# 12.755

Bhargav - EE25BTECH11013

October 10, 2025

# Question

**Question**:
Which one of the following vectors is an eigenvector corresponding to the eigenvalue $\lambda = 1$ for the matrix $\mathbf{A} = \begin{pmatrix} 1 & -1 & 0 \\ 1 & -1 & 1 \\ -1 & 0 & 1 \end{pmatrix}$ is

# Solution

The eigenvector $\mathbf{x}$ of matrix $\mathbf{A}$ corresponding to an eigenvalue $\lambda$ can be found using the relation:

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x} \implies (\mathbf{A} - \lambda\mathbf{I})\,\mathbf{x} = \mathbf{0} \tag{1}$$

For the given eigenvalue $\lambda = 1$:

$$(\mathbf{A} - \mathbf{I})\,\mathbf{x} = \mathbf{0} \tag{2}$$

$$\implies \begin{pmatrix} 0 & -1 & 0 \\ 1 & -2 & 1 \\ -1 & 0 & 0 \end{pmatrix} \mathbf{x} = \mathbf{0} \tag{3}$$

# Solution

This can be solved by representing it as an augmented matrix and using row elimination:

$$\left(\begin{array}{ccc|c} 0 & -1 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ -1 & 0 & 0 & 0 \end{array}\right) \xleftrightarrow{R_1 \leftarrow R_1 + R_2} \left(\begin{array}{ccc|c} 1 & -3 & 1 & 0 \\ 1 & -2 & 1 & 0 \\ -1 & 0 & 0 & 0 \end{array}\right) \tag{4}$$

$$\xleftrightarrow[R_3 \leftarrow R_3 + R_1]{R_2 \leftarrow R_2 - R_1} \left(\begin{array}{ccc|c} 1 & -3 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -3 & 1 & 0 \end{array}\right) \tag{5}$$

$$\xleftrightarrow[R_3 \leftarrow R_3 + 3R_2]{R_1 \leftarrow R_1 + 3R_2} \left(\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right) \tag{6}$$

Thus we get the trivial solution $\mathbf{x} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$.

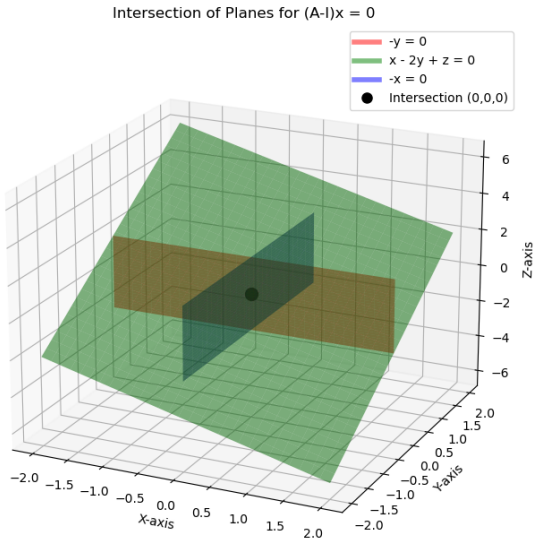
# Solution

This can be further verified by finding the intersection of the planes defined by the system of equations:

$$-y = 0 \tag{7}$$
$$x - 2y + z = 0 \tag{8}$$
$$-x = 0 \tag{9}$$

The only intersection point is the origin $\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$.

# Plot


Intersection of Planes for (A-I)x = 0
-y = 0
x - 2y + z = 0
-x = 0
Intersection (0,0,0)
X-axis
Y-axis
Z-axis

# C Code

```c
#include <stdio.h>
#include <math.h>

#define N 3

// Function to print a matrix
void print_matrix(double A[N][N]) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            printf("%8.3f ", A[i][j]);
        }
        printf("\n");
    }
}

// Function to compute B = A - λI
void subtract_lambda_identity(double A[N][N], double B[N][N],
    double lambda) {
    for (int i = 0; i < N; i++) {
```

# C Code

```c
void subtract_lambda_identity(double A[N][N], double B[N][N],
    double lambda) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            if (i == j)
                B[i][j] = A[i][j] - lambda;
            else
                B[i][j] = A[i][j];
        }
    }
}

// Function to check if λ= 1 is approximately an eigenvalue (det
    ~ 0)
double determinant(double A[N][N]) {
    double det =
        A[0][0]*(A[1][1]*A[2][2] - A[1][2]*A[2][1]) -
        A[0][1]*(A[1][0]*A[2][2] - A[1][2]*A[2][0]) +
        A[0][2]*(A[1][0]*A[2][1] - A[1][1]*A[2][0]);
```

