Numerical Linear Algebra: Homework #4

Due on September 9, 2022 at 10:00PM

Instructor: Professor Blake Barker
Section 1

Michael Snyder

Problem 4.4

Two matrices $A, B \in \mathbb{C}^{m\times m}$ are *unitarily equivalent* if $A = QBQ^*$ for some unitary $Q \in \mathbb{C}^{m\times m}$. Is it true or false that A and B are unitarily equivalent if and only if they have the same singular values?

Proof. ($\Rightarrow$) Let $A, B \in \mathbb{C}^{m\times m}$ and suppose $A = QBQ^*$ for some unitary $Q \in \mathbb{C}^{m\times m}$, that is, A and B are unitarily equivalent. By Theorem 4.1, B has an SVD, namely

$$B = U_B \Sigma V_B^*.$$

Note, by Theorem 4.1, the singular values of B are uniquely determined, hence Σ contains the only set of singular values of B.

Set $U_A = QU_B$ and $V_A^* = V_B^* Q^*$. Since a product of unitary matrices is unitary, U_A and U_B are unitary. Now, using the SVD of B and the fact that A and B are assumed unitarily equivalent,

$$A = QBQ^* = QU_B\Sigma V_B^* Q^* = U_A \Sigma V_A^*.$$

But this is an SVD of A and as stated above, Σ contains the uniquely determined singular values of A. Therefore, since the SVD of A and B share the same Σ, A and B have the same singular values.

($\Leftarrow$) Again, let $A, B \in \mathbb{C}^{m\times m}$, but assume now that the singular values of A and B are the same. Also, to obtain a proof by contradiction, we will assume that $A \neq B$, which is the trivial case. Because A and B are square, by Theorem 4.1, A and B each have an SVD with uniquely determined unitary matrices. Also by 4.1, their singular values are uniquely determined, but are equal in this case, by hypothesis. Using this fact, we have

$$A = U_A \Sigma V_A^* \qquad \textit{and} \qquad B = U_B \Sigma V_A^*.$$

But this implies

$$\Sigma = U_A^* A V_A = U_B^* B V_B,$$

which implies

$$U_A \Sigma V_A^* = A = U_A U_B^* B V_B V_A^*.$$

Since the product of unitary matrices is unitary, set $Q = U_A U_B^*$. Then $Q^* = U_B U_A^*$. Thus, for A and B to be unitarily equivalent, we need $U_B U_A^* = V_B V_A^*$. But since $A \neq B$, and U_A, U_B, V_A, V_B uniquely determined and Σ equal for both A and B, we have

$$A = U_A \Sigma V_A^* = U_A U_B^* B V_B V_A^* = A = U_A U_B^* B U_B U_A^*,$$

i.e., two different singular value decompositions for A. This is a contradiction. Therefore, A and B may not be unitarily similar.

□

Problem 4.5

Theorem 4.1 asserts that every $A \in \mathbb{C}^{m \times n}$ has an SVD $A = U\Sigma V^*$. Show that if A is real, then it has a real SVD ($U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$).

Proof. Following the proof of Theorem 4.1, we set $\sigma_1 = \left\|A\right\|_2$. Then by a compactness argument, there must be a vector $v_1 \in \mathbb{R}^n$ with $\left\|v_1\right\|_2 = 1$ and $\left\|u_1\right\|_2 = \sigma_1$, where $u_1 = Av_1$. Consider any extension of v_1 to an orthonormal basis $\{v_j\}$ for $\mathbb{R}^n$. Similarly, extend u_1 to a basis $\{u_j\}$ for $\mathbb{R}^m$. Then let U_1 and V_1 denote the unitary matrices with columns u_j and v_j, respectively. Then we have

$$U_1^* A V_1 = S = \begin{bmatrix} \sigma_1 & w^* \\ 0 & B \end{bmatrix}$$

Here 0 is an $(m-1) \times 1$ vector, w^* is an $1 \times (n-1)$ row vector, and B has dimension $(m-1) \times (n-1)$. Taking the norm of S, we see that

$$\left\|S\right\|_2 = \left\| \begin{bmatrix} \sigma_1 & w^* \\ 0 & B \end{bmatrix} \begin{bmatrix} \sigma_1 \\ w \end{bmatrix} \right\|_2 \geq \sigma_1^2 + w^* w = (\sigma_1^2 + w^* w)^{1/2} \left\| \begin{bmatrix} \sigma_1 \\ w \end{bmatrix} \right\|$$

But $\left\|S\right\|_2 \geq (\sigma_1^2 + w^* w)^{1/2}$ implies $w = 0$ since U_1, V_1 unitary mean $\left\|S\right\|_2 = \left\|A\right\|_2 = \sigma_1$.

