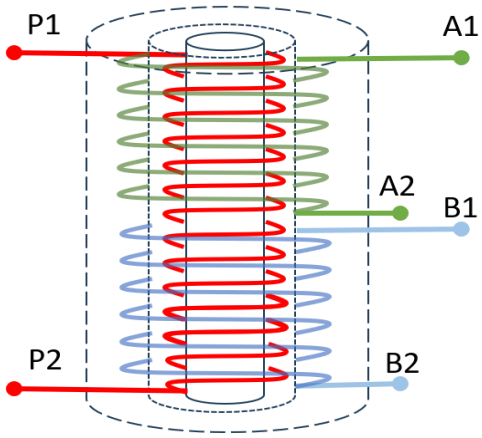


Fig 11: Circuit diagram of LVDT primary winding and Secondary winding.

A hollow cylinder of insulating material serves as its main component. This insulating cylinder has one main winding P and two secondary windings A and B looped around its circumference. In the middle of the insulating cylinder lies the main winding P, and on either side of it are two secondary windings, A and B, coiled in complete opposition to one another as shown in Fig 2. In other words, A and B are moving in opposite directions. A magnetic or

armature core is put within the insulating hollow cylinder and can move freely in both directions. Attaching the soft iron core to the object under study allows for the measurement of displacement. Nickel is commonly used in softcore because of its great sensitivity. [9]

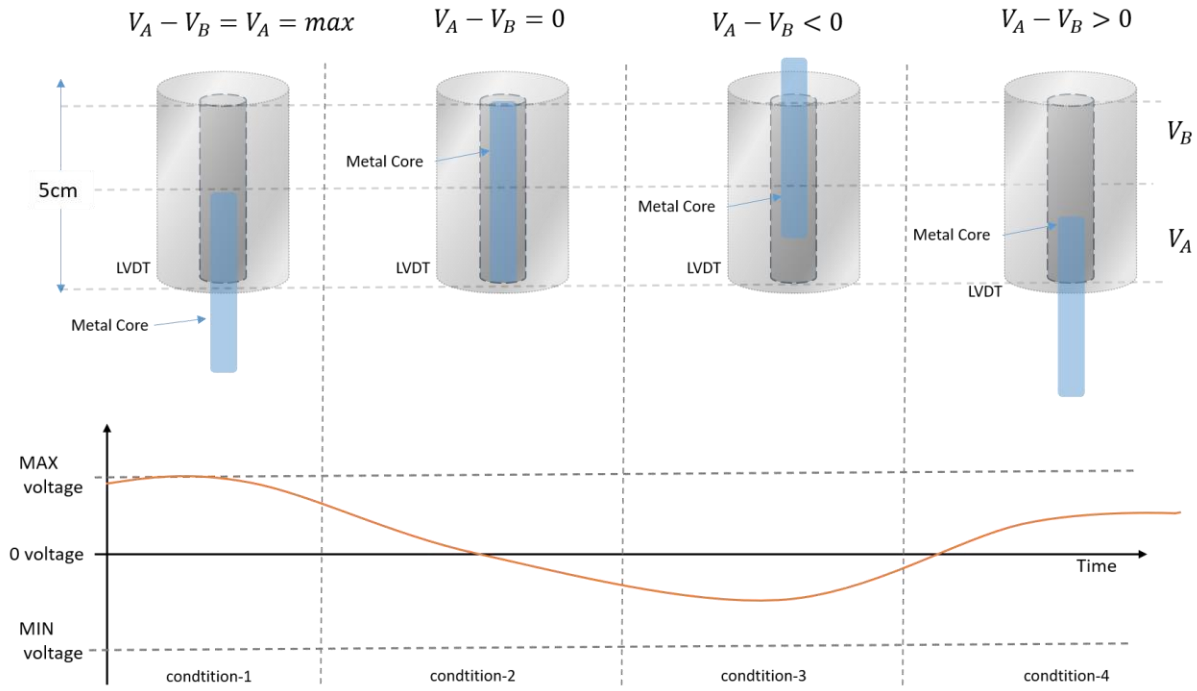


Figure 12: Different placements of metal core inside LVDT and corresponding voltage signal being generated.

