Exploring Eigenvalues and Eigenvectors: Calculation and Property Verification

A Beginner's Guide to NumPy-Wrap up

In this notebook we will learn how to calculate eigenvalues and eigenvectors for a given square matrix using NumPy.

I have also attempted to verify the fundamental linear algebra property that **each eigenvector of a matrix, when multiplied by the matrix, results in a vector that is a scalar multiple of itself (scaled version)** , expressed as:

A v = λ v

where (A) is a square matrix, (v) is an eigenvector, and (λ) is the corresponding eigenvalue.

Define and Calculate Eigenvalues and Eigenvectors

In [2]:
```
# Define the matrix A
import numpy as np
A = np.array([[4, 1],
              [2, 3]])

# Calculate eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(A)

print("Eigenvalues:", eigenvalues)
print("Eigenvectors:\n", eigenvectors)
```

```
Eigenvalues: [5. 2.]
Eigenvectors:
 [[ 0.70710678 -0.4472136 ]
 [ 0.70710678  0.89442719]]
```

Verifying the Eigenvalue-Eigenvector Relationship

For each eigenpair obtained from the matrix (A), we will verify the relationship (A v = λ v). This involves performing the following computations for each eigenvector (v):

- Calculate (A v) (matrix-vector multiplication).
- Calculate (λ v) (scalar-vector multiplication).
- Compare the results to confirm the relationship holds.

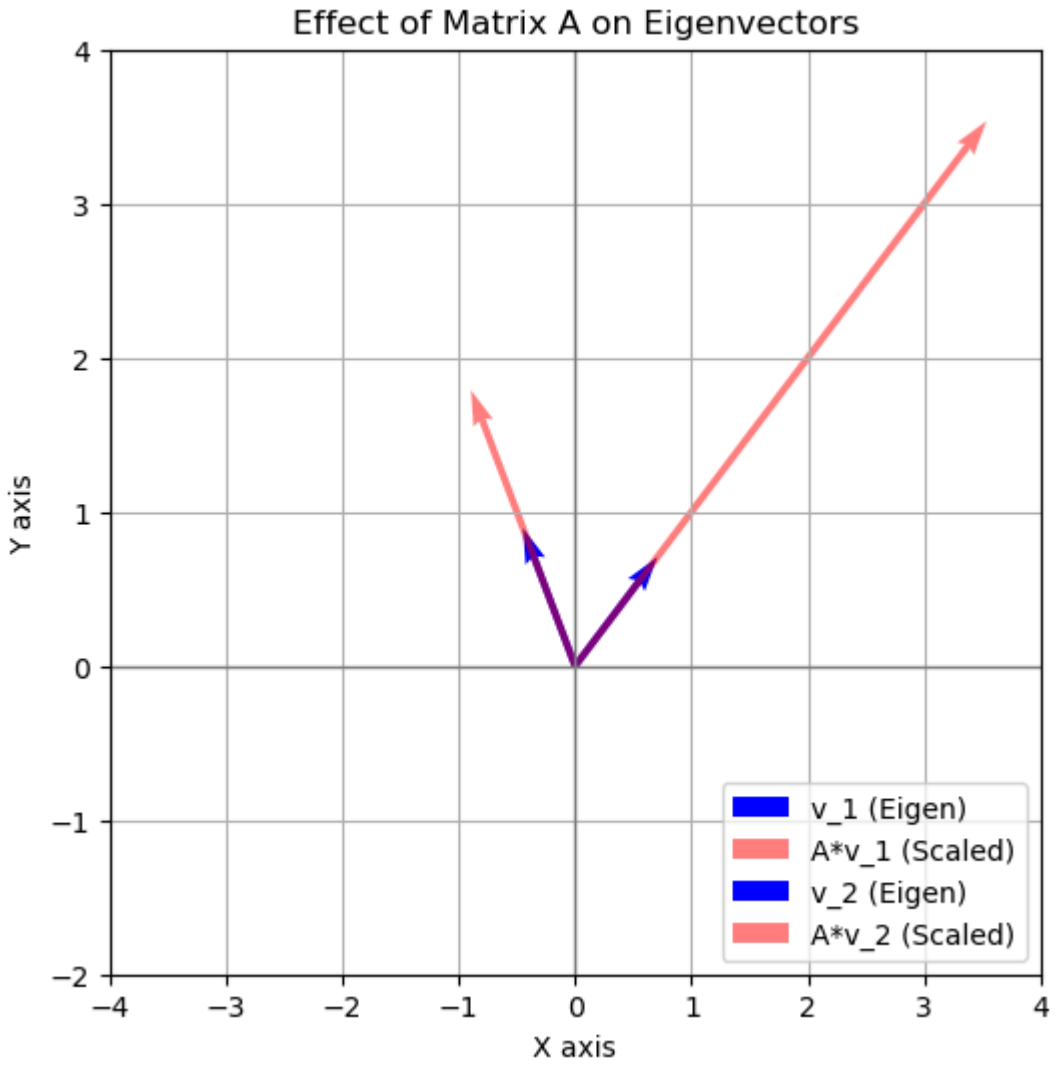
In [13]:
```
print("Verifying A * v = λ * v for each eigenvalue and eigenvector pair:")
for i in range(len(eigenvalues)):
    Av = np.dot(A, eigenvectors[:, i])  # Matrix multiplication of A and each eigenvec
    lambda_v = eigenvalues[i] * eigenvectors[:, i]  # Scalar multiplication of eigenva
    print(f"\nEigenvalue: {eigenvalues[i]}")
    print(f"Eigenvector:\n{eigenvectors[:, i]}")
```

```
    print(f"A * v_{i+1} =\n{Av}\nλ * v_{i+1} =\n{lambda_v}")
    # Using np.allclose to check if the results are numerically close within a toleran
    print("\n Are A*v and λ*v equal?", np.allclose(Av, lambda_v))
```

```
Verifying A * v = λ * v for each eigenvalue and eigenvector pair:

Eigenvalue: 5.0
Eigenvector:
[0.70710678 0.70710678]
A * v_1 =
[3.53553391 3.53553391]
λ * v_1 =
[3.53553391 3.53553391]

 Are A*v and λ*v equal? True

Eigenvalue: 2.0
Eigenvector:
[-0.4472136   0.89442719]
A * v_2 =
[-0.89442719  1.78885438]
λ * v_2 =
[-0.89442719  1.78885438]

 Are A*v and λ*v equal? True
```

In [12]:

```
import matplotlib.pyplot as plt

# Set up the figure and axis
fig, ax = plt.subplots(figsize=(6, 6))

# Original and transformed vectors
origin = [0, 0]  # origin point

for i in range(len(eigenvalues)):
    eigenvector = eigenvectors[:, i]
    transformed_vector = np.dot(A, eigenvector)

    # Plotting the original eigenvector
    ax.quiver(*origin, *eigenvector, scale=1, scale_units='xy', angles='xy', color='b'
    # Plotting the transformed vector
    ax.quiver(*origin, *transformed_vector, scale=1, scale_units='xy', angles='xy', cc

ax.set_xlim(-4, 4)
ax.set_ylim(-2, 4)
ax.set_xlabel('X axis')
ax.set_ylabel('Y axis')
ax.set_title('Effect of Matrix A on Eigenvectors')
ax.axhline(0, color='grey', lw=1)
ax.axvline(0, color='grey', lw=1)
ax.grid(True)
ax.legend(loc='lower right')
plt.show()
```



- Quiver Plot: This plot uses arrows to show vectors. The original vectors (eigenvectors) are plotted alongside their transformed versions (A*v). This clearly shows how each eigenvector is scaled when multiplied by the matrix.
- Scaling and Colors: The original vectors are in blue, and their scaled versions are in red with a slight transparency (alpha=0.5), making it easy to compare their lengths and directions.

Conclusion

The results confirm the linear algebra property that multiplying an eigenvector by its matrix results in a vector that is scaled by its corresponding eigenvalue. This fundamental concept is crucial in fields such as quantum mechanics, vibration analysis, and principal component analysis in data science.

Wrap-Up: A Beginner's Guide to NumPy

As we conclude "A Beginner's Guide to NumPy," we reflect on our journey through the essential features and capabilities of one of Python's most pivotal libraries for numerical computing. This

series was designed to build a solid foundation in NumPy, starting from the basics and progressing to more complex and powerful operations.

Recap of the Series

- **Part 1:** We started with the basics, learning how to create and manipulate NumPy arrays. We explored arithmetic operations and the importance of broadcasting, which enables array operations of varying shapes and sizes.
- **Part 2:** We delved into more specialized topics such as statistical, scientific, and trigonometric operations. Additionally, we covered essential techniques for array manipulation and slicing, enhancing our ability to handle and process data.
- **Part 3:** Our focus shifted to linear algebra. We explored the creation and manipulation of vectors and matrices and learned how to perform fundamental operations such as matrix multiplication, transposition, and calculating determinants.
- **Part 4:** We examined vector transformations and eigenvectors, essential concepts in understanding the behavior of matrices under various operations and transformations.

In this notebook i have **attempted to verify the fundamental linear algebra property** that each eigenvector of a matrix, when multiplied by the matrix, results in a vector that is a scalar multiple of itself (scaled version).

Thank You for Joining

Remember, the journey into Python and NumPy doesn't end here. Continue exploring, learning, and applying your knowledge to discover new solutions and create meaningful impacts with your work.