Code

June 6, 2023

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import random
import math
```

[2]:
```
# We know that it has 1 or 3 real roots
def ellipse(a,b):
    coef=[1,0,a,b]
    roots=np.roots(coef)
    roots=sorted(roots, key=lambda x: x.real)
    if np.isreal(roots[0]):
        if np.isreal(roots[1]):
            t1=np.linspace(roots[0],roots[1].real,101)
            t2=np.linspace(roots[2].real,5,101)
            n1=random.uniform(roots[0].real,roots[1].real)
            n2=random.uniform(roots[2].real,5)
            Px=random.choice([n1,n2])
            Py=random.choice([-np.sqrt(Px**3+a*Px+b),np.sqrt(Px**3+a*Px+b)])
            plt.scatter(Px,Py,c='red',label='P')
            plt.scatter(Px,-Py,c='green',label='-P')
            n3=random.uniform(roots[0].real,roots[1].real)
            n4=random.uniform(roots[2].real,5)
            Qx=random.choice([n3,n4])
            Qy=random.choice([-np.sqrt(Qx**3+a*Qx+b),np.sqrt(Qx**3+a*Qx+b)])
            plt.scatter(Qx,Qy,c='orange',label='Q')
            m=(Qy-Py)/(Qx-Px)
            c=Py-m*Px
            x3=m**2-Px-Qx
            y3=m*x3+c
            Hx=x3
            Hy=-y3
            plt.scatter(Hx,Hy,c='blue',label='P+Q')
            t1[0]=roots[0]
            t2[0]=roots[2]
            y1=np.ones(101)
            y1[0]=0
            for i in range(1,101):
```

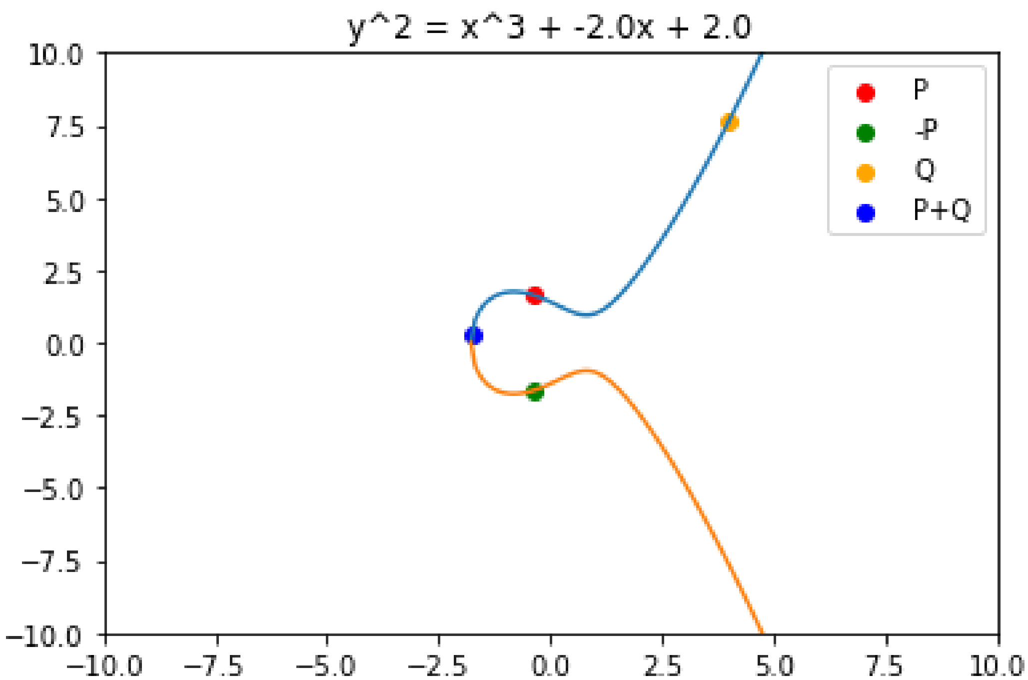
```
                y1[i]=np.sqrt(t1[i]**3+a*t1[i]+b)
            y2=np.ones(101)
            y2[0]=0
            for i in range(1,101):
                y2[i] =np.sqrt(t2[i]**3+a*t2[i]+b)
            plt.plot(t1,y1)
            plt.plot(t2,y2)
            plt.plot(t1,-y1)
            plt.plot(t2,-y2)
            plt.axis([-10, 10, -10, 10])
            plt.legend()
            plt.title("y^2 = x^3 + "+str(a)+"x + "+str(b))
            plt.show()
        else:
            t1=np.linspace(roots[0].real,7,101)
            t1[0]=roots[0].real
            y1=np.ones(101)
            y1[0]=0
            for i in range(1,101):
                y1[i]=np.sqrt(t1[i]**3+a*t1[i]+b)
            Px=random.uniform(roots[0].real,5)
            Py=random.choice([-np.sqrt(Px**3+a*Px+b),np.sqrt(Px**3+a*Px+b)])
            plt.scatter(Px,Py,c='red',label='P')
            plt.scatter(Px,-Py,c='green',label='-P')
            Qx=random.uniform(roots[0].real,5)
            Qy=random.choice([-np.sqrt(Qx**3+a*Qx+b),np.sqrt(Qx**3+a*Qx+b)])
            plt.scatter(Qx,Qy,c='orange',label='Q')
            m=(Qy-Py)/(Qx-Px)
            c=Py-m*Px
            x3=m**2-Px-Qx
            y3=m*x3+c
            Hx=x3
            Hy=-y3
            plt.scatter(Hx,Hy,c='blue',label='P+Q')
            plt.plot(t1,y1)
            plt.plot(t1,-y1)
            plt.axis([-10, 10, -10, 10])
            plt.title("y^2 = x^3 + "+str(a)+"x + "+str(b))
            plt.legend()
            plt.show()
    elif np.isreal(roots[1]):
        t1=np.linspace(roots[1].real,7,101)
        t1[0]=roots[1].real
        y1=np.ones(101)
        y1[0]=0
        for i in range(1,101):
            y1[i] = np.sqrt(t1[i]**3+a*t1[i]+b)
```