From Faraday's law of electromagnetic induction, an EMF is produced in the secondary winding. When the primary winding is supplied with an alternating current, a magnetic flux is generated that travels through both A and B. Primary winding flux is proportional to secondary winding conductor count. The primary coil is powered using an AC signal (here we used 2 kHz and 1.2 V). The magnetic induction into the secondary develops due to magnetic core placed in between (like a transformer). When the core of LVDT may slide to shifts to top, bottom or to the null position. There will be an increase or decrease in the output difference. In this case, the LVDT's output voltage is a linear function of core displacement up to a certain point (5 mm from the LVDT's null position limit). This graph displays the variation in output voltage versus displacement. (See Fig 12).

Measuring TEC requires monitoring the output voltage of the LVDT that changes when the temperature of the sample under consideration varies. With increment in temperature, material expands and so the connected shaft to the metal core inside LVDT moves, which gives change in voltage proportional to the change in position of the metal core. Therefore, LVDTs can be consider as length or position sensors that detect changes in length or position.

In order to measure the TEC a change in length of material with change of temperature is to be measured, and initial length of the sample should be known. With a good sensitive LVDT even a small expansion (sub mm) can also be measured accurately. Thus, the final system measurement should give the plot between change in length vs temperature. Change in length becomes change in Voltage by using LVDT's sensitivity factor. (typically given in mV/mm)

Let V_A and V_B are the voltages produced by the A and B coil respectively. Since A and B are connected in the opposite directions, they must be connected in series to create a single voltage, with a phase difference of 180 degrees. Because of this, the secondary coil's output will be the difference between the two voltages produced by the device.

$$V = V_A - V_B \tag{2}$$

For small displacements the secondary coil's voltage output is linear as seen in Fig 12. There are no discrete steps in the voltage output, and the resolution is more a function of the testing apparatus than the transducer itself. Since the output voltage is large and easily measurable, no further intermediary amplifiers are required. The transducer can withstand significant vibration and stress because of its high sensitivity. Most importantly, the measuring system is insensitive to changes in temperature and suffers no loss due to friction. This attribute is necessary because of its proximity to a high-temperature furnace.

3.3. Calibration and Measurement

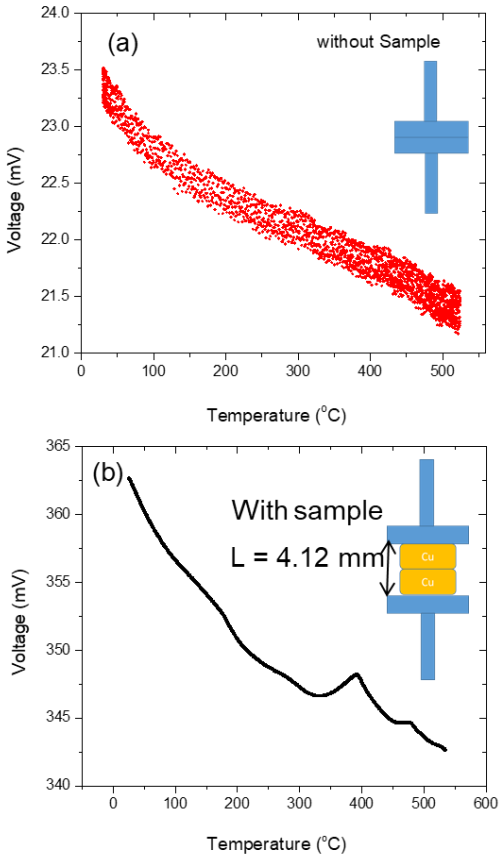


Fig. 13 (a) Without and (b) With sample thermal expansion measurement data. Two Cu samples in stack (each with L = 4.12mm) were used as sample.