If $n = m = 1$ the proof is complete. Otherwise, the matrix B describes the action of A on the subspace orthogonal to v_1. By the induction hypothesis, B has an SVD $B = U_2 \Sigma_2 V_2^*$. Repeating the process just shown it can be verified that

$$A = U_1 \begin{bmatrix} 1 & 0 \\ 0 & U_2 \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & V_2 \end{bmatrix}^* V_1^*$$

is an SVD of A. This completes the proof of existence. Since we restricted ourselves to the real numbers, the resulting SVD is real.

□

ch_4

September 9, 2022

[80]:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Ellipse
```

[ ]:

```
def plot_origin(matrix):
    return(np.zeros(len(matrix)))
```

[256]:

```
def angle(p_vec, x = None):
    """Computes the angle that the principle axis in an ellipse makes relative␣
 ↪to another vector x (x=(1, 0)* is default), i.e., the rotation angle of the␣
 ↪ellipse.

    Args:
        p_vec (arr): a numpy vector giving the direction of the principle axis␣
 ↪of the ellipse
    """
    # # If the vector is in Q3, reflect to Q1
    if ((p_vec[0] < 0) and (p_vec[1] < 0)):
        p_vec = -1 * p_vec

    # If the vector is in Q4, reflect to Q2
    if ((p_vec[0] > 0) and (p_vec[1] < 0)):
        p_vec = -1 * p_vec

    x = np.array([1, 0]) # Set default comparison vector as x-axis
    inner = p_vec @ x # Compute inner product
    norms = np.linalg.norm(p_vec) * np.linalg.norm(x) # Compute norms
    cos = inner / norms # Compute cos of angle between x and p_vec
    return np.arccos(np.clip(cos, -1.0, 1.0))
```

[257]:

```
# Test ellipse_angle function
print(angle(np.array([1, -1])))
print(angle(np.array([-1, -1])))
print(angle(np.array([-1, 1])))
print(angle(np.array([1, 1])))
print(angle(np.array([0, 1])))
print(angle(np.array([1, 0])))
print(angle(np.array([-1, 0])))
print(angle(np.array([0, -1])))
```

```
2.356194490192345
0.7853981633974484
2.356194490192345
0.7853981633974484
1.5707963267948966
0.0
3.141592653589793
1.5707963267948966
```

[328]:

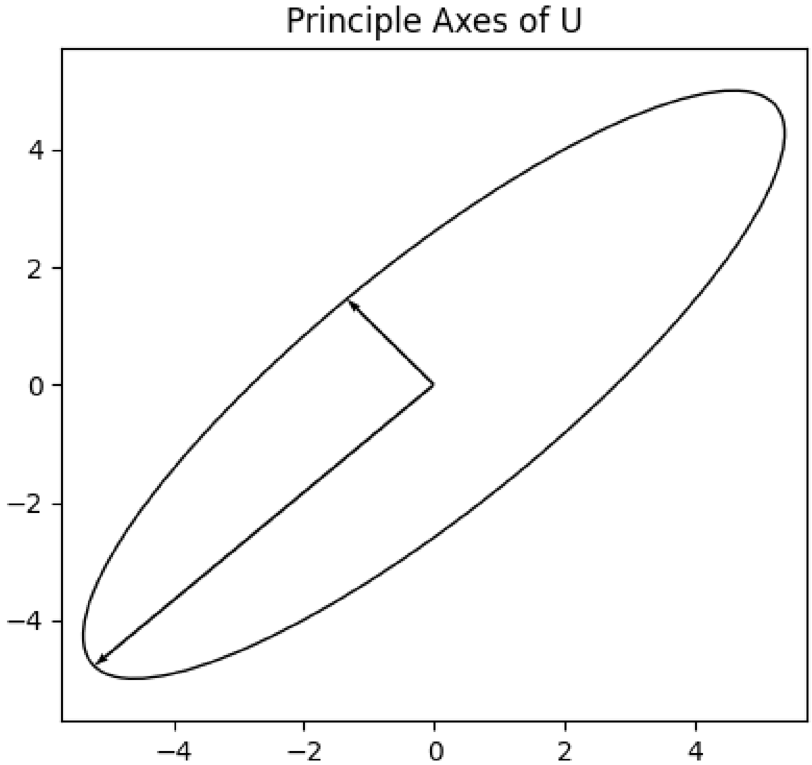
```
def plot_svd(A):
    """Function takes a 2x2 matrix A, computes the SVD of A, and then plots the␣
 ↪vectors u_1, u_2, v_1, and v_2 along with the appropriate ellipse for the␣
 ↪underlying space.

    Args:
        A (arr): 2 x 2 array of floats
    """
    # Check that the provided matrix is 2 x 2 or less and complain if it is not
    assert len(A) <= 2

    # Compute the SVD of the matrix A and store the results
    u, sig, vh = np.linalg.svd(A)

    # Compute U from u
    U = np.zeros((len(A), len(A)))
    U[:, 0] = sig[0] * u[:, 0]
    U[:, 1] = sig[1] * u[:, 1]

    # Define origin for U and vh vectors
    U_origin = np.zeros((len(U), len(U)))
    vh_origin = np.zeros((len(vh), len(vh)))

    # Define angles of principle axes for U and vh
    U_angle = angle(U[:, 0])
    vh_angle = angle(vh[:, 0])

    # Define plot canvas
    fig, ax = plt.subplots(2, 1, figsize=(5, 10))

    # Set axis limits
    U_lim = max([np.abs(U[0, 0]), np.abs(U[1, 0])])
    vh_lim = 1
    ax[0].set_xlim(-U_lim - 0.5, U_lim + 0.5)
    ax[0].set_ylim(-U_lim - 0.5, U_lim + 0.5)
    ax[1].set_xlim(-vh_lim - 0.5, vh_lim + 0.5)
    ax[1].set_ylim(-vh_lim - 0.5, vh_lim + 0.5)

    # Define plot Titles
    ax[0].set_title("Principle Axes of U")
    ax[1].set_title("Principle Axes vh")

    # Plot vectors
    ax[0].quiver(*U_origin, U[0, :], U[1, :], angles='xy', scale_units='xy',␣
↪scale=1, width=.004)
    ax[1].quiver(*vh_origin, vh[0, :], vh[1, :], angles='xy', scale_units='xy',␣
↪scale=1, width=.004)

    # Plot ellipses
    U_ellipse = Ellipse(
        (0, 0),
        width=2 * sig[0],
        height=2 * sig[1],
        angle=U_angle * (180/np.pi),
        zorder=0,
        fill=False
    )
    ax[0].add_patch(U_ellipse)

    vh_ellipse = Ellipse(
        (0, 0),
        width=2,
        height=2,
        angle=vh_angle * (180/np.pi),
        zorder=0,
        fill=False
    )
    ax[1].add_patch(vh_ellipse)

    plt.show()

    #return U, vh
```

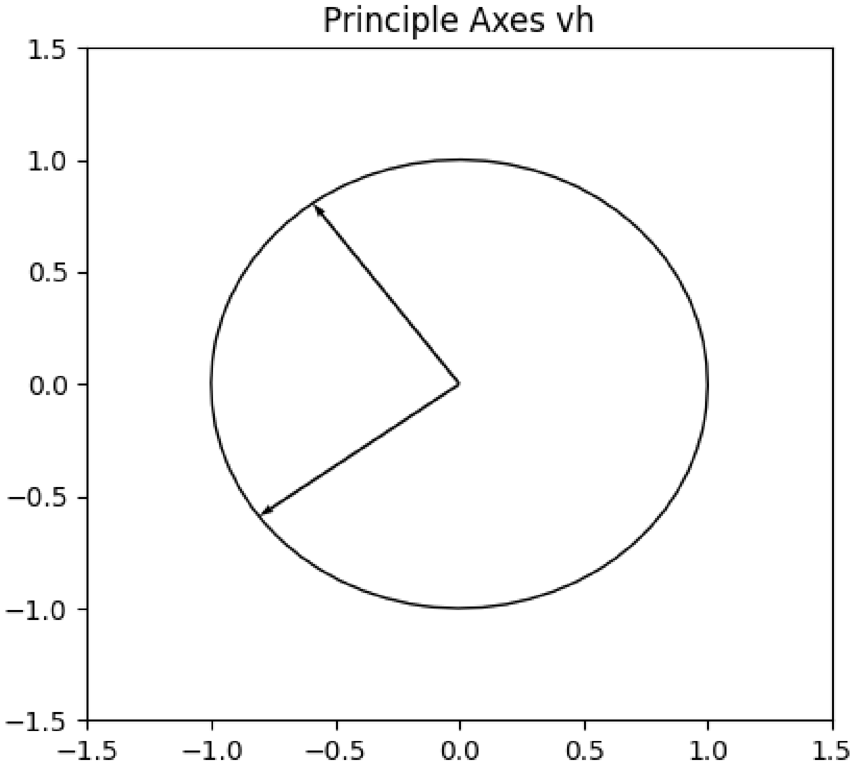
[326]:

