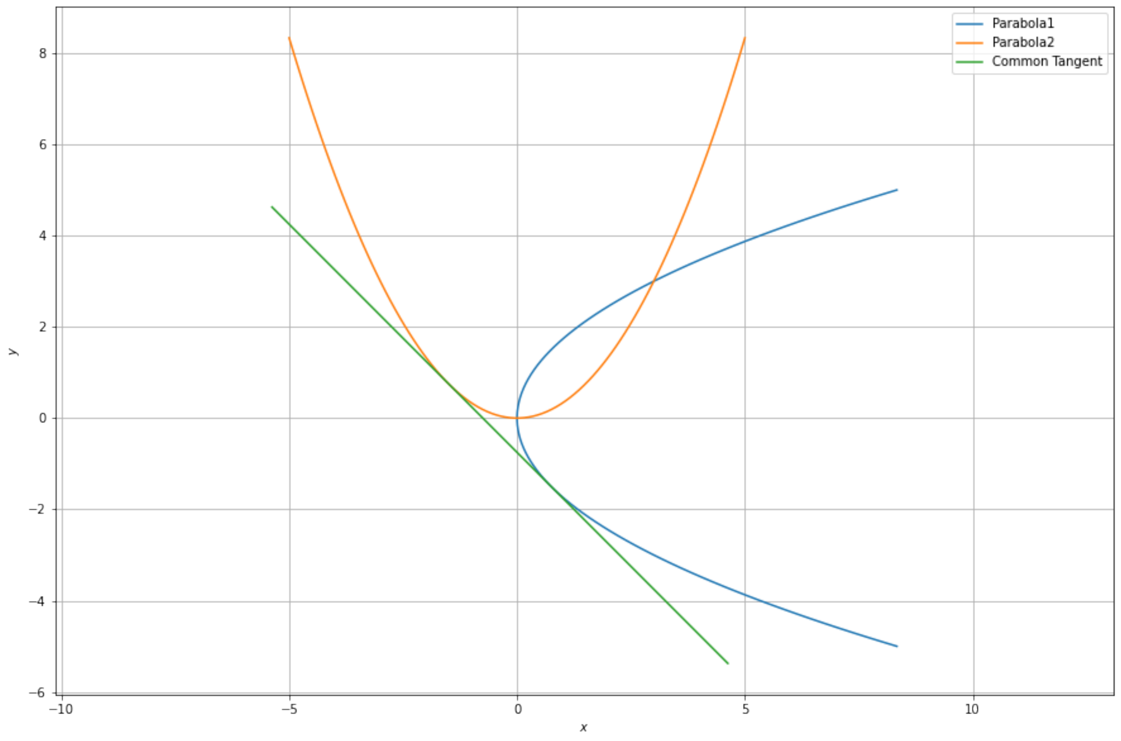
Conics

March 28, 2021

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from coeffs import *
```

[2]:
```
#setting up plot
fig = plt.figure(figsize=(15,10))
ax = fig.add_subplot(111, aspect='equal')

lrct = 3
len=100
y  = np.linspace(-5,5,len)
x = np.power(y,2)/3
P1 = np.vstack((x,y))
x_par1 = P1
plt.plot(x_par1[0,:],x_par1[1,:],label="Parabola1")

x  = np.linspace(-5,5,len)
y = np.power(x,2)/3
P2 = np.vstack((x,y))
x_par2 = P2
plt.plot(x_par2[0,:],x_par2[1,:],label="Parabola2")

D = np.array([-3/8,-3/8])
m = np.array([1,-1])
T = line_dir_pt(m,D,-5,5)

plt.plot(T[0,:],T[1,:],label='Common Tangent' )

ax.plot()
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.legend(loc='best')
plt.axis('equal')
plt.grid()
plt.show()
```

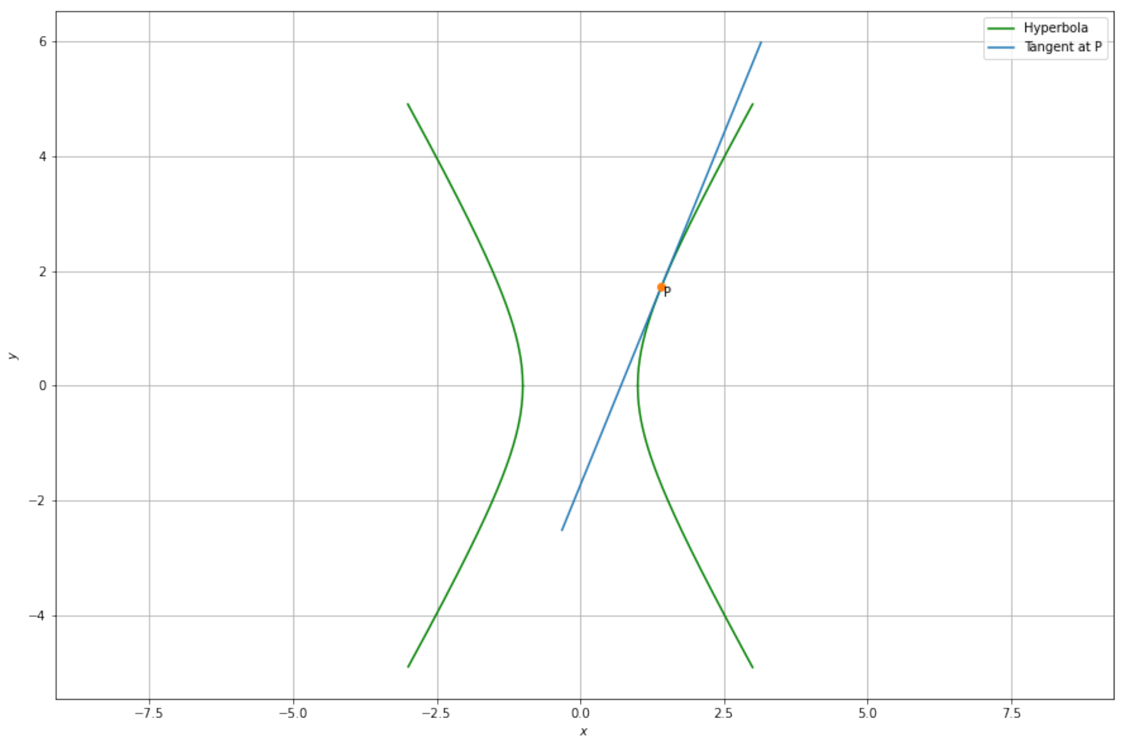


[3]:
```
fig = plt.figure(figsize=(15,10))
ax = fig.add_subplot(111, aspect='equal')

P  = np.array([2**0.5,3**0.5])
S1 = np.array([2,0])
S2 = np.array([-2,0])
O = np.array([0,0])
omat = np.array(([0,1],[-1,0]))

PS1 = np.linalg.norm(P - S1)
PS2 = np.linalg.norm(P - S2)

a = (np.absolute(PS1 -PS2))*0.5
e = np.linalg.norm(S1 -S2)/2*a

b = np.sqrt((a**2)*((e**2) - 1))

V = np.array(([a**-2,0],[0,-1*b**-2]))

eigval,eigvec = np.linalg.eig(V)
D = np.diag(eigval)
G = eigvec
```

```python
len = 1000

y1 = np.linspace(-3,3,len)
y2 = np.sqrt((1 - V[0,0]*np.power(y1,2))/V[1,1])
y3 = -1*np.sqrt((1 - V[0,0]*np.power(y1,2))/V[1,1])
y = np.hstack((np.vstack((y1,y2)),np.vstack((y1,y3))))
plt.plot(y[0,:len],y[1,:len],color='g',label='Hyperbola')
plt.plot(y[0,len+1:],y[1,len+1:],color='g')
n = V@P
m = omat@n

P  = np.array([2**0.5,3**0.5])
T = line_dir_pt(m,P,-3,3)
plt.plot(T[0,:],T[1,:],label='Tangent at P')
plt.plot(P[0],P[1],'o')
plt.text(P[0] * (1 + 0.03), P[1] * (1 - 0.1) , 'P')
print(n,'x=',n@P)
ax.plot()
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.legend(loc=1)
plt.grid() # minor
plt.axis('equal')
plt.show()
```

```
[ 1.41421356 -0.57735027] x= 1.0000000000000002
```

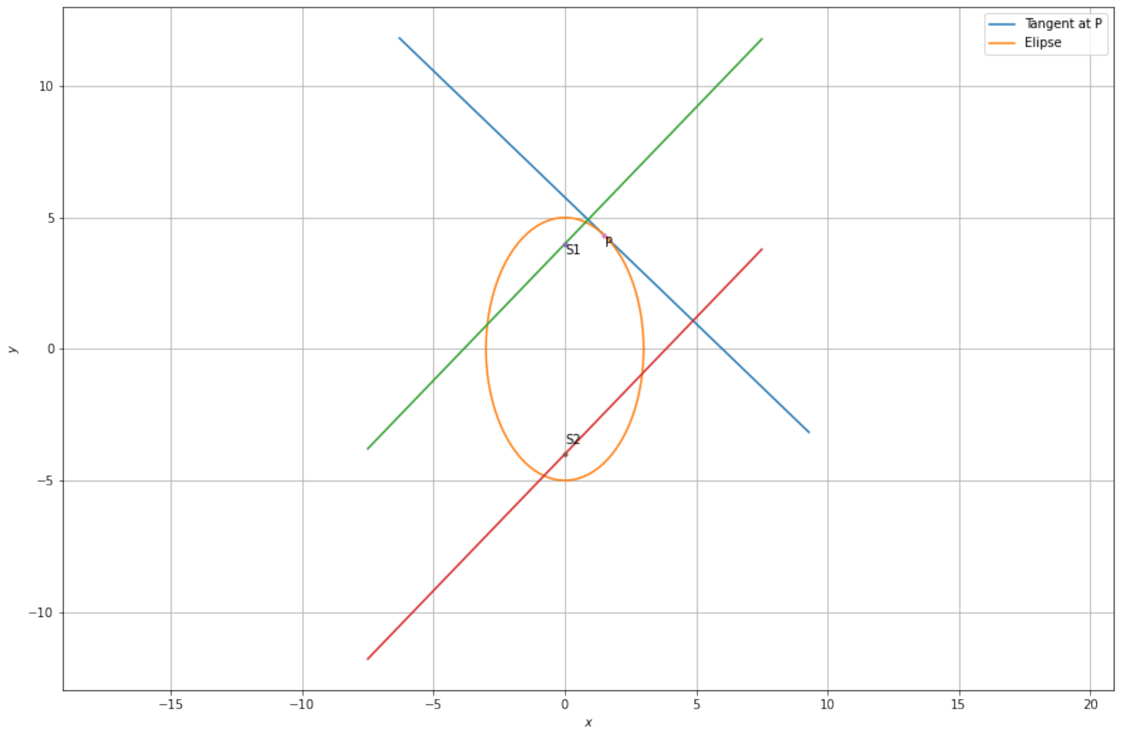
```
<ipython-input-3-5b58db6d7fe2>:27: RuntimeWarning: invalid value encountered in
sqrt
  y2 = np.sqrt((1 - V[0,0]*np.power(y1,2))/V[1,1])
<ipython-input-3-5b58db6d7fe2>:28: RuntimeWarning: invalid value encountered in
sqrt
  y3 = -1*np.sqrt((1 - V[0,0]*np.power(y1,2))/V[1,1])
```



[4]:
```
fig = plt.figure(figsize=(15,10))
ax = fig.add_subplot(111, aspect='equal')

V = np.array(([25,0],[0,9]))
V1 = V/225
P = np.array([3,5*(3**0.5)])*0.5

len =1000
eigval,eigvec = np.linalg.eig(V1)
D = np.diag(eigval)
T = eigvec
y = np.zeros((2,len))
thetha = np.linspace(0,2*np.pi,len)
y[0,:] = ((1/D[0,0])**0.5)*np.cos(thetha)
y[1,:] = ((1/D[1,1])**0.5)*np.sin(thetha)


if(D[1,1] > D[0,0]):
   e = np.sqrt(1 - (D[0,0]/D[1,1]))
   F1 = np.array([((1/D[0,0])**0.5)*e,0])
   F2 = np.array([(-1*(1/D[0,0])**0.5)*e,0])
else:
   e = np.sqrt(1 - (D[1,1]/D[0,0]))
```

```
    F1 = np.array([0,((1/D[1,1])**0.5)*e])
    F2 = np.array([0,(-1*(1/D[1,1])**0.5)*e])

n = V@P
l1 = np.absolute((n@(F1 - P))/np.linalg.norm(n))
l2 = np.absolute((n@(F2 - P))/np.linalg.norm(n))

omat = np.array(([0,1],[-1,0]))
m = omat@n
T = line_dir_pt(m,P,-0.2,0.2)
plt.plot(T[0,:],T[1,:],label='Tangent at P')
m = omat@n
plt.plot(y[0,:],y[1,:],label='Elipse')
K1 = line_dir_pt(n,F1,-0.2,0.2)
plt.plot(K1[0,:],K1[1,:])
m = omat@n
K2 = line_dir_pt(n,F2,-0.2,0.2)
plt.plot(K2[0,:],K2[1,:])
print("The product of the perpendicular distances between Tangent and Focii is␣
 ↪",l1*l2)

plt.plot(F1[0], F1[1], '.')
plt.text(F1[0] * (1 + 0.03), F1[1] * (1 - 0.1) , 'S1')

plt.plot(F2[0], F2[1], '.')
plt.text(F2[0] * (1 + 0.03), F2[1] * (1 - 0.1) , 'S2')

plt.plot(P[0], P[1], '.')
plt.text(P[0] * (1 + 0.03), P[1] * (1 - 0.1) , 'P')

ax.plot()
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.legend(loc=1)
plt.grid() # minor
plt.axis('equal')
plt.show()
```

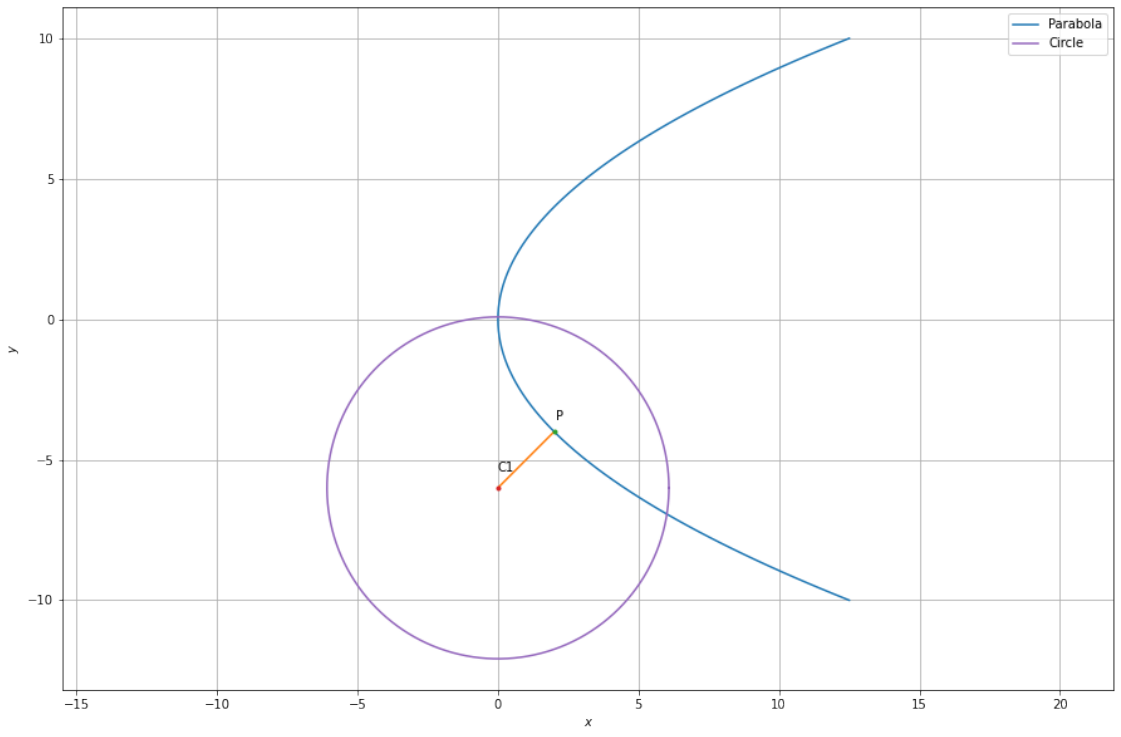
```
The product of the perpendicular distances between Tangent and Focii is
8.999999999999998
```



```
[5]: fig = plt.figure(figsize=(15,10))
     ax = fig.add_subplot(111, aspect='equal')


     C1 = np.array([0,-6])
     c = 1
     r = np.sqrt(((np.linalg.norm(C1)**2) + c))
     print(r)

     len = 1000000
     x_circ = np.zeros((2,len))
     thetha = np.linspace(0,2*np.pi,len)
     x_circ[0,:] = r *np.cos(thetha)
     x_circ[1,:] = r *np.sin(thetha)
     x_circ = (x_circ.T + C1).T

     len= 99
     V = np.array(([0,0],[0,1]))
     y = np.linspace(-10,10,len)
     x = np.power(y,2)/8
     P = np.vstack((x,y))
     m = np.array([1,1])
     c1 = 6
```

```
G = np.array([2,-4])
T = line_gen(C1,G)

# ~ H  =line_gen(T,O)
# ~ k =np.linalg.norm(T - C1)
# ~ print("The minimum distance between Parabola and Center of origin is",k)
# ~ plt.plot(H[0,:],H[1,:])

# ~ len = 1000000
# ~ x_circ1 = np.zeros((2,len))
# ~ x_circ1[0,:] = k *np.cos(thetha)
# ~ x_circ1[1,:] = r *np.sin(thetha)
# ~ x_circ1 = (x_circ.T + T).T
# ~ plt.plot(x_circ1[0,:],x_circ1[1,:])

plt.plot(P[0,:],P[1,:],label="Parabola")
plt.plot(T[0,:],T[1,:])
P = P.T
plt.plot(G[0],G[1],'.')
plt.text(G[0] * (1 + 0.03), G[1] * (1 - 0.1) , 'P')
plt.plot(C1[0],C1[1],'.')
plt.text(C1[0] * (1 + 0.03), C1[1] * (1 - 0.1) , 'C1')
plt.plot(x_circ[0,:],x_circ[1,:],label="Circle")

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

ax.plot()
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.legend(loc='best')
plt.grid()
plt.axis('equal')
plt.show()
```

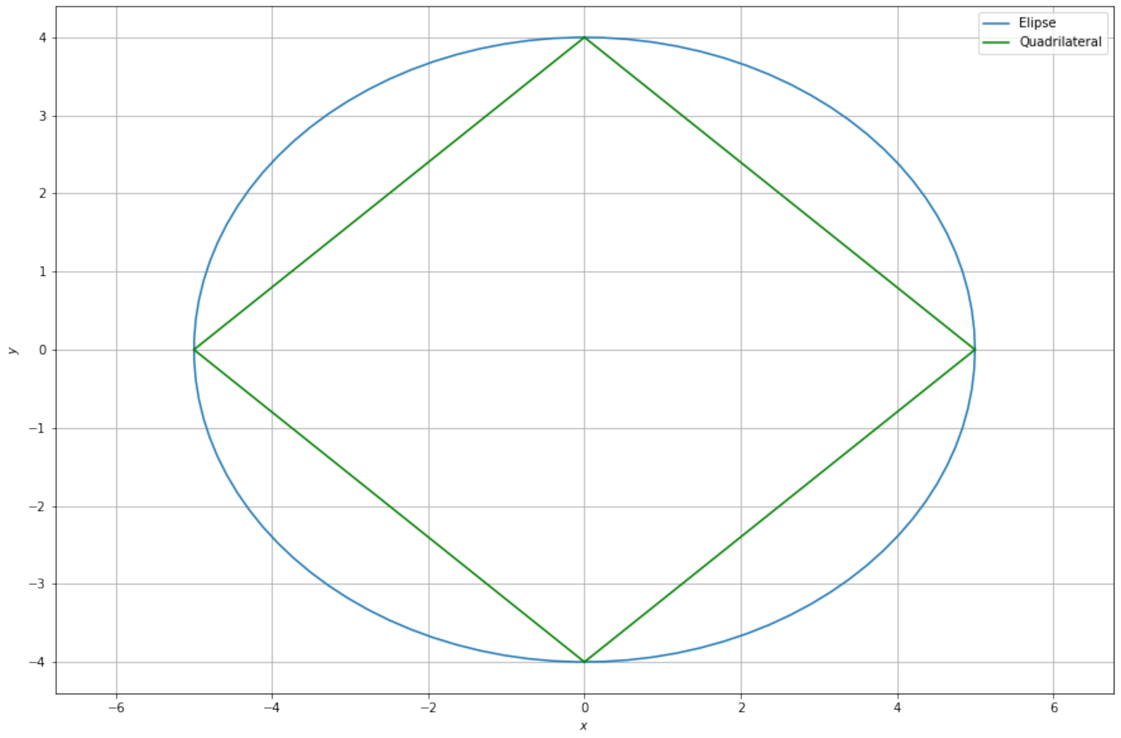
```
6.082762530298219
```



```
[6]: fig = plt.figure(figsize=(15,10))
     ax = fig.add_subplot(111, aspect='equal')

     e = 3/5
     d = 6
     a = d/(2*e)
     b = a*np.sqrt(1 - e**2)
     are = 2*a*b
     print("The area of Quadrilateral formed by vertices of Elipse",are)
     print(a)
     print(b)


     len = 100
     x_elips = np.zeros((2,len))
     thetha = np.linspace(0,2*np.pi,len)
     x_elips[0,:] = a*np.cos(thetha)
     x_elips[1,:] = b*np.sin(thetha)

     plt.plot(x_elips[0,:],x_elips[1,:],label= 'Elipse')

     B1 = np.array([0,b])
     B2 = np.array([0,-1*b])
```

```
A1 = np.array([a,0])
A2 = np.array([-1*a,0])

l1 = line_gen(A1,B1)
l4 = line_gen(A1,B2)
l3 = line_gen(A2,B2)
l2 = line_gen(B1,A2)
plt.plot(l1[0,:],l1[1,:],color='g',label="Quadrilateral")
plt.plot(l2[0,:],l2[1,:],color='g')
plt.plot(l3[0,:],l3[1,:],color='g')
plt.plot(l4[0,:],l4[1,:],color='g')


ax.plot()
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.legend(loc='best')
plt.grid()
plt.axis('equal')
plt.show()
```

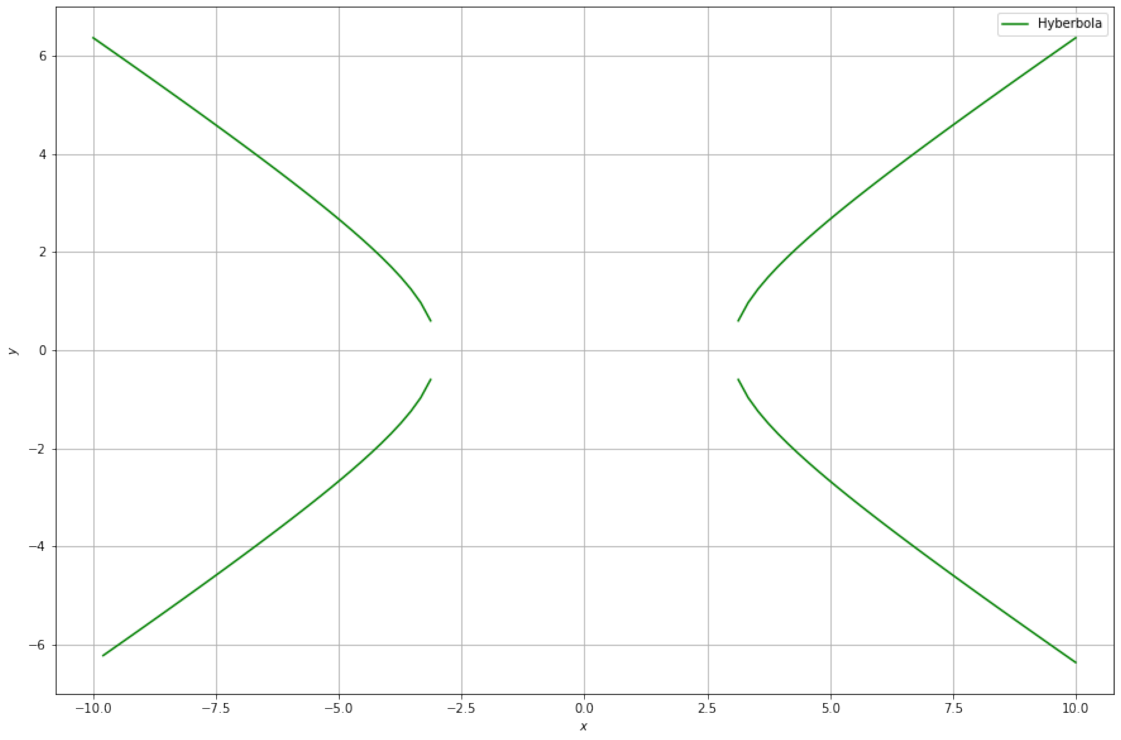
```
The area of Quadrilateral formed by vertices of Elipse 40.0
5.0
4.0
```



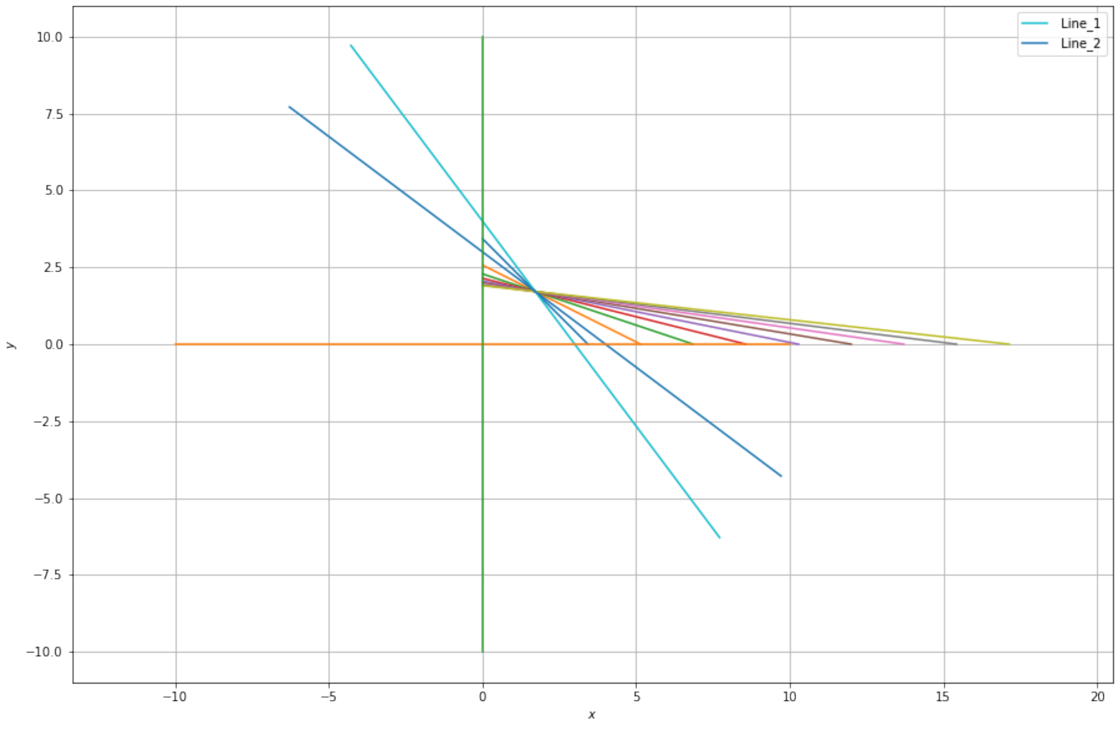
```
[7]: fig = plt.figure(figsize=(15,10))
     ax = fig.add_subplot(111, aspect='equal')



     e = np.roots([9,-18,5])
     if(e[0]>1):
             e =e[0]
     elif(e[0]>1):
             e = e[1]
     else:
             print("The Equation of ecentricity does not satisfy the condition of␣
      ↪hyperbola properties")

     S = np.array([5,0])

     a = np.linalg.norm(S[0])/e



     k = e**2 - 1
     A = a**2
     b = np.sqrt(A*k)
     print("a^2 - b^2 =",a**2  -  b**2)
     len = 100
     x = np.linspace(-10,10,len)
     y1 = np.sqrt(((x/a)**2 - 1)*b)
     y2 = -y1

     H = np.hstack((np.vstack((x,y1)),np.vstack((x,y2))))
     plt.plot(H[0,:len],H[1,:len],color='g',label="Hyberbola")
     plt.plot(H[0,len+1:],H[1,len+1:],color='g')








     ax.plot()
     plt.xlabel('$x$')
     plt.ylabel('$y$')
     plt.legend(loc='best')
     plt.grid()
     plt.axis('equal')
     plt.show()
```

```
a^2 - b^2 = -6.9999999999999964
```

```
<ipython-input-7-bb9af2898297>:24: RuntimeWarning: invalid value encountered in
sqrt
  y1 = np.sqrt(((x/a)**2 - 1)*b)
```



[8]:
```
fig = plt.figure(figsize=(15,10))
ax = fig.add_subplot(111, aspect='equal')


omat = np.array([[0,1],[-1,0]])
n_1 = np.array([4,3])
n_2 = np.array([3,4])
c_1 = np.array([12])
c_2 = np.array([12])
O = np.array([0,0])

m1 = omat@n_1
m2 = omat@n_2
V = np.vstack((n_1,n_2))
C = np.vstack((c_1,c_2))

P = np.linalg.inv(V)@C
P = P.reshape(2,)

for i in range(1,10):
        m = np.array([-1*i,1])
        l = line_dir_pt(m,P,-1*P[0]/m[0],-1*P[1]/m[1])
        plt.plot(l[0,:],l[1,:])

x = np.array([1,0])
y = np.array([0,1])
x_1 = line_dir_pt(x,0,-10,10)
y_1 = line_dir_pt(y,0,-10,10)
lg_1 = line_dir_pt(m1,P,-2,2)
lg_2 = line_dir_pt(m2,P,-2,2)

plt.plot(lg_1[0,:],lg_1[1,:],label=" Line_1")
plt.plot(lg_2[0,:],lg_2[1,:],label=" Line_2")
plt.plot(x_1[0,:],x_1[1,:])
plt.plot(y_1[0,:],y_1[1,:])



ax.plot()
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.legend(loc='best')
plt.grid()
plt.axis('equal')
plt.show()
```



```
[ ]:
```