The preliminary measurements performed with the system are shown in Fig 3. The system background data was measured without any sample and shown in Fig 3(a). because of the component used for sample holder there were non-zero background signal measured. Because of its low value there is a significant noise in the background. Similarly, the data was also measured by placing a sufficiently long Cu sample(s) between the two silver holders. This yielded about one order of magnitude large change in the voltage. Cu is used here for standardization for its known and high value of TEC.

Nevertheless, there are several non-monotonous changes in the voltage that are not expected. After a few modifications in the materials used and their arrangements we could

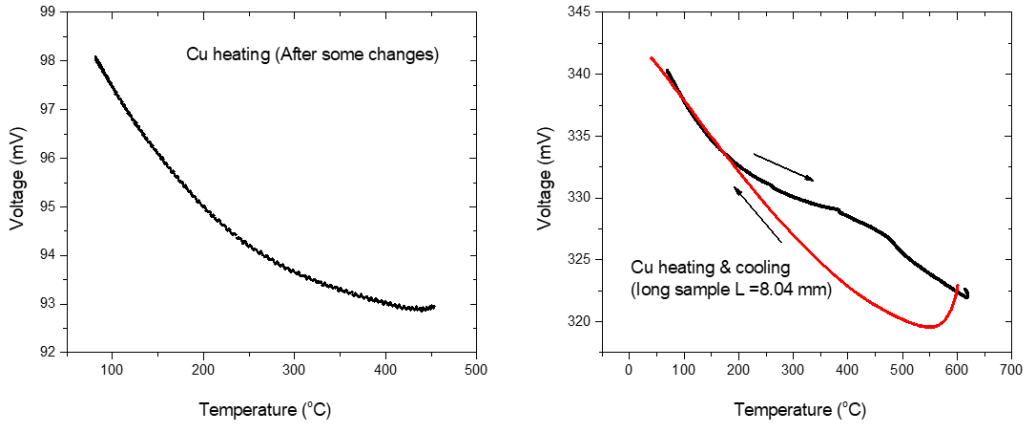


Fig 14: Measurements performed on the (a) same Cu sample, L= 4.12mm and (b) longer sample (8.04 mm length Cu sample)

Further, optimization of the system needs dynamic measurements. Also, the final data is to be represented as $\Delta L/L$. Here, copper has very high thermal expansion coefficient. However, for insulating samples like ceramics etc. the TEC value is low and hence the is of the order of background. In that case, it will be challenging to analyze the result when signal to noise ratio is low.

Chapter 4

Results

4.1 Comparison Between MYPCA v/s official PCA for different datasets

Iris data and digits data was taken from Sklearn and compared with our created library MYPCA results. Results were not the same.

