محدثه حکیمی - ۴۰۱۲۴۳۰۴۵
گزارش تمرین کامپیوتری

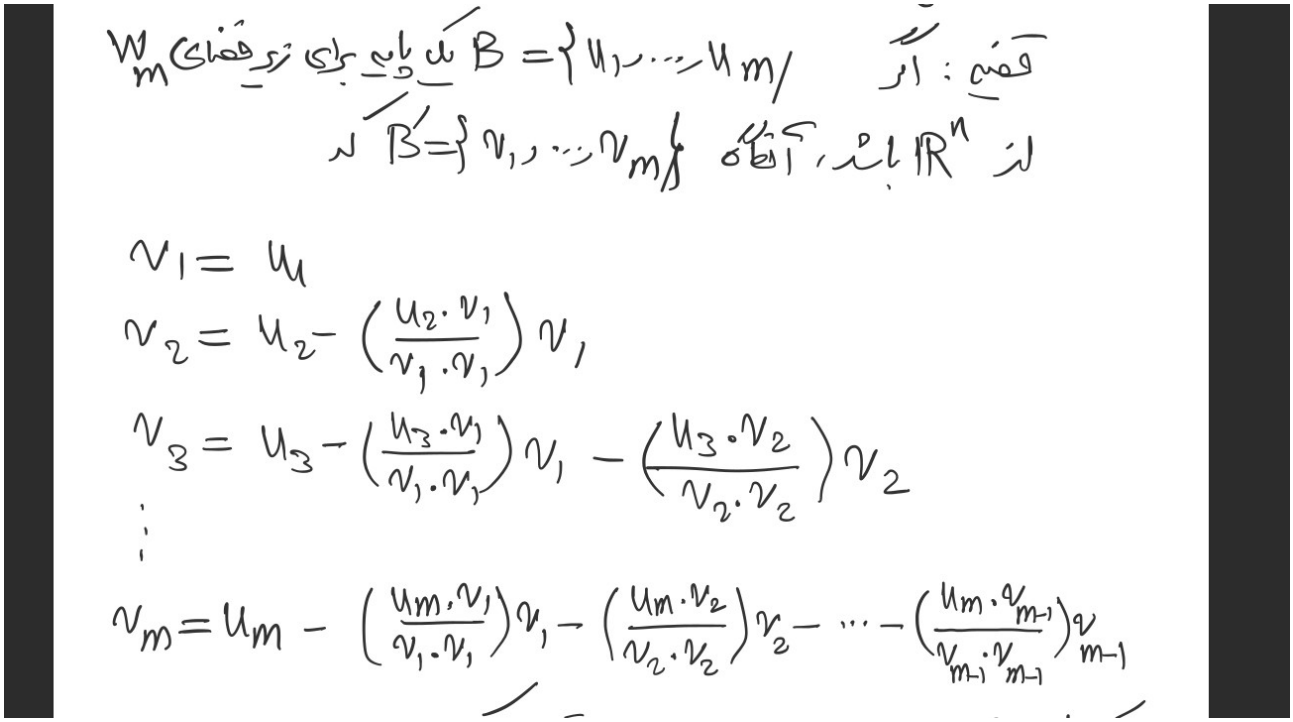
- سؤال اول:

برای این سؤال اول باید چک بشه که x_new داخل اسپن بردارهای داده شده هست یا خیر. برای چک کردن این مورد فقط کافیه که رنک ماتریس A رو با رنک ماتریس افزوده‌ی (A|B) مقایسه کنیم. اگر برابر بودن، متوجه می‌شیم که x_new رو می‌تونیم به صورت ترکیبی خطی از بردارهای قبلی بنویسیم، پس داخل اسپن اونها هست. اگر هم رنک A کمتر بود، پس x_new داخل اسپن نیست و به لیست بردارهای قبلی اضافه‌ش می‌کنیم:

check if x_new is in the span of vectors: the x_new should be written as a linear combination of vectors to be in the span of vectors. so, create an augmented matrix and check if it has a solution. if rank(A) == rank(A|b) => has a solution if rank(A) < rank(A|b) => no solution

