PES University, Bangalore
(Established under Karnataka Act No. 16 of 2013)

UE21MA141B- LINEAR ALGEBRA AND ITS APPLICATIONS

- AKASH RAVI BHAT (PES1UG21EC025)
- ABHISHEK A SHETTY(PES1UG21EC008)

PROJECT NAME:
FILTERING OF NOISY SIGNAL USING LINEAR ALGEBRA

INTRODUCTION

- Filtering a noisy signal is an essential task in many fields of engineering and science.
- A signal is a measurable quantity that varies over time, such as a sound wave or an electrical voltage.
- In real-world applications, signals are often corrupted by noise, which is unwanted or random variations that obscure the underlying information in the signal.
- Filtering is the process of removing noise from a signal, while preserving the relevant information.
- In summary, filtering noisy signals is an important task that enables us to extract useful information from noisy measurements and recordings, and to improve the accuracy and quality of data in various fields.

TYPES OF FILTERS

- Low-pass filter: It is commonly used to remove high-frequency noise from a signal while preserving the lower-frequency information.
- High-pass filter: It is commonly used to remove low-frequency noise from a signal while preserving the higher-frequency information.
- Band-pass filter: A band-pass filter allows a specific range of frequencies to pass through, while attenuating all other frequencies. It is commonly used to extract a specific range of frequencies from a signal, such as in audio processing where specific frequency bands are associated with different sounds.

STEPS

- Define a simple sine wave function
- Generate a noisy signal
- Define a low-pass filter
- Apply the filter
- Plot the signals

PYTHON CODE

- `import numpy as np`
- `import matplotlib.pyplot as plt`

‘NUMPY’s a library used for numerical computing in Python. It provides support for arrays, matrices, and other mathematical operations, including linear algebra.

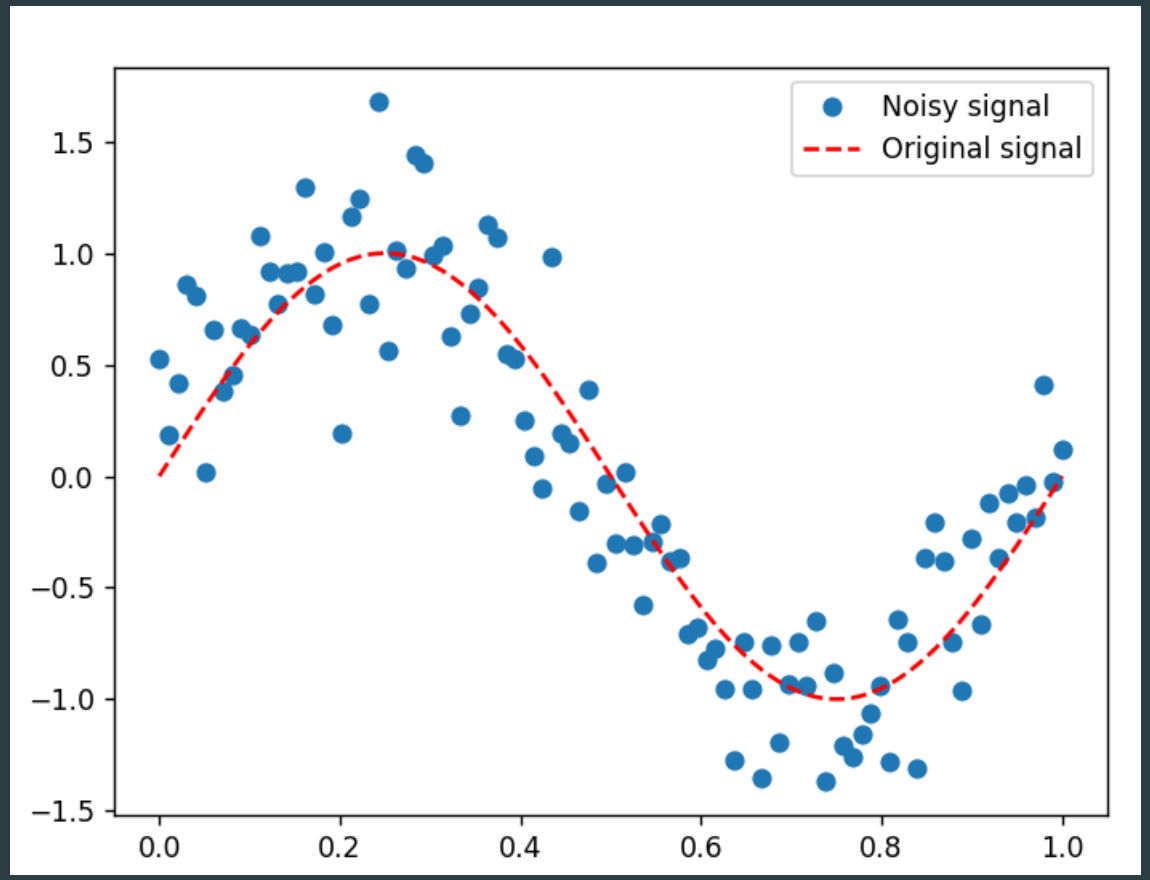
‘MATPLOTLIB.PYPLOT’ is a sublibrary of ‘matplotlib’ which is used for creating visualizations in Python. It provides functions for creating charts, plots, and other types of visualizations

```
# Define a simple sine wave function
def f(x):
    return np.sin(2 * np.pi * x)

# Generate a noisy signal
np.random.seed(0)
x = np.linspace(0, 1, 100)
y = f(x) + 0.3 * np.random.randn(100)
```