```
A = np.array([[5, 2], [3, 4]])
```

[329]:

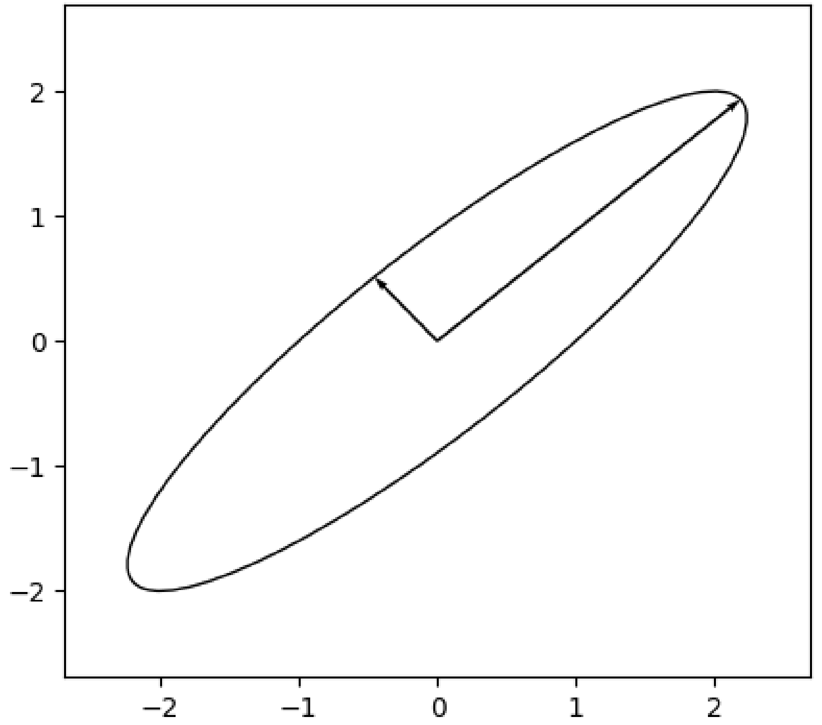
```
plot_svd(A)
```


Principle Axes of U
4
2
0
−2
−4
−4
−2
0
2
4

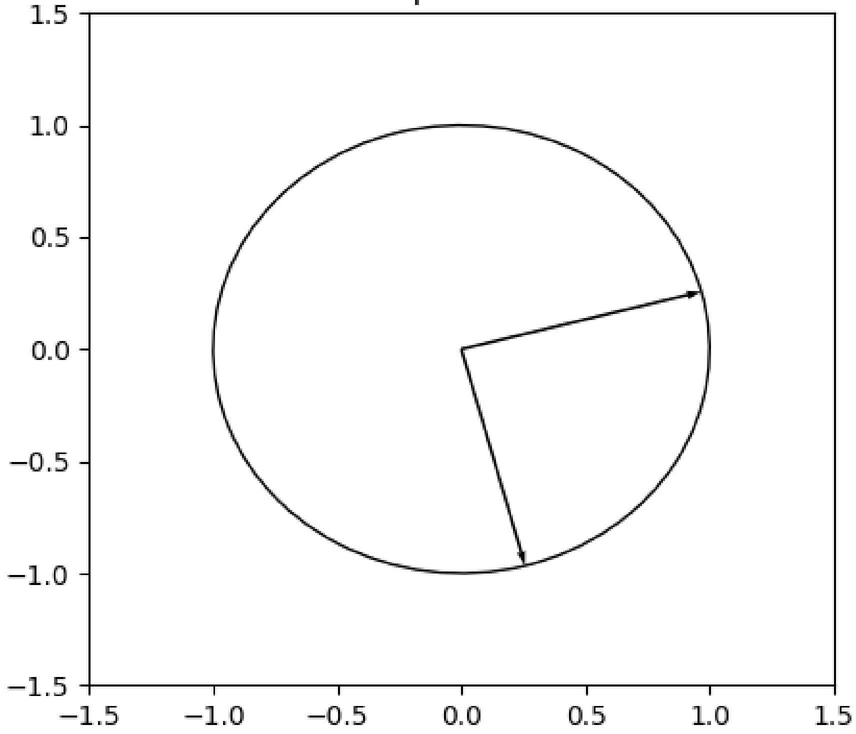

Principle Axes vh
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

[331]:

```
A_37 = np.array([[1, 2], [0, 2]])
plot_svd(A_37)
```



Principle Axes of U



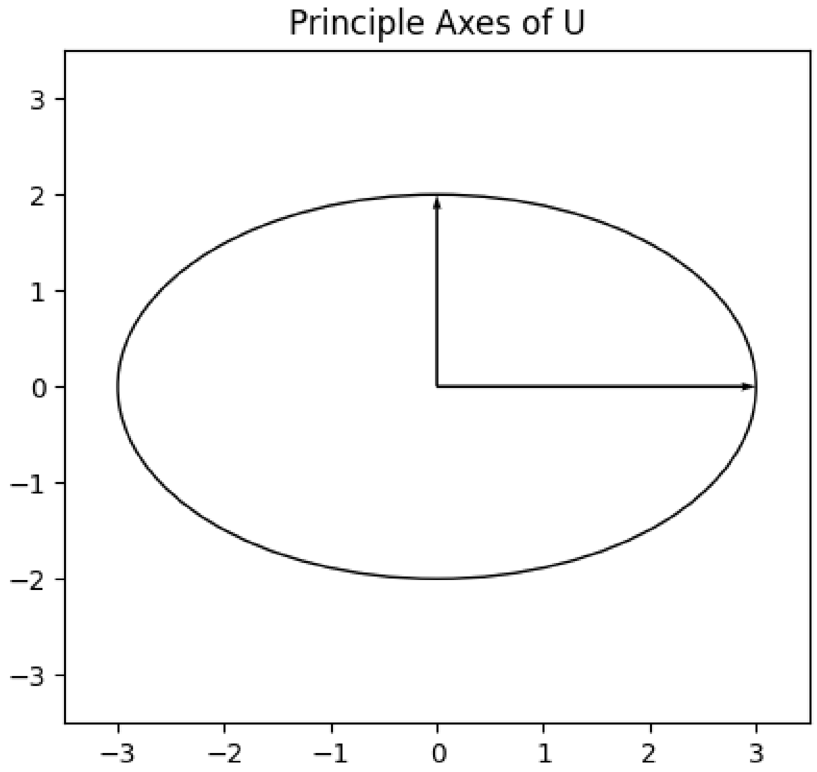
Principle Axes vh

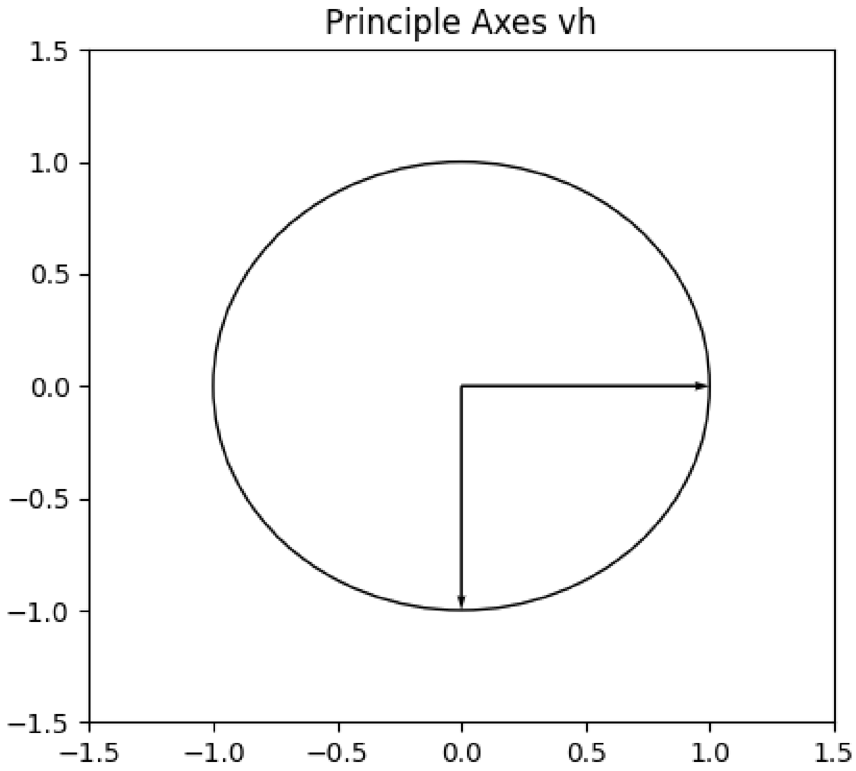
[332]:

```
A_41_a = np.array([[3, 0], [0, -2]])
A_41_b = np.array([[2, 0], [0, 3]])
A_41_d = np.array([[1, 1], [0, 0]])
A_41_e = np.array([[1, 1], [1, 1]])
```

[333]:

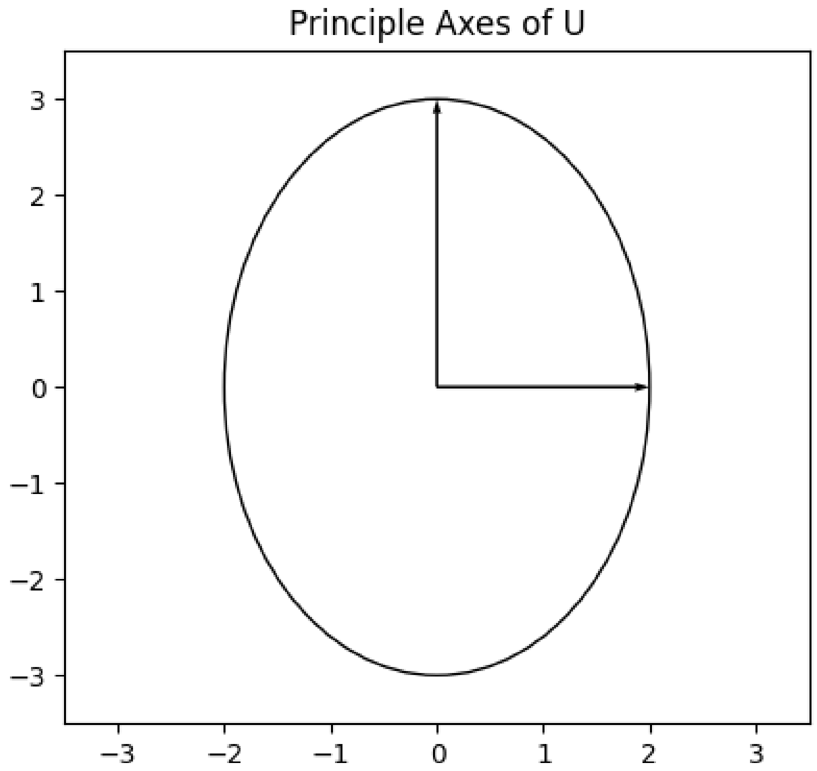
```
plot_svd(A_41_a)
```

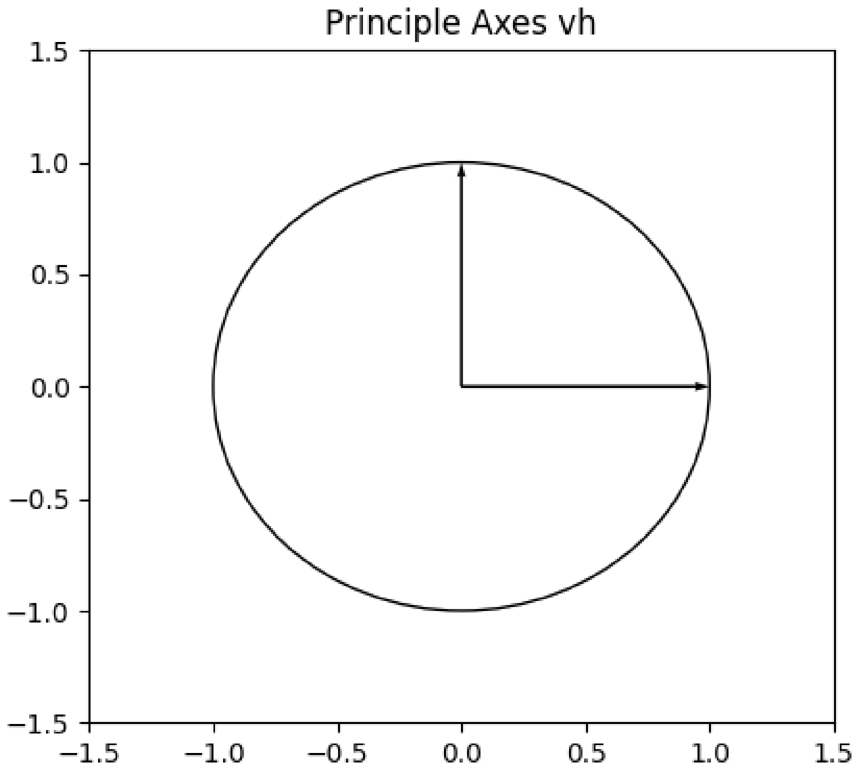

Principle Axes of U


Principle Axes vh

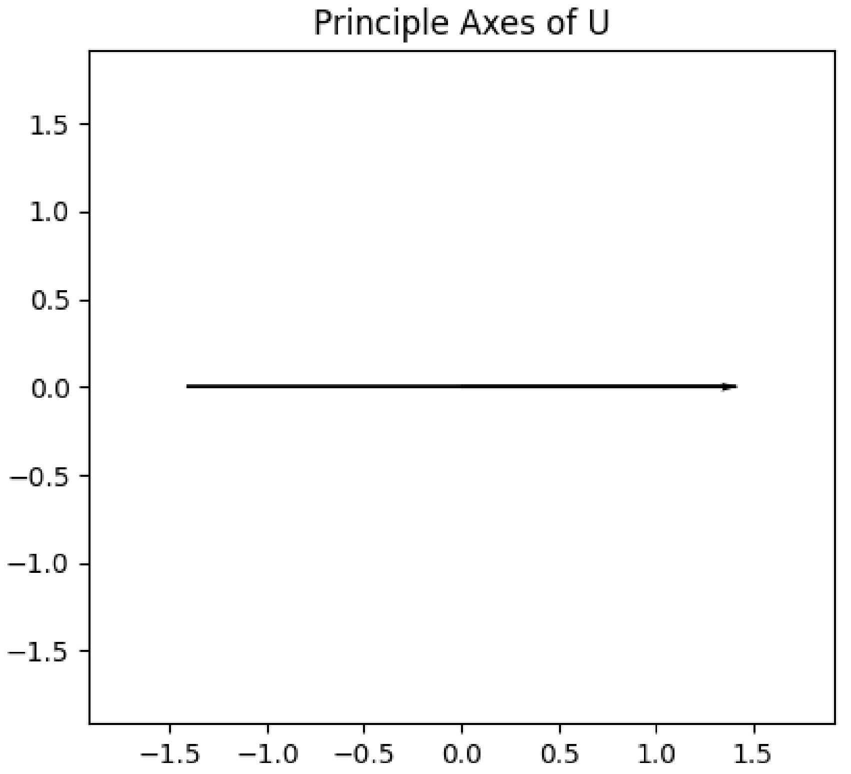
[334]:

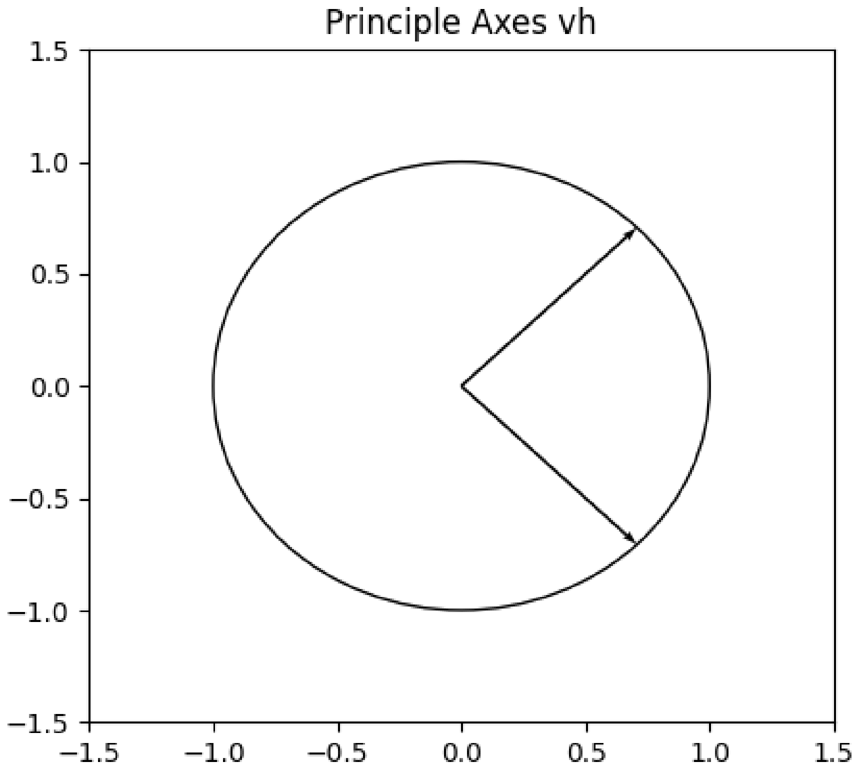
```
plot_svd(A_41_b)
```





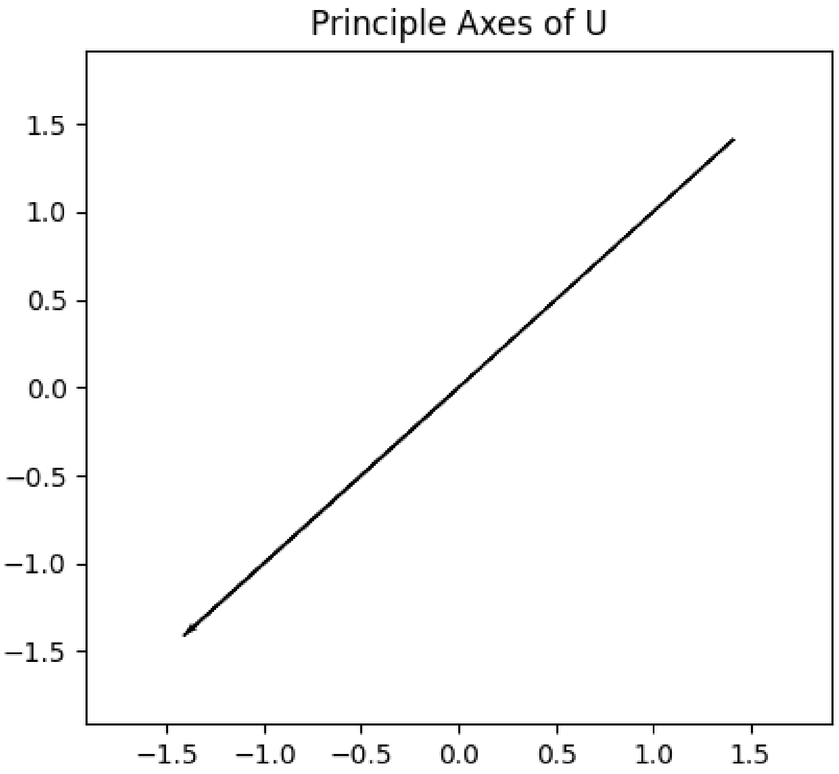
```
[335]: plot_svd(A_41_d)
```

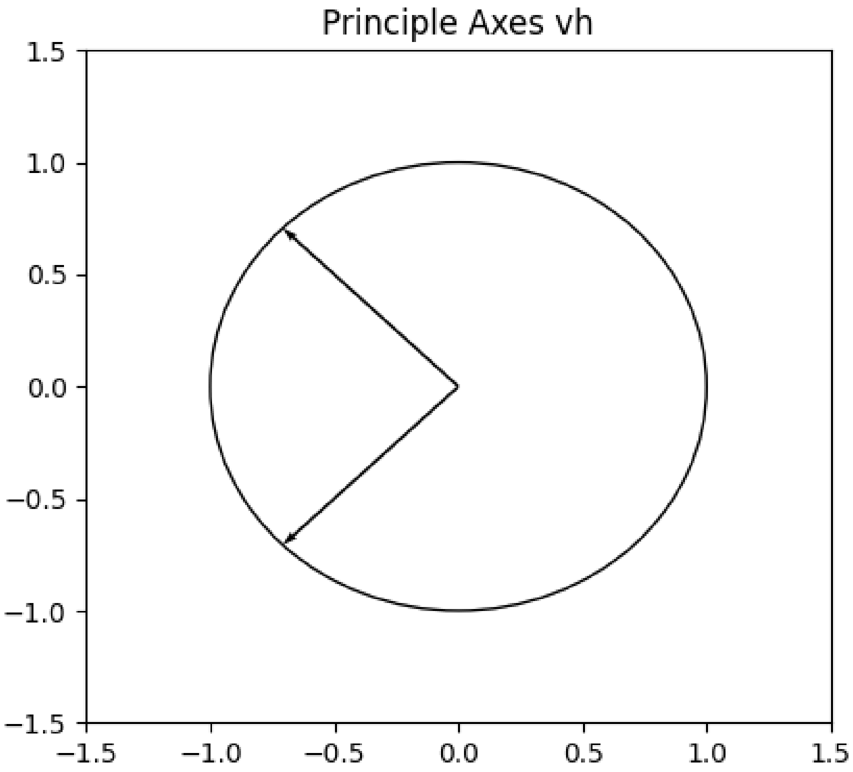

Principle Axes of U


Principle Axes vh

[336]:

```
plot_svd(A_41_e)
```


Principle Axes of U


Principle Axes vh

```
[ ]:
```