HW 2 - Vectors and Linear Combinations
University of Chicago - Linear Algebra and Python
Professor Chao-Jen Chen

Jennifer Gunawan
Duke University
September 2023

Problem 1

No, three vectors u, v, and w cannot have $u \cdot v < 0$, $u \cdot w < 0$, $v \cdot w < 0$ in an xy plane. A negative dot product between two vectors means that the angle between them are greater than 90°. The three statements above cannot be true in an xy plane because the angle between any two vectors should be between 0° and 180°. Said differently, if u · v < 0, then u and v point in nearly opposite directions (the angle is greater than 90°). Similarly, if u · w < 0, the angle between u and w will be greater than 90°. Coupling both of these statements is not possible in an xy plane since the angle between two vectors has to be between 0° and 180°.

We can prove this mathematically using dot products and trigonometry.
Let θ_{UV}, θ_{UW}, and θ_{VW} be the angles between u and v, u and w, and v and w respectively.
Given that $u \cdot v < 0$ and $u \cdot w < 0$,

$$u \cdot v = |u||v|cos(\theta_{UV})$$
$$u \cdot w = |u||w|cos(\theta_{UW})$$

Since both $u \cdot v$ and $u \cdot w$ are negative, we have:

$$cos(\theta_{UV}) < 0$$
$$cos(\theta_{UW}) < 0$$

Similarly, for θ_{VW}:

$$v \cdot w = |v||w|cos(\theta_{VW})$$

If $\cos(\theta_{UV}) < 0$ and $\cos(\theta_{UW}) < 0$, then the product of these two negative numbers should be positive. I.e.

$$cos(\theta_{UV}) \times cos(\theta_{UW}) > 0$$

The above statement contradicts with the last statement ($v \cdot w < 0$) which implies that $\cos(\theta_{VW}) < 0$. Therefore, we cannot have $u \cdot v < 0$, $u \cdot w < 0$, $v \cdot w < 0$ simultaneously for the three vectors in an xy plane.

Problem 2

To find z_1, z_2, and z_3 in terms of b_1, b_2, and b_3, we can solve $\Delta z = b$ for z by using the inverse matrix (Δ^{-1}).
Since Δ is an upper triangular matrix, we can find its inverse (Δ^{-1}) by taking the inverse of its diagonal elements. Given that the diagonal elements are all -1,

$$\Delta^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Multiplying both sides of $\Delta z = b$ by Δ^{-1}, we get:

$$\Delta^{-1}\Delta z = \Delta^{-1}b$$

This allows us to find z_1, z_2, and z_3 in terms of b_1, b_2, and b_3.
Since $\Delta^{-1}\Delta$ is the identity matrix, we get:

$$z = \Delta^{-1}b$$

We can compute z_1, z_2, and z_3 in terms of b_1, b_2, and b_3 in Python (see the attached code below). Using random numbers to define b_1, b_2, and b_3 as 0.624, 1.557, and 9.667 respectively, we get z_1, z_2, and z_3 as 0.624, 2.181, and 11.849 respectively which verifies the forward difference matrix and its inverse.

Problem 3

Since we don't know the population standard deviation, we can compute the sample standard error using np.var() in Python (see the attached code below).[1] We re-ran the same three cases in HW01 Problem 1 and report the estimates of Var[X] and standard error for each case below.

1. M = 1,000; Mean = 15.071; Var[X] = 38.791; Standard Error = 0.197
2. M = 10,000; Mean = 14.625; Var[X] = 37.976; Standard Error = 0.0616
3. M = 100,000; Mean = 14.664; Var[X] = 38.341; Standard Error = 0.0196

Problem 4

Given that v + w = $\begin{bmatrix}5\\1\end{bmatrix}$ and v - w = $\begin{bmatrix}1\\5\end{bmatrix}$, we can compute and draw v and w as follows:

$$v + w = \begin{bmatrix}5\\1\end{bmatrix} \tag{1}$$

$$v - w = \begin{bmatrix}1\\5\end{bmatrix} \tag{2}$$

Solving simultaneously:

$$v + w + v - w = \begin{bmatrix}5\\1\end{bmatrix} + \begin{bmatrix}1\\5\end{bmatrix}$$

Simplifying:

$$2v = \begin{bmatrix}5\\1\end{bmatrix} + \begin{bmatrix}1\\5\end{bmatrix}$$

$$2v = \begin{bmatrix}(5+1)\\(1+5)\end{bmatrix}$$

$$2v = \begin{bmatrix}6\\6\end{bmatrix}$$

$$v = \begin{bmatrix}6\\6\end{bmatrix} \div 2$$

$$v = \begin{bmatrix}36\\6\end{bmatrix} \div 2$$

$$v = \begin{bmatrix}18\\12\end{bmatrix}$$

[1] I included "ddof=1" parameter to get the unbiased estimate; omitting it yields quantitatively similar results.