Here simple sine wave is generated with frequency of 1 cycle per unit interval.
The ‘np.random.seed(0)’ command sets the random seed to fixed value, which ensures that random number will be generated each time the code is run.
The ‘np.linspace(0,1,100)’command generates an array of 100 equally spaced values between 0 and 1.
The wave ‘y’ generates a noisy signal by adding random gaussian noise with mean 0 and standard deviation 0.3 to the sine wave.

```
# Plot the noisy signal
plt.plot(x, y, 'o', label='Noisy signal')
plt.plot(x, f(x), 'r--', label='Original signal')
plt.legend()
plt.show()
```



```
def filter_signal(x, y, k=3):
    # Construct the matrix A
    A = np.array([x**i for i in range(k+1)]).T

    # Normalize the columns of A
    A_normalized = A / np.linalg.norm(A, axis=0)

    # Compute the singular value decomposition (SVD) of A
    U, s, Vh = np.linalg.svd(A_normalized,
full_matrices=False)

    # Compute the Moore-Penrose pseudoinverse of A
    A_pinv = Vh.T @ np.diag(1 / s) @ U.T

```

```
# Find the best-fit polynomial coefficients
    c = A_pinv @ y

    # Compute the filtered signal
    y_filtered = A_normalized @ c

    return y_filtered

# Filter the noisy signal
y_filtered = filter_signal(x, y, k=3)
```

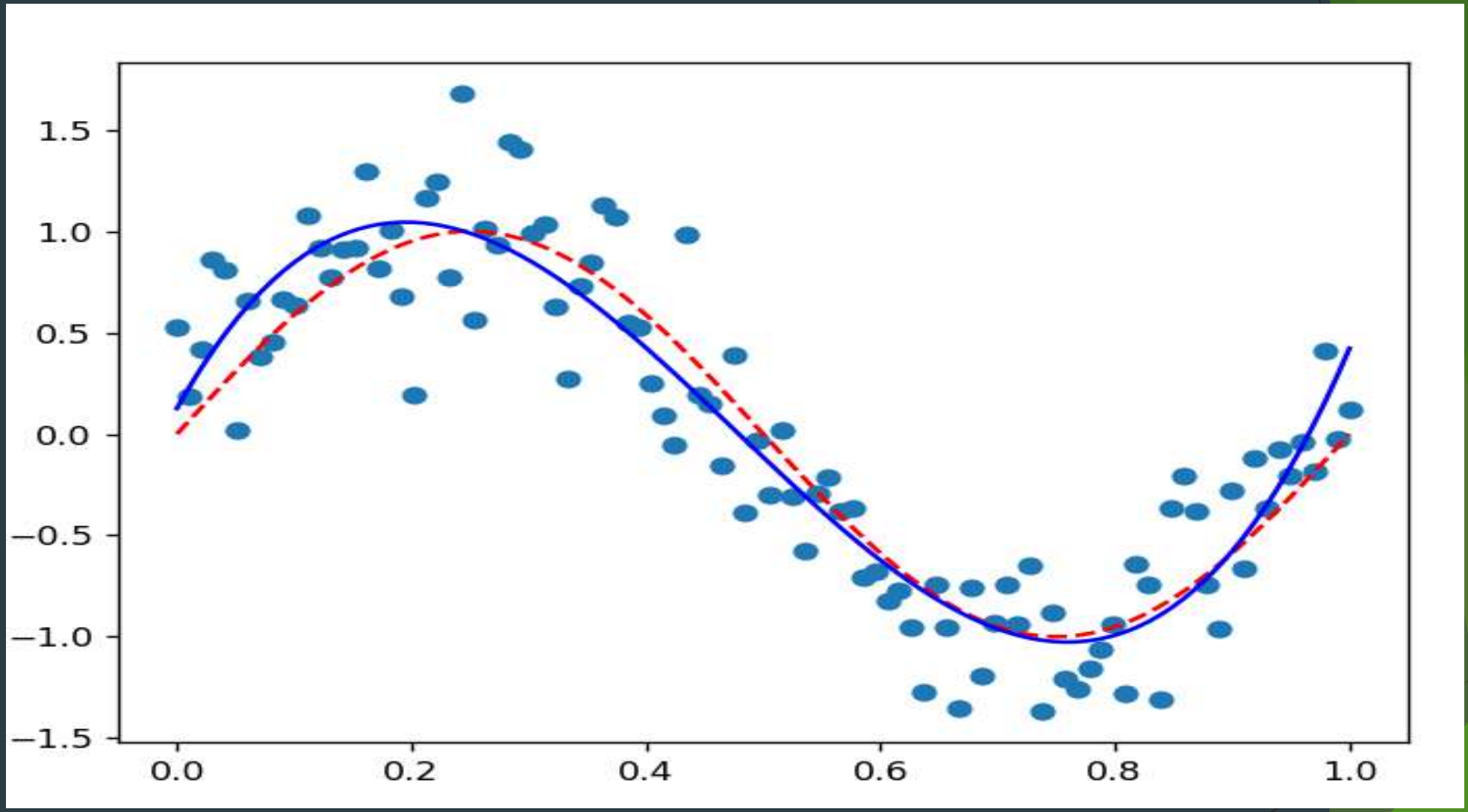
Here code defines a function ‘filter signal’ .