In [36]:
```
def solve_augmented_matrix(vectors, v):
    A = np.array(vectors)
    # convert A to a column vector
    A = A.T
    b = np.array(v)
    # print(b)
    aug_matrix = np.concatenate((A, b.reshape(-1, 1)), axis=1)
    print(aug_matrix)
    # print(b.reshape(-1, 1))
    rank_A = linalg.matrix_rank(A)
    rank_aug_matrix = linalg.matrix_rank(aug_matrix)
    if rank_A == rank_aug_matrix:
        return "YES"
    else:
        vectors.append(v)
        return "NO"
```

حالا باید با لیست بردارها - که می‌دونیم مستقل خطی هستن - بردارهای متعامد یکه رو به روش گرام اشمیت بدست بیاریم. داخل جزوه هم فرمول‌ها رو به این شکل داریم:



داخل کد به چنین شکلی درمیاد:

find projection of A onto B

In [16]:
```
def projection(A, B):
    A = np.array(A)
    B = np.array(B)
    proj = (np.dot(A, B) / np.dot(B, B)) * B
    return proj
```

using Gram-Schmidt process, find the orthonormal basis for the vectors

In [17]:
```
def GramSchmidt(vectors):
    b_prime = [vectors[0]]
    for i in range(1, len(vectors)):
        v = vectors[i]
        for j in range(i):
            v = v - projection(v, b_prime[j])
        b_prime.append(v)
    # normalize the vectors
    b_prime = [v / linalg.norm(v) for v in b_prime]
    return b_prime
```

Input

تا اینجای کار فانکشن‌های موردنیاز رو دیدیم، حالا هم ورودی و خروجی:

Input

In [37]:
```
import numpy as np
from numpy import linalg

n = int(input())
m = int(input())
# vectors are in R^n space
vectors = []
for i in range(m):
    vector = list(map(int, input().split()))
    vectors.append(vector)

x_new = list(map(int, input().split()))
```

Output

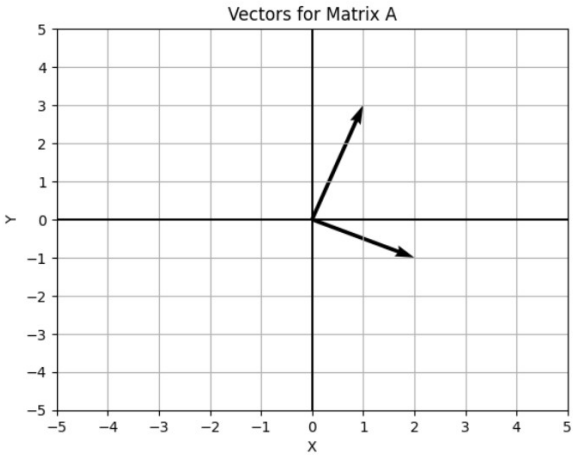
In [41]:
```
print(solve_augmented_matrix(vectors, x_new))
orthonormal_basis = GramSchmidt(vectors)
for v in orthonormal_basis:
    print(v)
```

```
[[1 1 1 1]
 [1 2 1 1]
 [1 2 0 0]]
YES
[0.57735027 0.57735027 0.57735027]
[-0.81649658  0.40824829  0.40824829]
[ 0.          0.70710678 -0.70710678]
```

مثال تست شده رو هم از داخل جزوه برداشتم.

- سؤال دوم:

۵ و ۶) خب اینجا همونطور که خواسته شده بود بردارها رو با quiver داخل صفحه مختصات رسم کردم:

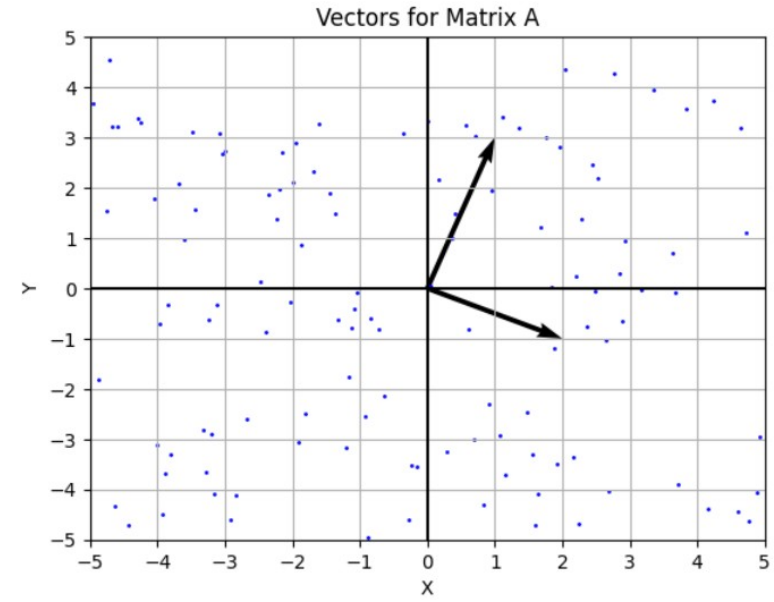


In [2]:
```
import matplotlib.pyplot as plt
import numpy as np


v = np.array([[1, 2], [3, -1]])

fig = plt.figure()
ax = fig.add_subplot(111)

ax.quiver(0, 0, v[0, 0], v[1, 0], angles='xy', scale_units='xy', scale=1)
ax.quiver(0, 0, v[0, 1], v[1, 1], angles='xy', scale_units='xy', scale=1)

ax.grid(True)
ax.set_xlim(-5, 5)
ax.set_ylim(-5, 5)
# make the 0 line thicker
ax.axhline(0, color='black', lw=1.5)
ax.axvline(0, color='black', lw=1.5)
# set ticks
x = np.arange(-5, 6)
y = np.arange(-5, 6)
ax.set_xticks(x)
ax.set_yticks(y)

# set labels
ax.set_xlabel('X')
ax.set_ylabel('Y')

# set title
ax.set_title('Vectors for Matrix A')
```

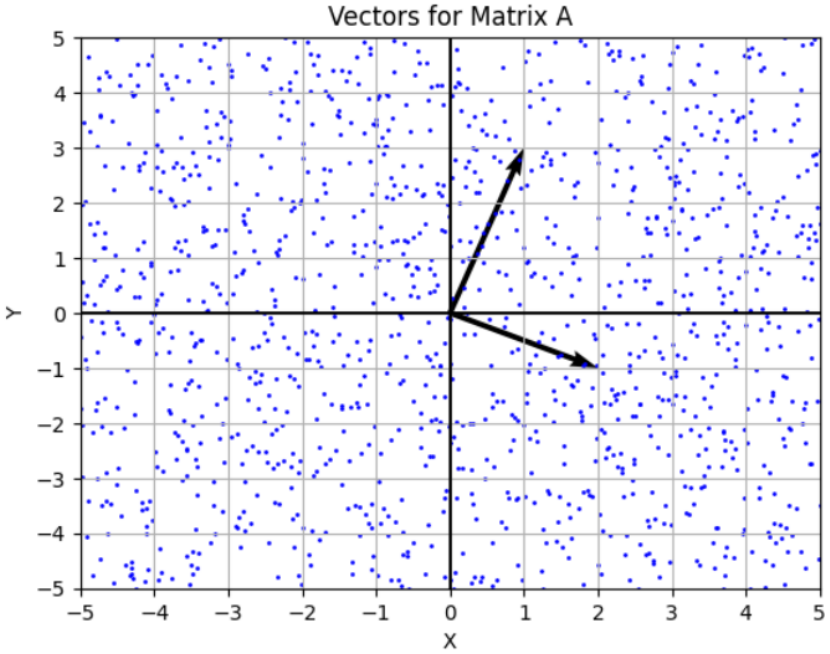
۷) اینجا هم باید نمودار اسپن رو به کمک scatter رسم می‌کردیم. اول کد رو میارم و بعد هم خروجی‌ها رو:

```
# set title
ax.set_title('Vectors for Matrix A')

scaler = np.arange(-10, 10, 0.01)
scaler = scaler.reshape(1, scaler.shape[0])
# random scales to multiply the vectors by
np.random.shuffle(scaler)

# create the span
span1 = v[0][0] * scaler + v[1][0] * scaler
np.random.shuffle(span1[0])
span2 = v[0][1] * scaler + v[1][1] * scaler
np.random.shuffle(span2[0])
ax.scatter(span1, span2, s=1, color='b')

plt.show()
```