4.1.1 Random 4*5 matrix

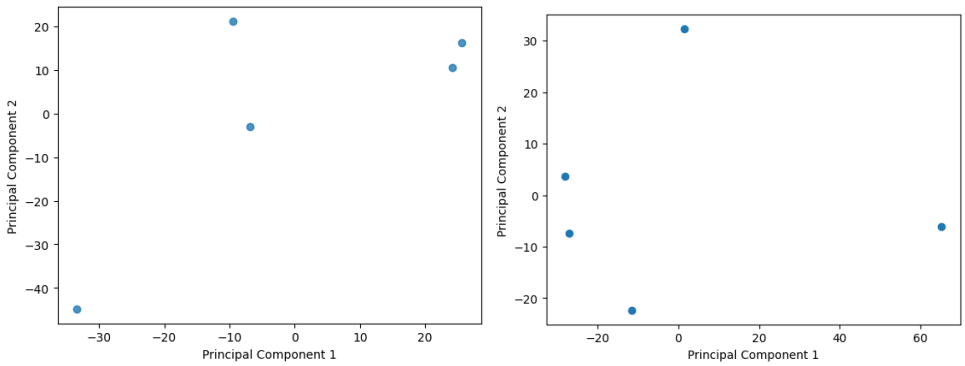


Fig-15 random matrix dataset Left: using MYPCA, right using Sklearn library

4.1.2 IRIS dataset

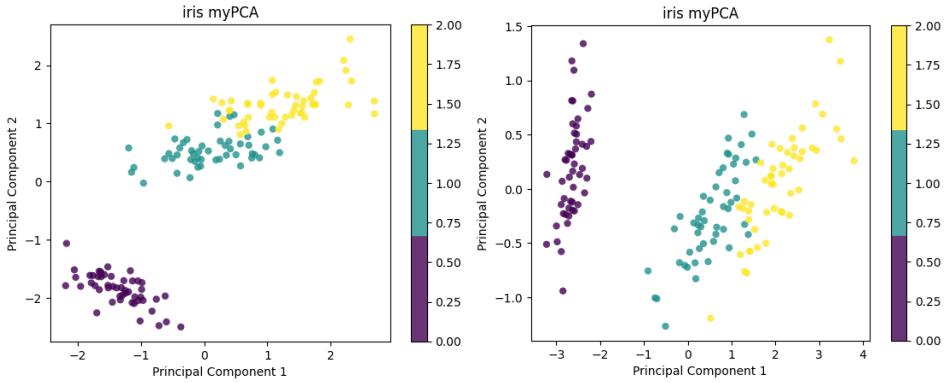


Fig-16 IRIS dataset Left: using MYPCA, right using Sklearn library

4.1.3 Digits Dataset

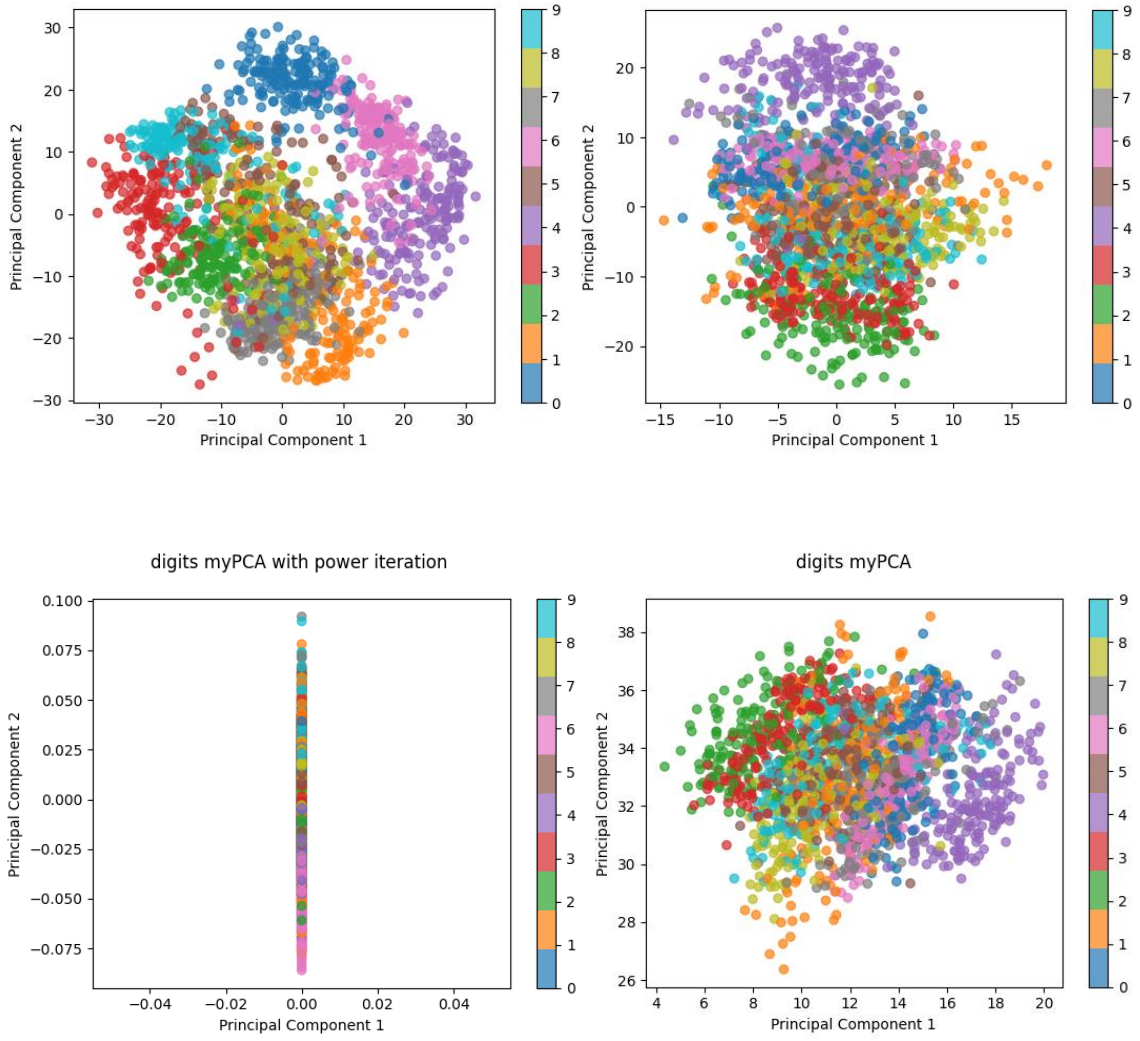


Fig-17 Digits dataset Left-Top : using Sklearn PCA, right-top using Sklearn library Left-bottom: using MYPCA with power iteration, right-bottom: reconstructed Data

4.2 For Thermal Expansion Temperature v/s Voltage

Firstly, expansion in the device itself was measured for the calibration. Without sample.

After that samples with length l= 4.12mm, 8.08mm was used to measure the expansion.