Finding w:

$$v + w - (v - w) = \begin{bmatrix}5\\1\end{bmatrix} - \begin{bmatrix}1\\5\end{bmatrix}$$

Simplifying:

$$\begin{aligned}
v + w - v + w =& \begin{bmatrix}5\\1\end{bmatrix} - \begin{bmatrix}1\\5\end{bmatrix}\\
2w =& \begin{bmatrix}5\\1\end{bmatrix} - \begin{bmatrix}1\\5\end{bmatrix}\\
2w =& \begin{bmatrix}(5-1)\\(1-5)\end{bmatrix}\\
2w =& \begin{bmatrix}4\\-4\end{bmatrix}\\
w =& \begin{bmatrix}4\\-4\end{bmatrix} \div 2\\
w =& \begin{bmatrix}16\\-4\end{bmatrix} \div 2\\
w =& \begin{bmatrix}8\\-4\end{bmatrix}
\end{aligned}$$

Finding the angle between v and w,

$$cos(\theta) = (v \cdot w) \div (||v|| \times ||w||)$$

where, $\cos(\theta)$ denotes the cosine of the angle, $v \cdot w$ is the dot product of v and w, $||v||$ is the magnitude of v, and $||w||$ is the magnitude of w.

$$\begin{aligned}
v \cdot w =& \begin{bmatrix}18\\12\end{bmatrix} \cdot \begin{bmatrix}8\\-4\end{bmatrix}\\
=&(18 \times 8) + (12 \times -4)\\
=&144 - 48\\
=&96
\end{aligned}$$

$$||v|| = \sqrt{(18^2 + 12^2)} = \sqrt{324 + 144} = \sqrt{468} = 2\sqrt{117}$$

$$||w|| = \sqrt{(8^2 + -4^2)} = \sqrt{64 + 16} = \sqrt{80} = 4\sqrt{5}$$

$$\begin{aligned}
cos(\theta) =& \frac{v \cdot w}{||v|| \times ||w||}\\
=& \frac{96}{2\sqrt{117} \times 4\sqrt{5}}\\
=& \frac{96 \times 5}{2\sqrt{117} \times 4\sqrt{5} \times 5}\\
=& \frac{480}{10\sqrt{117} \times 4\sqrt{5}}\\
=& \frac{480}{40\sqrt{117}}\\
=& \frac{12}{\sqrt{117}}\\
\theta =& arccos\left(\frac{12}{\sqrt{117}}\right) \approx 1.237 \text{ radians}
\end{aligned}$$

The angle between v and w is approximately 1.237 radians. We can fill the entire xy plane using linear combinations of v and w because v and w are linearly independent since they have different directions (meaning that they are not scalar multiples of each other).
Given that v = [18, 12] and w = [8, -4], we can draw vectors v and w and represent them as arrows on the xy plane (see Figure 1).

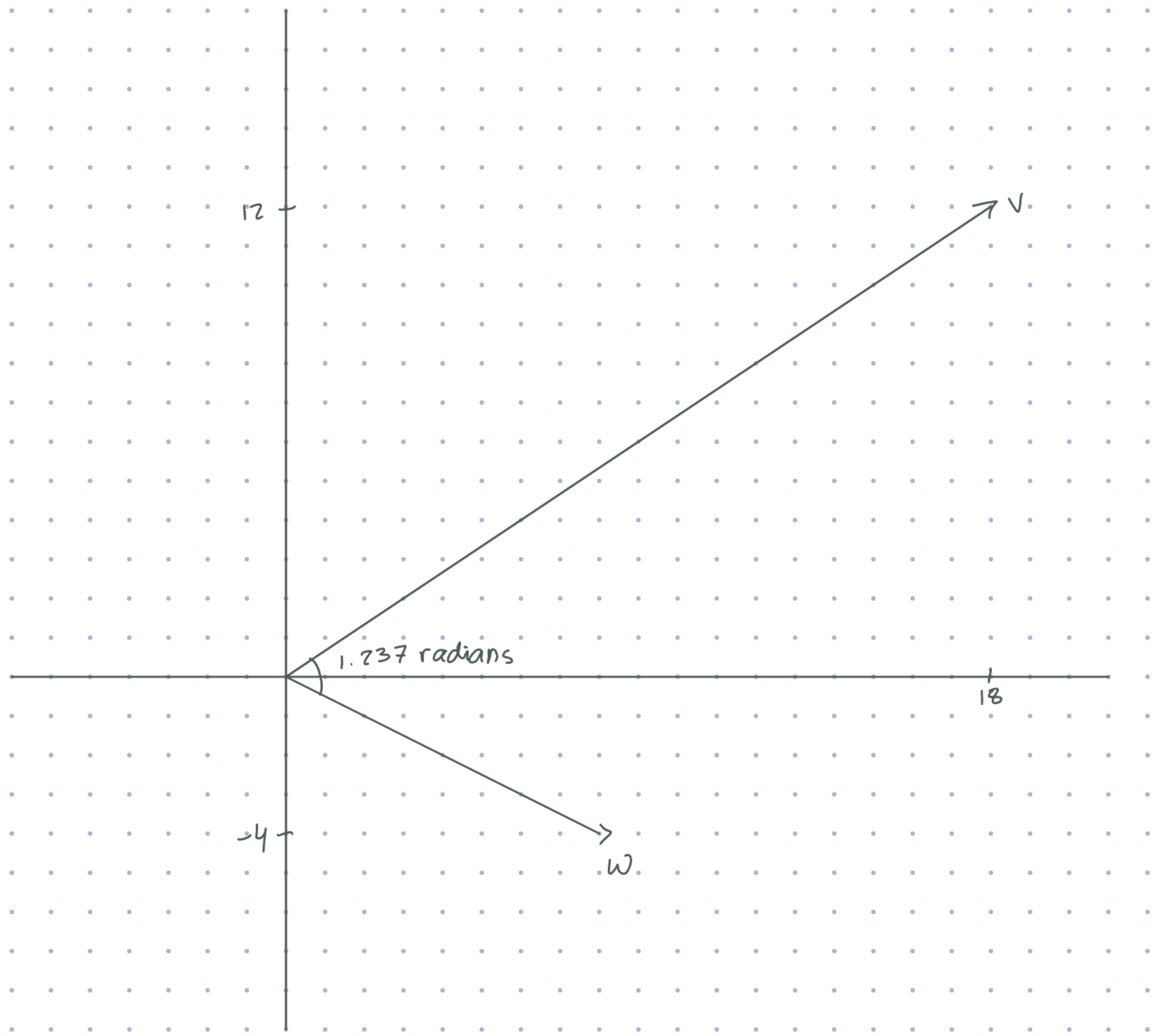

Figure 1: Vector V and W

Homework M2HW02

In [1]:
```
# Problem 2

import numpy as np
import random

# Define the inverse upper triangular matrix
delta_inv = np.array([[1,0,0],[1,1,0],[1,1,1]])

# Define b1, b2, b3
#random.seed(42) #to check work; not part of the actual assignment.
b1 = random.uniform(0,10)
b2 = random.uniform(0,10)
b3 = random.uniform(0,10)

# Define b
b = np.array([b1,b2,b3])

# Solve for z
z = np.dot(delta_inv, b)

# Extract z1, z2, z3
z1, z2, z3 = z

# Print b1, b2, b3, z1, z2, z3
print(f"b1 = {b1}, b2 = {b2}, b3 = {b3}")
print(f"z1 = {z1}, z2 = {z2}, z3 = {z3}")
    '''
    b1 = 0.624363496349557, b2 = 1.5570660360057254, b3 = 9.66712519419846
    z1 = 0.624363496349557, z2 = 2.181429532355282, z3 = 11.84855472655374
    '''
```

```
b1 = 0.624363496349557, b2 = 1.5570660360057254, b3 = 9.667125194198464
z1 = 0.624363496349557, z2 = 2.181429532355282, z3 = 11.848554726553747
```

In [2]:
```
import random
from numpy import array, mean, var, sqrt
```

In [3]:

```
def M2HW02P3(M=100, N=6):
    """
    M: number of trials
    N: number of faces in a die, which should be set to 6 for this problem
    """

    """
    Fill in your Monte Carlo simultion code here.
    """
    # Initialize arrays to store trial results
    X_trials = np.zeros(M)

    # Monte Carlo Simulation for M trials
    #random.seed(42) #to check work; not part of the actual assignment.
    for i in range(M):
        die_numbers = set()
        rolls = 0
        while len(die_numbers) < N:
            roll = random.randint(1, N)
            die_numbers.add(roll)
            rolls += 1
        X_trials[i] = rolls

    # Sample mean
    m = np.mean(X_trials)

    # Sample variance (Var[X]) for X (r.v.)
    v = np.var(X_trials, ddof=1) #specify ddof=1 for unbiased estimate; no

    # Standard error
    s = np.sqrt(v/M)

    """
    You need to calculate and return the following three variables: m, v,
    m: sample mean, i.e., Monte Carlo estimate C.
    v: variance of the random variable X you are simulating.
    s: standard error, i.e., the standard deviation of C.
    """
    #m = 0.
    #v = 0.
    #s = 0.

    return m, v, s
```

In [4]:

```
%time M2HW02P3(10000)

# Call M2HW02P3 for the three cases
num_trials = [1000, 10000, 100000]

for M in num_trials:
    mvs = M2HW02P3(M)
    print(f"M = {M}; mean, Var[X], standard error = {mvs}")
```

```
CPU times: total: 31.2 ms
Wall time: 74.8 ms
M = 1000; mean, Var[X], standard error = (15.071, 38.79074974974975, 0.1
9695367412097126)
M = 10000; mean, Var[X], standard error = (14.6245, 37.975897339733976,
0.061624587089678724)
M = 100000; mean, Var[X], standard error = (14.66415, 38.34077818528185
6, 0.019580801358800883)
```