1.4.19

Aditya Appana - EE25BTECH11004

August 26, 2025

Question

Find a point on the X axis, which is equidistant from the points

$$\begin{pmatrix} 7 \\ 6 \end{pmatrix} \text{ and } \begin{pmatrix} 3 \\ 4 \end{pmatrix}$$

Given Information

Let vector **P** be:

$$\begin{pmatrix} 7 \\ 6 \end{pmatrix} \tag{1}$$

Let vector **Q** be:

$$\begin{pmatrix} 3 \\ 4 \end{pmatrix} \tag{2}$$

Formula

The formula to calculate the x-coordinate of the point **R** is

$$x = \frac{||\mathbf{P}||^2 - ||\mathbf{Q}||^2}{2(\mathbf{P}-\mathbf{Q})^{\mathbf{T}}\mathbf{e_1}}$$

Solution

Substituting $\mathbf{P}, \mathbf{Q}$, and e_1 in this formula :

$$x = \frac{7^2 + 6^2 - (3^2 + 4^2)}{2\begin{pmatrix}4\\2\end{pmatrix}^T \begin{pmatrix}1\\0\end{pmatrix}}$$

$$= \frac{60}{8}$$

$$= 7.5$$

Therefore, the required point is (7.5,0)

Python Code

```
import sys

import numpy as np
import matplotlib.pyplot as plt


def line_gen(A,B):
  len =10
  dim = A.shape[0]
  x_AB = np.zeros((dim,len))
  lam_1 = np.linspace(0,1,len)
  for i in range(len):
    temp1 = A + lam_1[i]*(B-A)
    x_AB[:,i]= temp1.T
  return x_AB
```

Python Code

```
v1 = np.array([7,6]).reshape(-1,1)
v2 = np.array([3,4]).reshape(-1,1)

e1 = np.array([1,0]).reshape(-1,1)

diff = (v1-v2).T

dot_product = diff@e1

denominator = 2*(dot_product)

norm1 = np.linalg.norm(v1)
norm1 = norm1*norm1

norm2 = np.linalg.norm(v2)
norm2 = norm2*norm2
```

Python Code

```
R = (ratio*Q + P) / (ratio + 1)
#Calculating vector R with the first formula

S = (ratio*Q - P) / (ratio - 1)
#Calculating vector S with the second formula
```

Python Code

```
xcoord = (norm1-norm2)/(denominator)
print(xcoord)

x = xcoord[0,0]


reqdpoint = np.array([x,0]).reshape(-1,1)



allcoords = np.block([v1,v2,reqdpoint])




x_1r = line_gen(v1,reqdpoint)
x_2r = line_gen(v2,reqdpoint)
```

Python Code

```
#Plotting all lines
plt.plot(x_1r[0,:],x_1r[1,:],label='$AB$')
plt.plot(x_2r[0,:],x_2r[1,:],label='$BC$')

#Labeling the coordinates
colors = np.arange(1,4)
allcoords = np.block([[v1,v2,reqdpoint]])
plt.scatter(allcoords[0,:], allcoords[1,:], c=colors)
vert_labels = ['v1','v2','required point']
for i, txt in enumerate(vert_labels):

    plt.annotate(f'{txt}\n({allcoords[0,i]:.2f}, {allcoords[1,i
        ]:.2f})',
                 (allcoords[0,i], allcoords[1,i]), textcoords=
                    offset points, xytext=(25,5), ha='center')
```

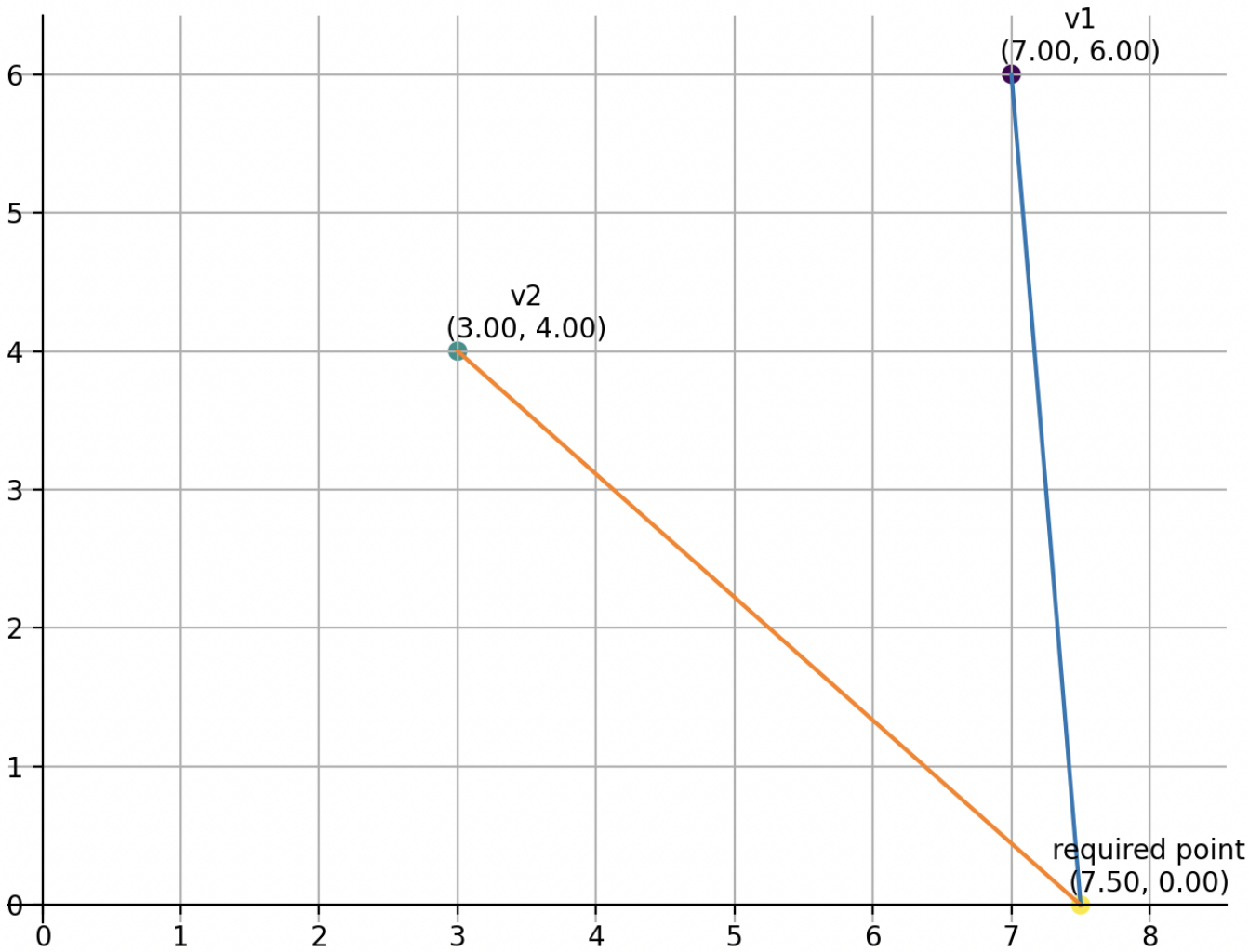
Python Code

```
ax = plt.gca()
ax.spines['top'].set_color('none')
ax.spines['left'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['bottom'].set_position('zero')

plt.grid() # minor
plt.axis('equal')

plt.show()
```

Plot



C Code

```c
#include<stdio.h>

float xfinder (float x1, float y1, float x2, float y2){

float norm1 = x1*x1 + y1*y1;

float norm2 = x2*x2 + y2*y2;

float denominator = x1 - x2;

float xcoord = (norm1 - norm2)/(2 * denominator);

return xcoord;
}
```

Python and C Code

```
import sys
import ctypes

import numpy as np
import matplotlib.pyplot as plt


def line_gen(A,B):
  len =10
  dim = A.shape[0]
  x_AB = np.zeros((dim,len))
  lam_1 = np.linspace(0,1,len)
  for i in range(len):
    temp1 = A + lam_1[i]*(B-A)
    x_AB[:,i]= temp1.T
  return x_AB
```

Python and C Code

```
c_lib=ctypes.CDLL('./main.so')

# Define the argument types for the x function
c_lib.xfinder.argtypes = [ctypes.c_float, ctypes.c_float,ctypes.
    c_float, ctypes.c_float]
# Define the return type of the x function
c_lib.xfinder.restype = ctypes.c_float
# --- Define Points and Calculate 'm' using C function ---

v1 = np.array([7,6])
v2 = np.array([3,4])

xcoord = c_lib.xfinder(
    ctypes.c_float(v1[0]),
    ctypes.c_float(v1[1]),
    ctypes.c_float(v2[0]),
    ctypes.c_float(v2[1])
)
```

Python and C Code

```
v1 = np.array([7,6]).reshape(-1,1)
v2 = np.array([3,4]).reshape(-1,1)

reqdpoint = np.array([xcoord, 0]).reshape(-1,1)

allcoords = np.block([v1,v2,reqdpoint])




x_1r = line_gen(v1,reqdpoint)
x_2r = line_gen(v2,reqdpoint)
```

Python and C Code

```
#Plotting all lines
plt.plot(x_1r[0,:],x_1r[1,:],label='$AB$')
plt.plot(x_2r[0,:],x_2r[1,:],label='$BC$')

#Labeling the coordinates
colors = np.arange(1,4)
allcoords = np.block([[v1,v2,reqdpoint]])
plt.scatter(allcoords[0,:], allcoords[1,:], c=colors)
vert_labels = ['v1','v2','required point']
for i, txt in enumerate(vert_labels):
    #plt.annotate(txt, # this is the text
    plt.annotate(f'{txt}\n({allcoords[0,i]:.2f}, {allcoords[1,i
        ]:.2f})',
                 (allcoords[0,i], allcoords[1,i]), textcoords=
                    offset points, xytext=(25,5), ha='center')
```

Python and C Code

```
# use set_position
ax = plt.gca()
ax.spines['top'].set_color('none')
ax.spines['left'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['bottom'].set_position('zero')

plt.grid() # minor
plt.axis('equal')

plt.show()
```

Plot