مقدار ۰.۰۱ برای ارگومان سوم:



مقدار ۰.۰۰۱ برای ارگومان سوم:

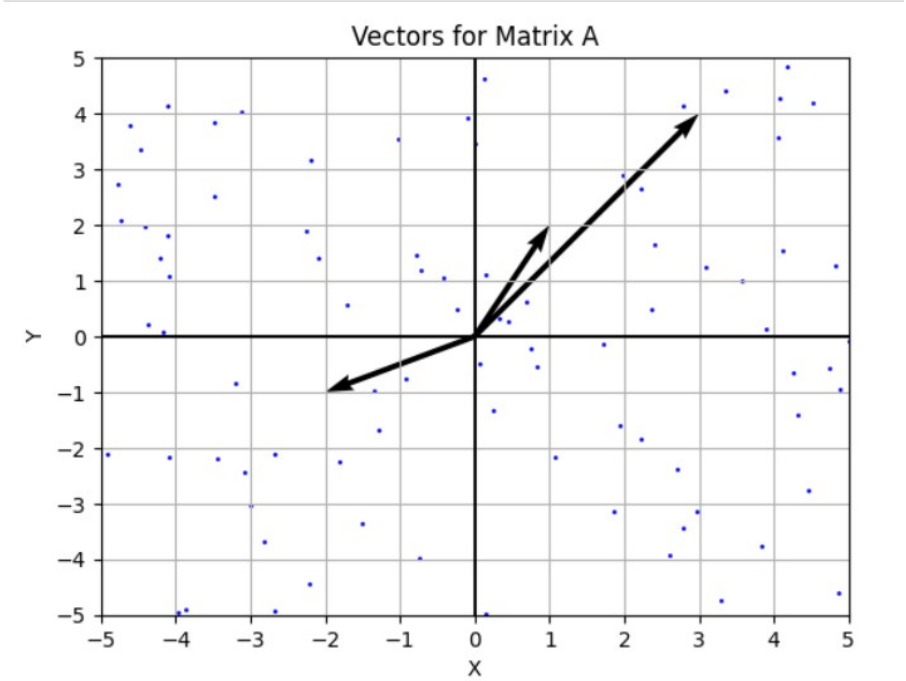


می‌بینیم که تراکم نقاط بیشتر شده. به طور کلی ارگومان سوم در range پایتون، نشون دهنده‌ی step یا گام توی بازه هست، که با بیشترشدنش فاصله‌ی بین مقادیر هم بیشتر میشه، همونطور که داخل شکل‌ها هم مشخصه.

۸) برای ماتریس دوم هم دوتا اسکرین شات خواسته شده رو یک‌جا میارم:

```
ax.quiver(0, 0, v[0, 0], v[1, 0], angles='xy', scale_units='xy', scale=1)
ax.quiver(0, 0, v[0, 1], v[1, 1], angles='xy', scale_units='xy', scale=1)
ax.quiver(0, 0, v[0, 2], v[1, 2], angles='xy', scale_units='xy', scale=1)
```

```
# create the span
span1 = v[0][0] * scaler + v[1][0] * scaler
np.random.shuffle(span1[0])
span2 = v[0][1] * scaler + v[1][1] * scaler
np.random.shuffle(span2[0])
span3 = v[0][2] * scaler + v[1][2] * scaler
np.random.shuffle(span3[0])
ax.scatter(span1, span2, s=1, color='b')
ax.scatter(span1, span3, s=1, color='b')
ax.scatter(span2, span3, s=1, color='b')
```



۹) اسپن یک مجموعه بردار، درواقع مجموعه‌ی تمام بردارهاییه که از ترکیب خطی اون بردارها به دست میان. می‌تونیم بگیم تمام نقاطی در فضا رو نشون میده که میتونیم توسط بردارهای تحت پوشش بهشون برسیم. اگر بردارهای داده شده استقلال خطی داشته باشن میتونن تمام فضا رو پوشش بدن اما در غیر اینصورت، فقط یک زیرفضا پوشش داده میشه.

پس اسپن ماتریس دوم تراکم کمتری داره چون استقلال خطی نداره و فضای کمتری رو میتونه پوشش بده.