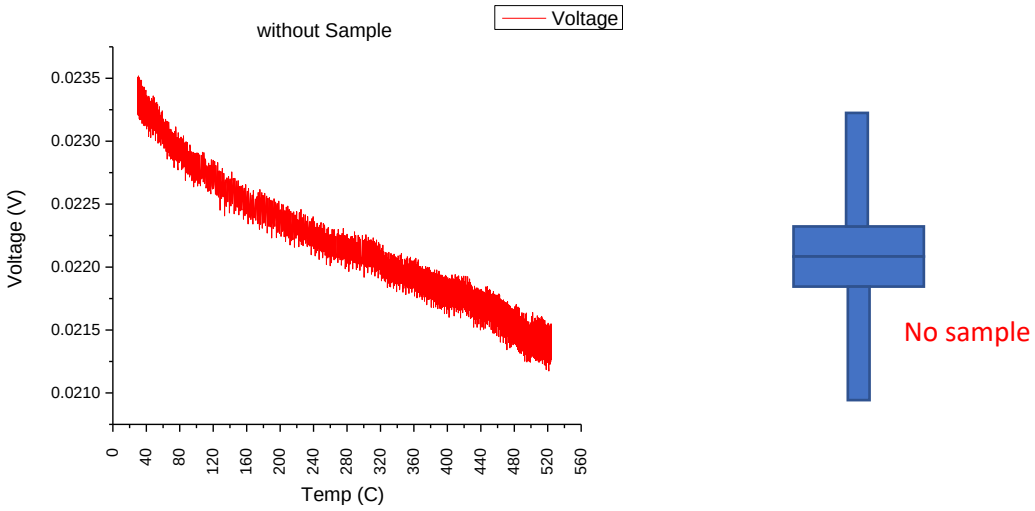


Fig 18: without sample thermal expansion measurement

Conclusion and Future work

MYPCA library was able to perform PCA for IRIS and Digit data set without using any external library. Although methods to calculate eigenvectors using power iteration method was giving values far away from the actual values, so some other methods to find the eigenvector is needed. Results were not same in comparison with Sklearn, more optimization may improve the result. All the aspects of Principal components were analyzed and compared which results in more work is needed for a production level Library. For future work target will be to find other methods such as autoencoders to reduce the dimension of dataset and get some useful result.

Thermal expansion Device needs better calibration and some change in design is also needed. Extra noise at low voltage from the LVDT can be cancel out by using some machine learning model. Perhaps, outputs from the device confirms the reliability of the constructs. Calibration position and quartz tube structure to reduce the expansion in device itself, all these integration of these components in experimental set-up can improve their usability and efficiency.

Reference

[1] Wilmott, Paul. "Machine learning: an applied mathematics introduction." Machine Learning and the City: Applications in Architecture and Urban Design (2022): 217-248.

[2] James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. An introduction to statistical learning. Vol. 112. New York: springer, 2013.

[3] Prodi, Cari, and Try Out. "Eigenvalues and eigenvectors."

[4] Paul, Liton Chandra, Abdulla Al Suman, and Nahid Sultan. "Methodological analysis of principal component analysis (PCA) method." International Journal of Computational Engineering & Management 16, no. 2 (2013): 32-38.

[5]Sonawane, Sachin Ashok, and M. L. Kulkarni. "Optimization of machining parameters of WEDM for Nimonic-75 alloy using principal component analysis integrated with Taguchi method." journal of king saud university-Engineering sciences 30, no. 3 (2018): 250-258.

[6] Mishra, Sidharth Prasad, Uttam Sarkar, Subhash Taraphder, Sanjay Datta, D. Swain, Reshma Saikhom, Sasmita Panda, and Menalsh Laishram. "Multivariate statistical data analysis-principal component analysis (PCA)." International Journal of Livestock Research 7, no. 5 (2017): 60-78.

[7] Sudadama, Muhsang. “Advanced Linear Algebra (Third Edition) by Steven Roman.” universitasnegerimalang, November 11, 2023.

[8] James, J. D., J. A. Spittle, S. G. R. Brown, and R. W. Evans. "A review of measurement techniques for the thermal expansion coefficient of metals and alloys at elevated temperatures." Measurement science and technology 12, no. 3 (2001): R1.

[9] Krishnan, Rappal Sangameswara, Ramachandran Srinivasan, and S. Devanarayanan. Thermal expansion of crystals: international series in the science of the solid state. Elsevier, 2013.

[10] Touloukian, Yeram Sarkis, R. K. Kirby, R. E. Taylor, and P. D. Desai. "Thermal expansion: metallic elements and alloys." (1975).

APPENDIX-I

MYPCA, this class has two versions first version uses numpy to calculate every other required mathematical tool such as mean, standard deviation, covariance, eigenvalues, eigenvectors and projection. And there is another version also which has all mathematical steps hard coded.

```
class MYPCA:
    def __init__(self, n_components):
        self.n_components = n_components
        self.components = None
        self.mean = None
        self.cov = None
        self.eigval = None
        self.eigvec = None
        self.components = None


    def fit(self, X):

        self.mean = np.mean(X, axis=0)
        X = X - self.mean

        self.cov = np.cov(X.T)

        # eigenvalues, eigenvectors
        eigenvalues, eigenvectors = np.linalg.eig(self.cov)

        # sort eigenvectors
        eigenvectors = eigenvectors.T
        idxs = np.argsort(eigenvalues)[::-1]
        eigenvalues = eigenvalues[idxs]
        eigenvectors = eigenvectors[idxs]

        self.eigval = eigenvalues
        self.eigvec = eigenvectors

        # store first n eigenvectors
        self.components = eigenvectors[0 : self.n_components]
```

```
    def transform(self, X):
        # project data
        X = X - self.mean
        return np.dot(X, self.components.T)
```

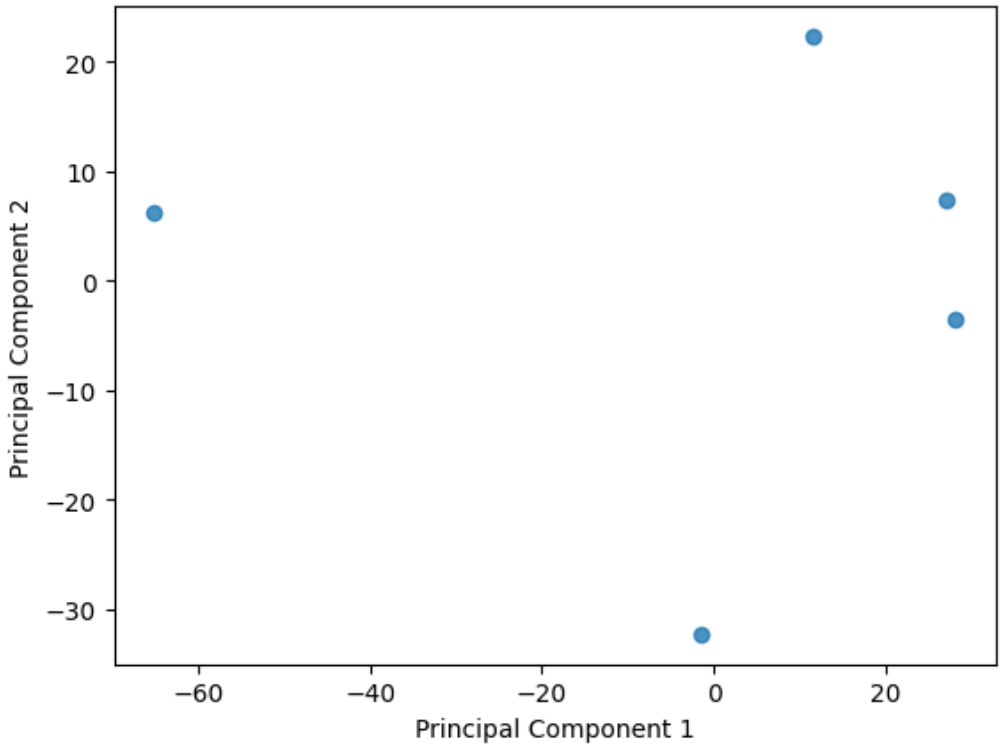
APPENDIX-II

Here is how to use MYPCA class

```
    A =   np.array([
        [4, 21, 3, 13],
        [1, 31, 21, 45],
        [5, 81, 73, 6],
        [9, 10, 12, 11],
        [42, 23, 41, 121]
# Project the data onto the 2 primary principal components
pca = MYPCA(2)
pca.fit(A)
A_projected = pca.transform(A)

print("Shape of A:", A.shape)
print("Shape of transformed A:", A_projected.shape)

print("\ncov\n", pca.cov)

print("\n eigenvalues: \n", pca.eigval)
print("\n eigenvectors: \n", pca.eigvec)

print("\n selected eig vectors  \n", pca.components.T)

print("\n pca transformed\n", A_projected)    ])
```

```
Shape of A: (5, 4)
Shape of transformed A: (5, 2)

cov
 [[ 285.7  -112.3    99.5   -82.85]
 [-112.3   770.2   677.5   -87.1 ]
 [  99.5   677.5   776.   -141.  ]
 [ -82.85  -87.1  -141.    245.3 ]]

 eigenvalues:
 [1471.86142224  412.11559893  180.26671057   12.95626825]

 eigenvectors:
 [[-0.00231747 -0.69681067 -0.70524078  0.13070944]
 [-0.79249023  0.34488353 -0.25814286  0.43171385]
 [-0.36301016  0.09152712 -0.25450239 -0.89166975]
 [ 0.49007906  0.62220015 -0.60928338  0.03825249]]

 selected eig vectors
 [[-0.00231747 -0.79249023]
 [-0.69681067  0.34488353]
 [-0.70524078 -0.25814286]
 [ 0.13070944  0.43171385]]

 pca transformed
 [[ 26.98647296   7.3611572 ]
 [ 11.51368685  22.35573476]
 [-65.10630554   6.16968146]
 [ 28.03121711  -3.58172624]
 [ -1.42507138 -32.30484718]]
```

```
a1 = A_projected[:, 0]
a2 = A_projected[:, 1]

plt.scatter(a1, a2, alpha=0.8,)


plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()
```


Principal Component 2
20
10
0
−10
−20
−30
−60
−40
−20
0
20
Principal Component 1