```
        Px=random.uniform(roots[1].real,5)
        Py=random.choice([-np.sqrt(Px**3+a*Px+b),np.sqrt(Px**3+a*Px+b)])
        p.scatter(Px,Py,c='red',label='P')
        p.scatter(Px,-Py,c='green',label='-P')
        Qx=random.uniform(roots[1].real,5)
        Qy=random.choice([-np.sqrt(Qx**3+a*Qx+b),np.sqrt(Qx**3+a*Qx+b)])
        p.scatter(Qx,Qy,c='orange',label='Q')
        m=(Qy-Py)/(Qx-Px)
        c=Py-m*Px
        x3=m**2-Px-Qx
        y3=m*x3+c
        Hx=x3
        Hy=-y3
        plt.scatter(Hx,Hy,c='blue',label='P+Q')
        plt.plot(t1,y1)
        plt.plot(t1,-y1)
        plt.axis([-10, 10, -10, 10])
        plt.title("y^2 = x^3 + "+str(a)+"x + "+str(b))
        plt.legend()
        plt.show()
    else:
        t1=np.linspace(roots[2].real,7,101)
        t1[0]=roots[2].real
        y1=np.ones(101)
        y1[0]=0
        for i in range(1,101):
            y1[i] = np.sqrt(t1[i]**3+a*t1[i]+b)
        Px=random.uniform(roots[2].real,5)
        Py=random.choice([-np.sqrt(Px**3+a*Px+b),np.sqrt(Px**3+a*Px+b)])
        p.scatter(Px,Py,c='red',label='P')
        p.scatter(Px,-Py,c='green',label='-P')
        Qx=random.uniform(roots[2].real,5)
        Qy=random.choice([-np.sqrt(Qx**3+a*Qx+b),np.sqrt(Qx**3+a*Qx+b)])
        p.scatter(Qx,Qy,c='orange',label='Q')
        m=(Qy-Py)/(Qx-Px)
        c=Py-m*Px
        x3=m**2-Px-Qx
        y3=m*x3+c
        Hx=x3
        Hy=-y3
        plt.scatter(Hx,Hy,c='blue',label='P+Q')
        plt.plot(t1,y1)
        plt.plot(t1,-y1)
        plt.axis([-10, 10, -10, 10])
        plt.title("y^2 = x^3 + "+str(a)+"x + "+str(b))
        plt.legend()
        plt.show()
```

[3]:

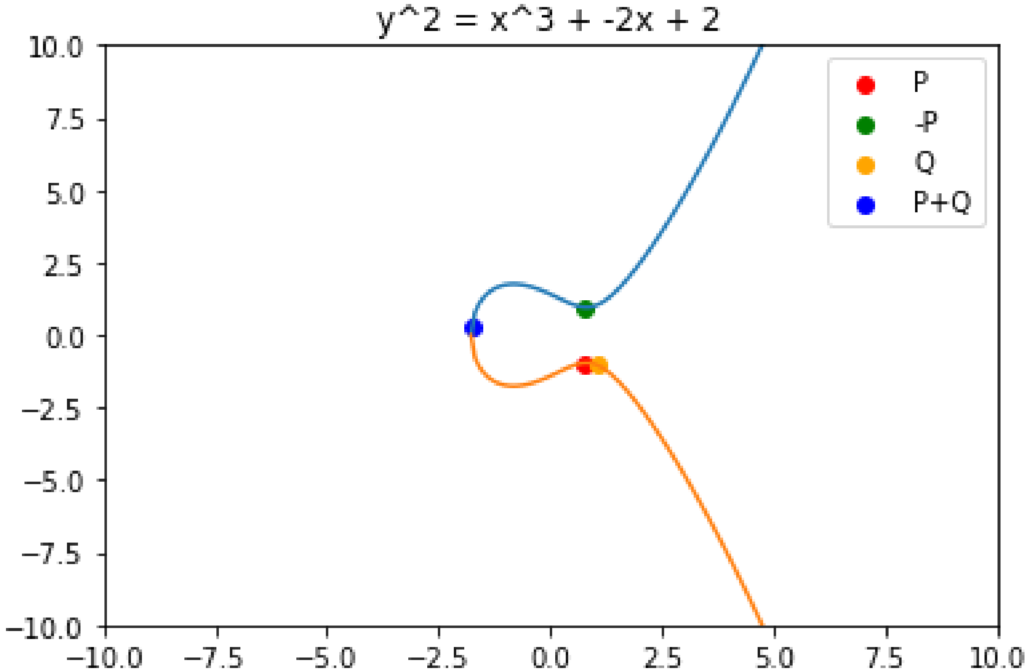
```python
print("Elliptic curve y^2=x^3+ax+b")