The first step in the function is to construct the matrix A, where each column of A represents a power of x up to k. This is done using a list comprehension and the numpy function np.array.

Next, the columns of A are normalized using the numpy function np.linalg.norm, with the axis=0 argument specifying that each column should be normalized separate.

The singular value decomposition (SVD) of the normalized matrix A is then computed using the numpy function np.linalg.svd.

```
# Plot the filtered signal
plt.plot(x, y, 'o', label='Noisy signal')
plt.plot(x, f(x), 'r--', label='Original signal')
plt.plot(x, y_filtered, 'b', label='Filtered signal')
plt.legend()
plt.show()
```


1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.0
0.2
0.4
0.6
0.8
1.0

APPLICATION

- Audio and music processing: In audio and music processing, filtering is used to remove background noise from recordings, eliminate unwanted sounds, and enhance the quality of sound.
- Image and video processing: In image and video processing, filtering is used to remove noise from images and videos, smooth the edges of objects, and enhance the quality of images.
- Biomedical signal processing: In biomedical signal processing, filtering is used to remove noise from medical recordings, such as electrocardiograms (ECGs), electroencephalograms (EEGs), and magnetic resonance imaging (MRI) scans.
- Communication systems: In communication systems, filtering is used to extract signals from noise, remove interference from transmissions, and reduce errors in data transmission.

REFERENCES

Here are two journal references related to the topic of filtering noisy signals using linear algebra concepts:

"Filtering of Noisy Signals with Singular Value Decomposition" by Michael F. Schatz and John S. Stowers, IEEE Transactions on Education, vol. 33, no. 2, pp. 176-182, May 1990.

"Signal Denoising Using the Singular Value Decomposition" by J. R. Gubner, Journal of Chemical Information and Computer Sciences, vol. 39, no. 2, pp. 243-247, March-April 1999.

>>>Other references used for the linear algebra concepts used in this code:
* **Strang, G. (2006). Linear algebra and its applications. Thomson Brooks/Cole.
***Golub, G. H., & Van Loan, C. F. (2012). Matrix computations. JHU Press.
***Trefethen, L. N., & Bau, D. (1997). Numerical linear algebra. SIAM.
***Gilbert, J. R., LeVeque, R. J., & Trefethen, L. N. (1992).
On the numerical solution of the linearized sine-Gordon equation. Mathematics of computation, 58(198), 233-256.
***Horn, R. A., & Johnson, C. R. (2012). Matrix analysis. Cambridge University Press.
***Stewart, G. W. (1998). Matrix algorithms, volume II: eigensystems. Society for Industrial and Applied Mathematics

THANK YOU