# Python + C Code

```
import numpy as np, ctypes, matplotlib.pyplot as plt
from matplotlib.lines import Line2D

lib = ctypes.CDLL("./libeig.so")
lib.process_matrix.argtypes = [
    np.ctypeslib.ndpointer(dtype=np.double, ndim=2, flags="
        C_CONTIGUOUS"),
    np.ctypeslib.ndpointer(dtype=np.double, ndim=2, flags="
        C_CONTIGUOUS"),
    ctypes.c_double
]; lib.process_matrix.restype = None

A = np.array([[1,-1,0],[1,-1,1],[-1,0,1]],dtype=np.double)
B = np.zeros((3,3),dtype=np.double)
lib.process_matrix(A,B,1.0)
```

# Python + C Code

```
print("\nEigenvalues of A:", np.real_if_close(np.linalg.eigvals(A
    )))

fig = plt.figure(); ax = fig.add_subplot(111, projection='3d')
d = np.linspace(-2,2,30)
ax.plot_surface(*np.meshgrid(d,d), 0*np.meshgrid(d,d)[0], alpha
    =0.5,color='r')
ax.plot_surface(*np.meshgrid(d,d), -np.meshgrid(d,d)[0]+2*np.
    meshgrid(d,d)[1], alpha=0.5,color='g')
ax.plot_surface(0*np.meshgrid(d,d)[0], *np.meshgrid(d,d), alpha
    =0.5,color='b')
ax.scatter(0,0,0,color='k',s=50)

# Legend using Line2D proxies
ax.legend(handles=[Line2D([0],[0], color='r', lw=4, alpha=0.5,
    label='-y=0'),
                   Line2D([0],[0], color='g', lw=4, alpha=0.5,
                       label='x-2y+z=0'),
                   Line2D([0],[0], color='b', lw=4, alpha=0.5,
```

# Python + C Code

```
ax.set_xlabel('X'); ax.set_ylabel('Y'); ax.set_zlabel('Z')
ax.set_title('Intersection of Planes for (A-I)x=0')
ax.view_init(20,-65)
plt.savefig("/mnt/c/Users/bharg/Documents/backupmatrix/
    ee25btech11013/matgeo/12.755/figs/Figure_1.png")
plt.show()
```

# Python Code

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

# Define vectors and point
u1 = np.array([1.0, 1.0, 0.0])
u2 = np.array([0.0, 1.0, 1.0])
P = np.array([1.0, 1.0, 1.0])

# Stack u1 and u2 as columns to form U
U = np.column_stack((u1, u2))

# Compute projection coefficients: inv(U^T U) * U^T * P
coeff = np.linalg.inv(U.T @ U) @ (U.T @ P)

# Compute projection point
P_proj = U @ coeff
```

# Python Code

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

A = np.array([
    [1, -1, 0],
    [1, -1, 1],
    [-1, 0, 1]
])

print("Matrix A:\n", A, "\n" + "-"*30)

eigenvalues, _ = np.linalg.eig(A)

print("Actual Eigenvalues of A:\n", np.real_if_close(eigenvalues)
    )
print("\nNote: 1 is not an eigenvalue of matrix A.")
print("This indicates an error in the problem statement.", "\n" +
     "-"*30)
```

# Python Code

```
# Plane 2: x - 2y + z = 0
X2, Y2 = np.meshgrid(d, d)
Z2 = -X2 + 2*Y2
surf2 = ax.plot_surface(X2, Y2, Z2, alpha=0.5, color='g')
surf2._facecolors2d = surf2._facecolor3d
surf2._edgecolors2d = surf2._edgecolor3d
```

# Python Code

```
# Plane 3: -x = 0
Y3, Z3 = np.meshgrid(d, d)
X3 = np.zeros_like(Y3)
surf3 = ax.plot_surface(X3, Y3, Z3, alpha=0.5, color='b')
surf3._facecolors2d = surf3._facecolor3d
surf3._edgecolors2d = surf3._edgecolor3d

ax.scatter([0], [0], [0], color='black', s=100, label='
    Intersection (0,0,0)')

ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_zlabel('Z-axis')
ax.set_title('Intersection of Planes for (A-I)x = 0')

ax.legend([surf1, surf2, surf3], ['-y = 0', 'x - 2y + z = 0', '-x
     = 0'])

ax.view_init(elev=20, azim=-65)
```