# User inputs for a and b
a = float(input("Enter the value of a: "))
b = float(input("Enter the value of b: "))

if 4*(a**3)+27*(b**2)==0:
    print("The cubic polynomial x^3 + ax + b has repeated roots")
    while True:
        try:
            a = float(input("Enter the value of a: "))
            b = float(input("Enter the value of b: "))
            break
        except Exception as e:
            print("The input must be a real number!")

ellipse(a,b)
```

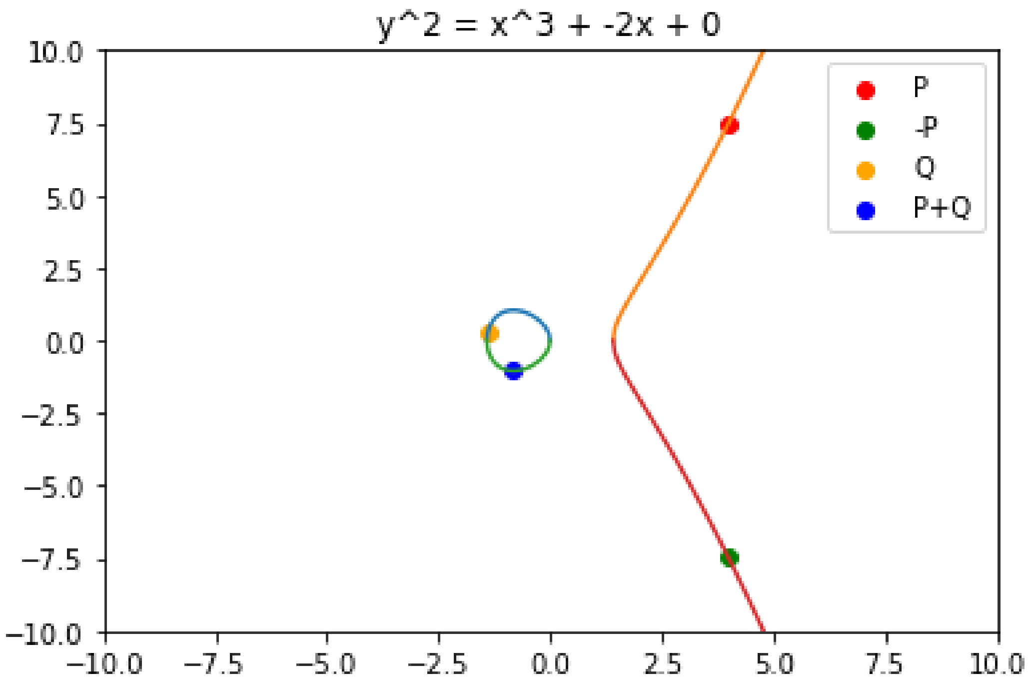
```
Elliptic curve y^2=x^3+ax+b
Enter the value of a: -2
Enter the value of b: 2
```



[5]:

```python
ellipse(-2,2)
ellipse(-2,0)
```


y^2 = x^3 + -2x + 2
P
-P
Q
P+Q
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


y^2 = x^3 + -2x + 0
P
-P
Q
P+Q
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

[ ]: