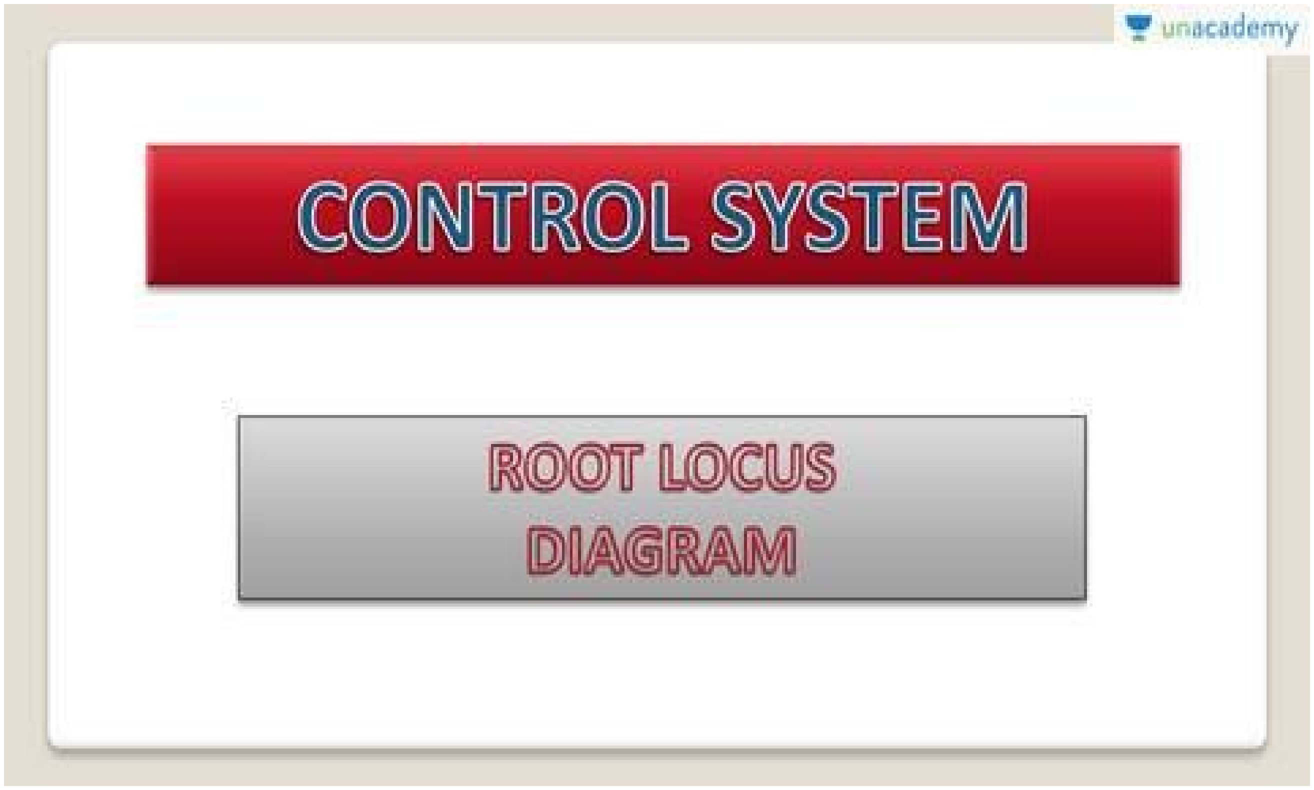

CONTROL SYSTEM

ROOT LOCUS DIAGRAM

Root locus

—

إسم الطالب : أفنان موسى مبروك موسى عبدالله   رقم الجلوس : 15

المادة (الكود) : نظم التحكم الخطي (069yicc)   الفرقة : الثانية

Overview

Root locus analysis is a graphical method used in control theory and stability theory to examine how the roots of a system change with variation of a certain system parameter, commonly a gain within a feedback system. This is a technique used in the field of classical control theory developed by Walter R. Evans as a stability criterion which can determine the stability of the system. The root locus plots the poles of the closed loop transfer function as a function of a gain parameter in the complex s-plan.

Goals

1. Draw poles (there are no zeros).
2. Draw the asymptotes (dotted lines).
3. Find break away points between any two real poles (show them in the output figure).
4. Find the intersections with the imaginary axis using Routh.
5. Find Departure angles for complex poles: departure (angle) + sum(poles_angles) = 180 (using the distances between poles and triangles geometry, angles can be computed, and show the found angles in the drawing).
6. Draw a figure of the root locus containing all information.

Source Code

https://drive.google.com/file/d/10X1lKT5OHQFNDKZC059crgV0T2pH228w/view?usp=sharing

Design

➢ After the user inserts The Number of Poles and The element of Poles :

- The **Main** Method calls Method **PlotPoles** which make the centre of axises in the centre of figure and plot all Poles.
- The function **DrawAsymptotes** calculates the center of Asymptotes and angle of them by calling **CenterAsymptotes** function and **AngleOfAsymptotes** function ,then draws Asymptotes lines.
- **The BreakAwayPoint** Method calculates the Points of Break away and draws it .
- The **ImagAxisCrossing** function calls a **Routh** function to create a Routh table then find the K value and root which cross the imaginary axises.
- The **AngleOfDeparture** function call **angle** function to calculate the angles and draw it.

Main Data Structures

I. Array Two Dimensional

➢ To create and store the Routh Table .
➢ To store the Departure Angles For Complex Poles.

II. Array OneDimensional

➢ To store the specific angles of Asymptotic lines .
➢ To store the Poles which users input them.

III. Vector

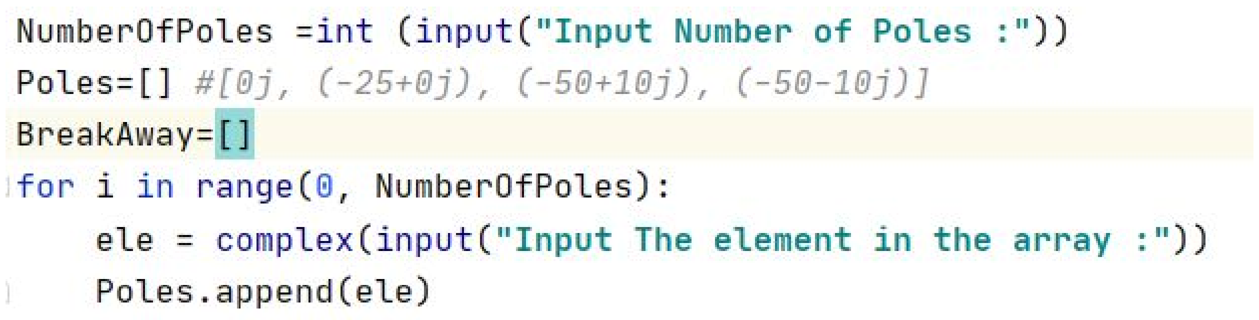
➢ To store the points of the Intersections with the Imaginary axis

Algorithms description

I. Take the input

➢ Take the input **NumberOf Poles** as **int** from the user ,create **Poles array** which insert in it the **complex number**

- If the input is 0, -25 this for loop convert it to `0j, (-25+0j)`



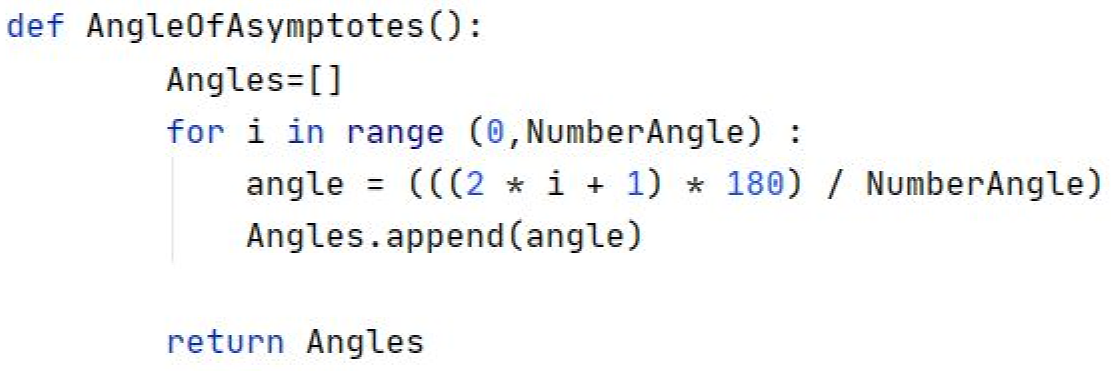
```
NumberOfPoles =int (input("Input Number of Poles :"))
Poles=[] #[0j, (-25+0j), (-50+10j), (-50-10j)]
BreakAway=[]
for i in range(0, NumberOfPoles):
    ele = complex(input("Input The element in the array :"))
    Poles.append(ele)
```

II. Basic function

➢ Calculate the **angle of Asymptotes l**ine by the equation

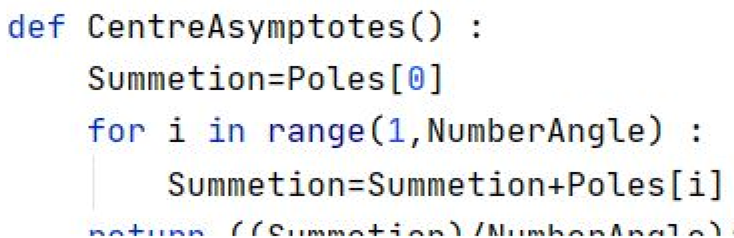
$$\phi_l = \frac{180^\circ + (l-1)360^\circ}{P-Z}, l = 1, 2, \ldots, P-Z$$

In Method AngleOfAsymptotes as shown in the photo :



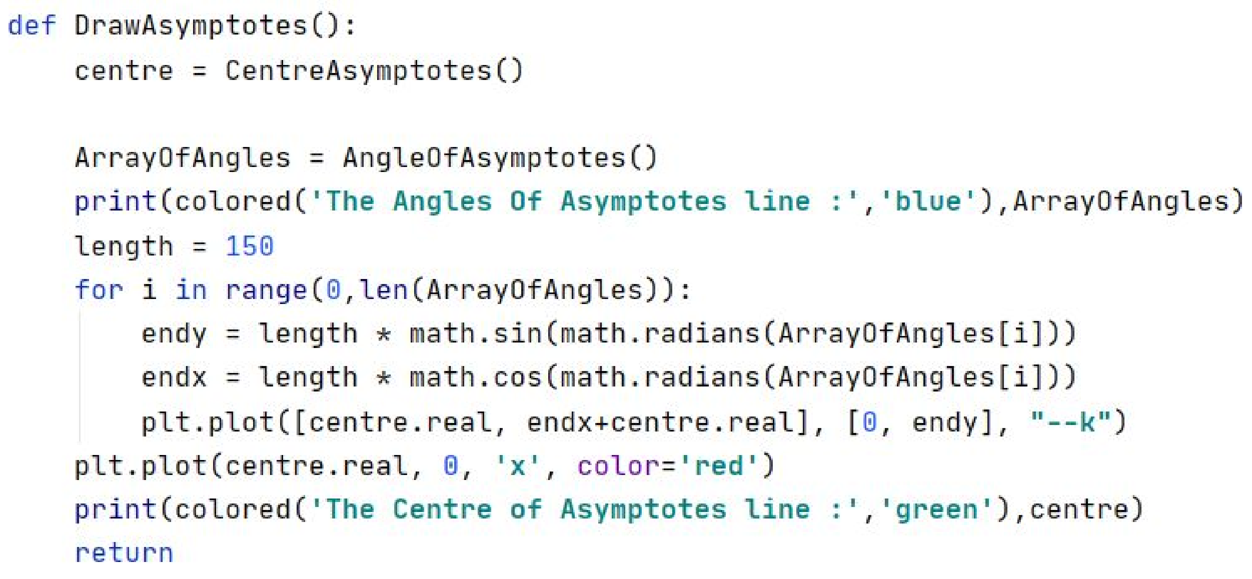
```
def AngleOfAsymptotes():
        Angles=[]
        for i in range (0,NumberAngle) :
            angle = (((2 * i + 1) * 180) / NumberAngle)
            Angles.append(angle)

        return Angles
```

➢ Calculate the **alpha** which is the center to draw the Asymptotes by the following equation $$\alpha = \frac{\sum_P - \sum_Z}{P-Z}$$ which P is poles and Z is Zeroes ,as we know this project design for no zeroes so the **Summation** is to **Poles** only as obvious in the snapped of code



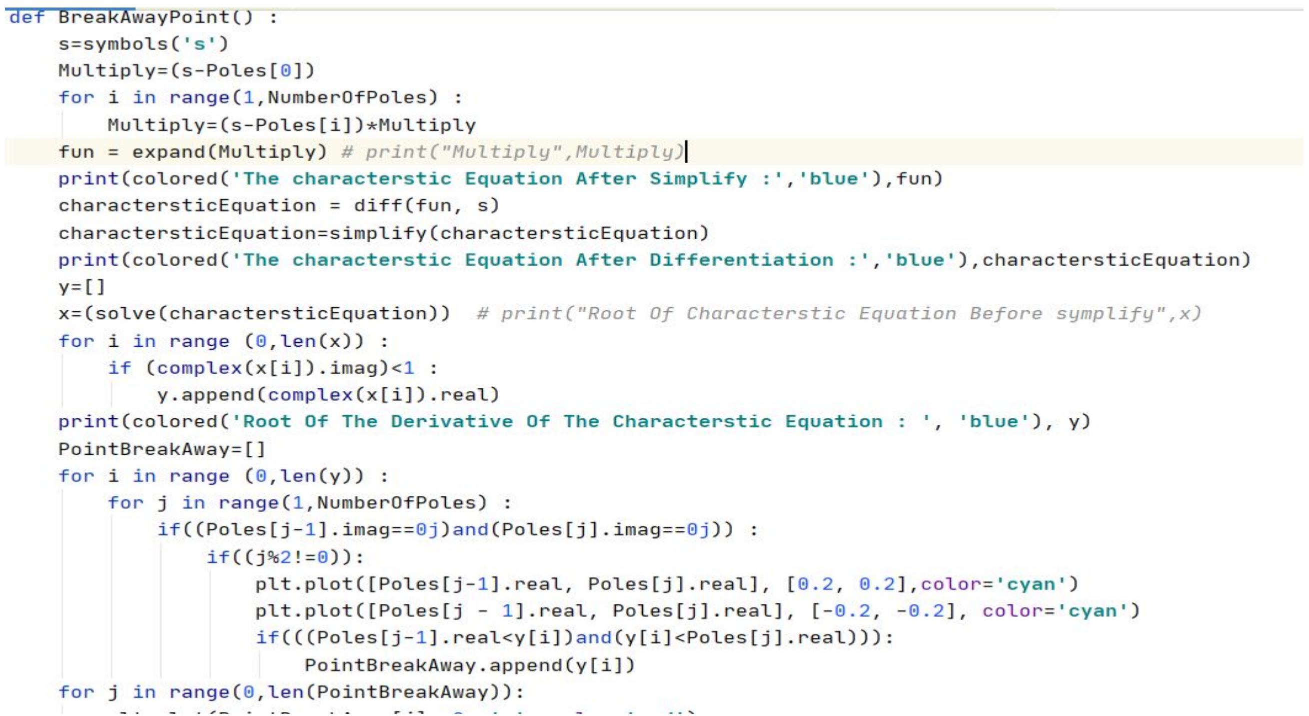
```
def CentreAsymptotes() :
    Summetion=Poles[0]
    for i in range(1,NumberAngle) :
        Summetion=Summetion+Poles[i]
```

- ➢ Draw the **Asymptotes** as dotted lines by using mathematical **Sin ,Con** to calculate the final point in the line depending on the Assumption value of length .



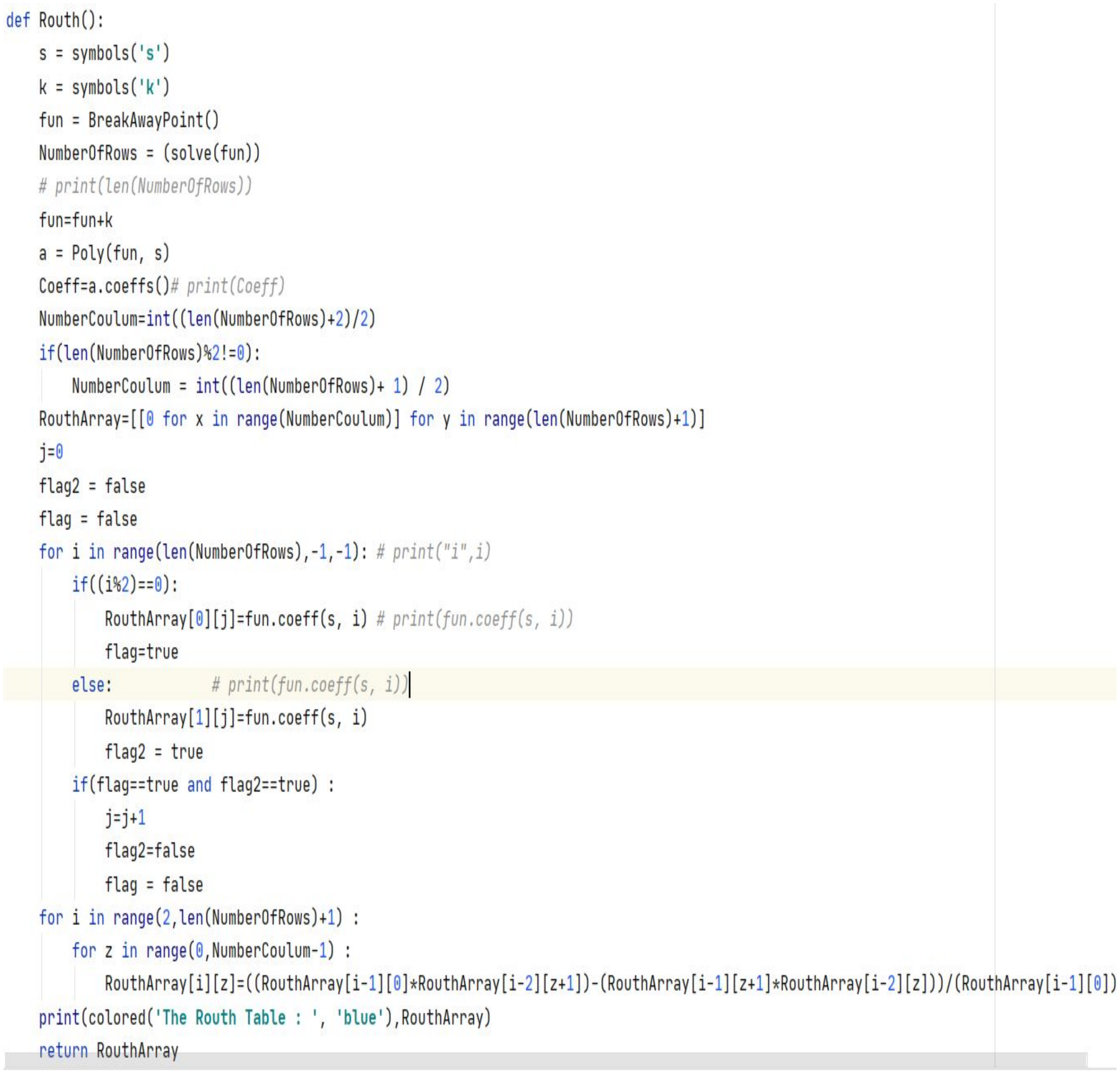
```
def DrawAsymptotes():
    centre = CentreAsymptotes()

    ArrayOfAngles = AngleOfAsymptotes()
    print(colored('The Angles Of Asymptotes line :','blue'),ArrayOfAngles)
    length = 150
    for i in range(0,len(ArrayOfAngles)):
        endy = length * math.sin(math.radians(ArrayOfAngles[i]))
        endx = length * math.cos(math.radians(ArrayOfAngles[i]))
        plt.plot([centre.real, endx+centre.real], [0, endy], "--k")
    plt.plot(centre.real, 0, 'x', color='red')
    print(colored('The Centre of Asymptotes line :','green'),centre)
    return
```

- ➢ To calculate the break away point we must find the equation of characteristic and differentiation it then solve it to get the points and felteration this point according to the Poles my make for loop ,Then draw it .



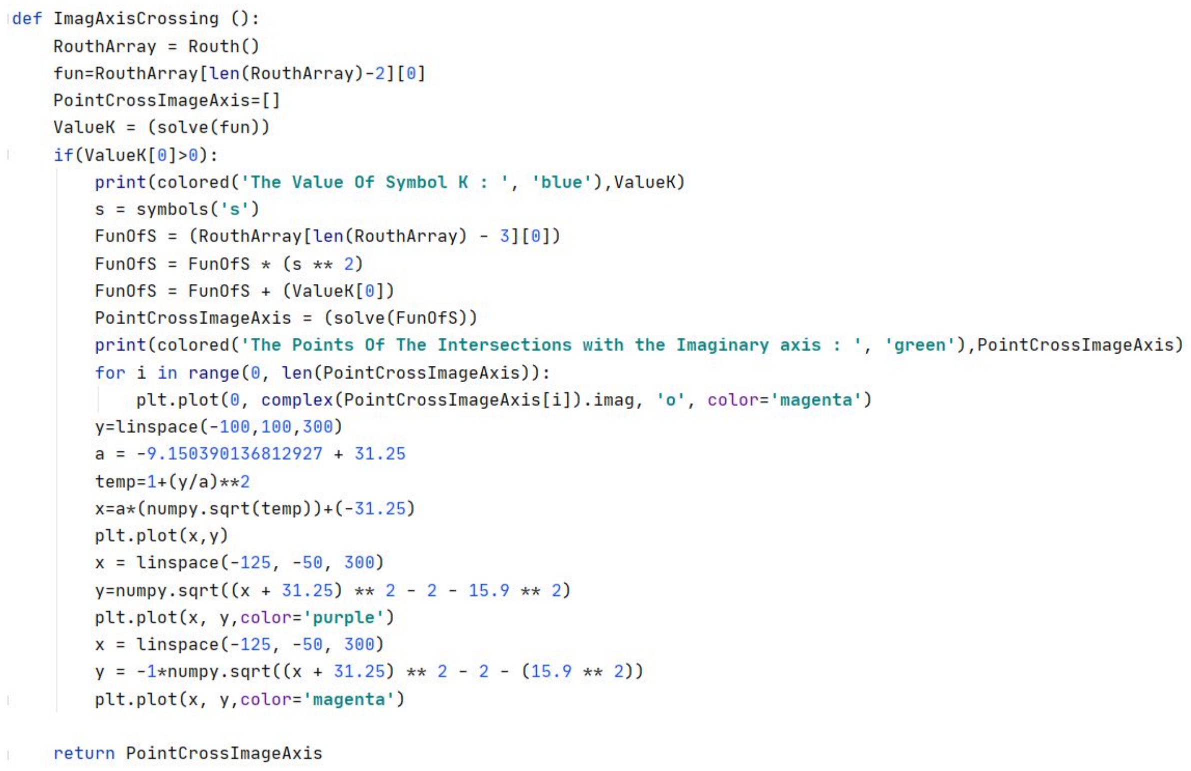
```
def BreakAwayPoint() :
    s=symbols('s')
    Multiply=(s-Poles[0])
    for i in range(1,NumberOfPoles) :
        Multiply=(s-Poles[i])*Multiply
    fun = expand(Multiply) # print("Multiply",Multiply)
    print(colored('The characterstic Equation After Simplify :','blue'),fun)
    characactersticEquation = diff(fun, s)
    characactersticEquation=simplify(characactersticEquation)
    print(colored('The characterstic Equation After Differentiation :','blue'),characactersticEquation)
    y=[]
    x=(solve(characactersticEquation))  # print("Root Of Characterstic Equation Before symplify",x)
    for i in range (0,len(x)) :
        if (complex(x[i]).imag)<1 :
            y.append(complex(x[i]).real)
    print(colored('Root Of The Derivative Of The Characterstic Equation : ', 'blue'), y)
    PointBreakAway=[]
    for i in range (0,len(y)) :
        for j in range(1,NumberOfPoles) :
            if((Poles[j-1].imag==0j)and(Poles[j].imag==0j)) :
                if((j%2!=0)):
                    plt.plot([Poles[j-1].real, Poles[j].real], [0.2, 0.2],color='cyan')
                    plt.plot([Poles[j - 1].real, Poles[j].real], [-0.2, -0.2], color='cyan')
                    if(((Poles[j-1].real<y[i])and(y[i]<Poles[j].real))):
                        PointBreakAway.append(y[i])
    for j in range(0,len(PointBreakAway)):
```

➢ Routh function apply the rule to calculate the **Routh Table** by store the first two line and then apply the general teaching as shown in code



```
def Routh():
    s = symbols('s')
    k = symbols('k')
    fun = BreakAwayPoint()
    NumberOfRows = (solve(fun))
    # print(len(NumberOfRows))
    fun=fun+k
    a = Poly(fun, s)
    Coeff=a.coeffs()# print(Coeff)
    NumberCoulum=int((len(NumberOfRows)+2)/2)
    if(len(NumberOfRows)%2!=0):
        NumberCoulum = int((len(NumberOfRows)+ 1) / 2)
    RouthArray=[[0 for x in range(NumberCoulum)] for y in range(len(NumberOfRows)+1)]
    j=0
    flag2 = false
    flag = false
    for i in range(len(NumberOfRows),-1,-1): # print("i",i)
        if((i%2)==0):
            RouthArray[0][j]=fun.coeff(s, i) # print(fun.coeff(s, i))
            flag=true
        else:            # print(fun.coeff(s, i))
            RouthArray[1][j]=fun.coeff(s, i)
            flag2 = true
        if(flag==true and flag2==true) :
            j=j+1
            flag2=false
            flag = false
    for i in range(2,len(NumberOfRows)+1) :
        for z in range(0,NumberCoulum-1) :
            RouthArray[i][z]=((RouthArray[i-1][0]*RouthArray[i-2][z+1])-(RouthArray[i-1][z+1]*RouthArray[i-2][z]))/(RouthArray[i-1][0])
    print(colored('The Routh Table : ', 'blue'),RouthArray)
    return RouthArray
```

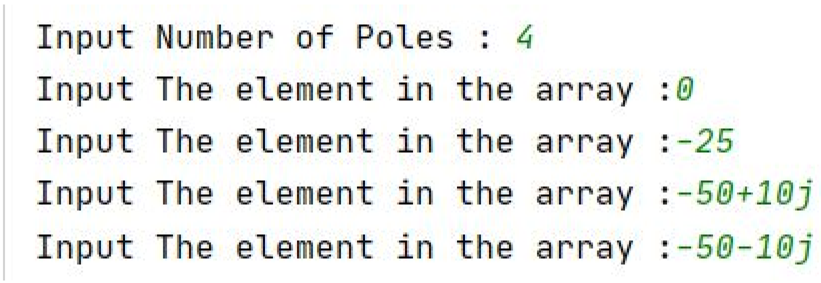
➢ Finally calculate the **departure angle** and draw the **curves** depend on this angle :



```python
def ImagAxisCrossing ():
    RouthArray = Routh()
    fun=RouthArray[len(RouthArray)-2][0]
    PointCrossImageAxis=[]
    ValueK = (solve(fun))
    if(ValueK[0]>0):
        print(colored('The Value Of Symbol K : ', 'blue'),ValueK)
        s = symbols('s')
        FunOfS = (RouthArray[len(RouthArray) - 3][0])
        FunOfS = FunOfS * (s ** 2)
        FunOfS = FunOfS + (ValueK[0])
        PointCrossImageAxis = (solve(FunOfS))
        print(colored('The Points Of The Intersections with the Imaginary axis : ', 'green'),PointCrossImageAxis)
        for i in range(0, len(PointCrossImageAxis)):
            plt.plot(0, complex(PointCrossImageAxis[i]).imag, 'o', color='magenta')
        y=linspace(-100,100,300)
        a = -9.150390136812927 + 31.25
        temp=1+(y/a)**2
        x=a*(numpy.sqrt(temp))+(-31.25)
        plt.plot(x,y)
        x = linspace(-125, -50, 300)
        y=numpy.sqrt((x + 31.25) ** 2 - 2 - 15.9 ** 2)
        plt.plot(x, y,color='purple')
        x = linspace(-125, -50, 300)
        y = -1*numpy.sqrt((x + 31.25) ** 2 - 2 - (15.9 ** 2))
        plt.plot(x, y,color='magenta')

    return PointCrossImageAxis
```

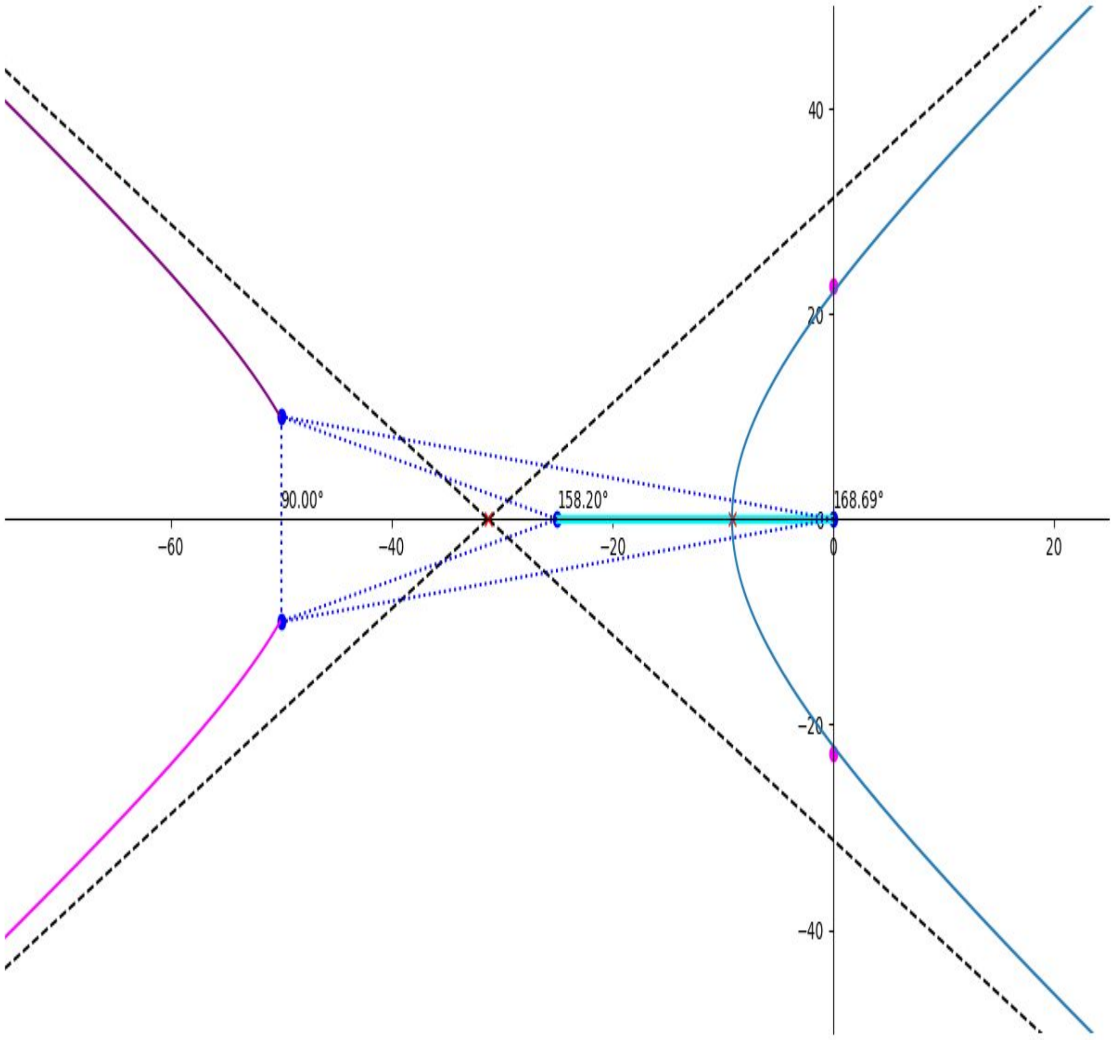
Sample Run

- The user input the Input as **int** to Number of poles and **complex or int** for another

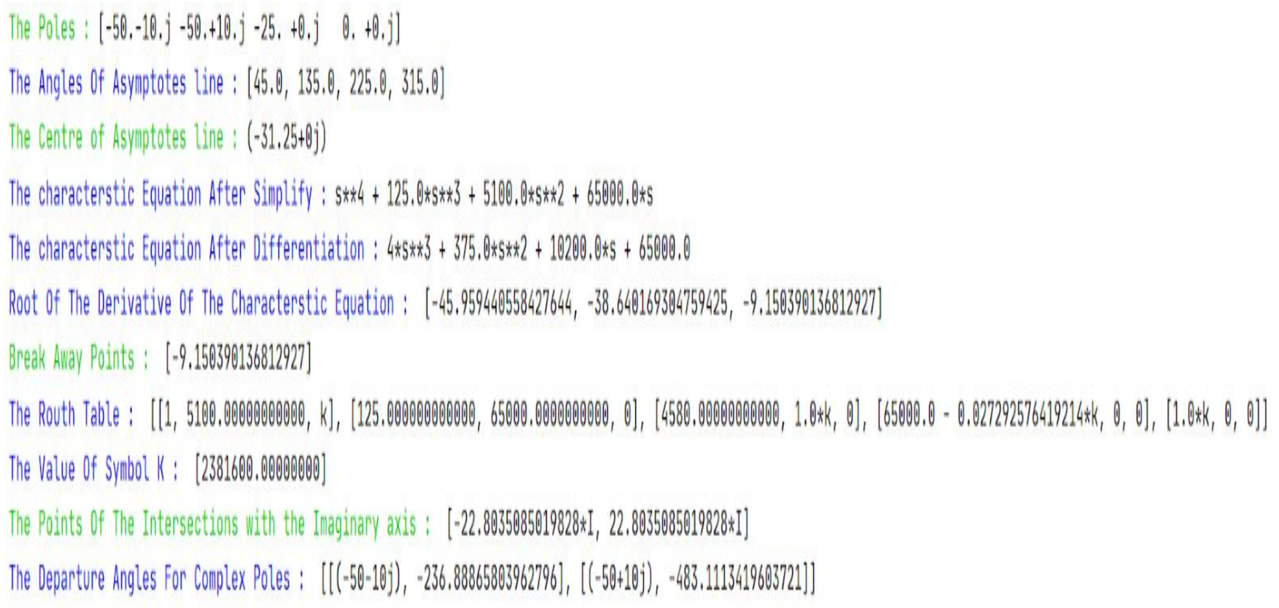


```
Input Number of Poles : 4
Input The element in the array :0
Input The element in the array :-25
Input The element in the array :-50+10j
Input The element in the array :-50-10j
```

- The Output figure Be as shown ..

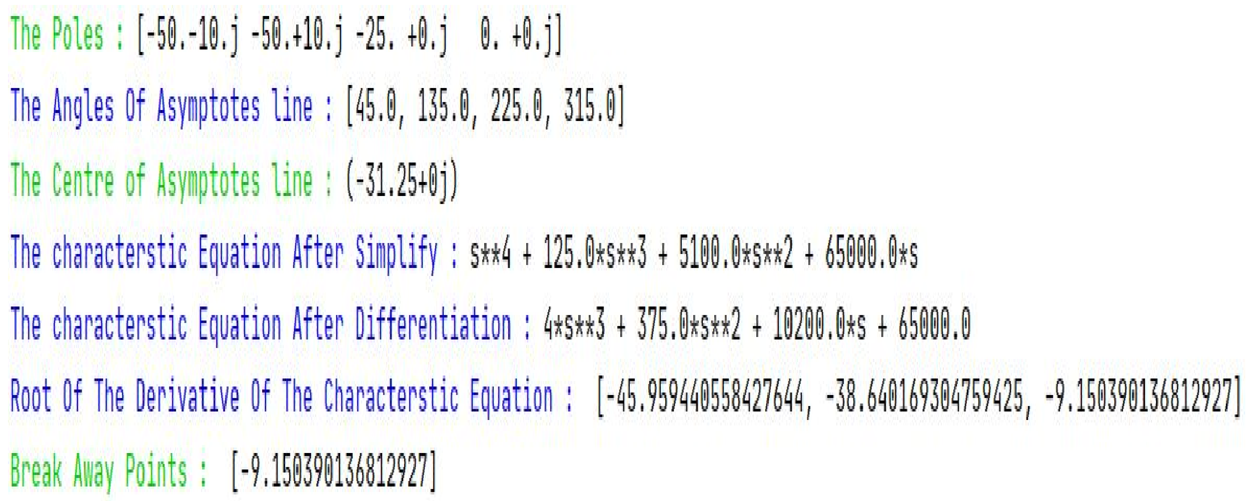


- **The Output Data result is :**
  - ➢ The Poles : [-50.-10.j -50.+10.j -25. +0.j 0. +0.j]
  - ➢ The Angles Of Asymptotes line : [45.0, 135.0, 225.0, 315.0]
  - ➢ The Centre of Asymptotes line : (-31.25+0j)
  - ➢ The characteristic Equation After Simplify : s**4 + 125.0*s**3 + 5100.0*s**2 + 65000.0*s
  - ➢ The characteristic Equation After Differentiation : 4*s**3 + 375.0*s**2 + 10200.0*s + 65000.0
  - ➢ Root Of The Derivative Of The Characteristic Equation : [-45.959440558427644, -38.640169304759425, -9.150390136812927]
  - ➢ Break Away Points : [-9.150390136812927]
  - ➢ The Routh Table : [[1, 5100.00000000000, k], [125.000000000000, 65000.0000000000, 0], [4580.00000000000, 1.0*k, 0], [65000.0 - 0.027292576419214*k, 0, 0], [1.0*k, 0, 0]]
  - ➢ The Value Of Symbol K : [2381600.00000000]
  - ➢ The Points Of The Intersections with the Imaginary axis : [-22.8035085019828*I, 22.8035085019828*I]
  - ➢ The Departure Angles For Complex Poles : [[(-50-10j), -236.88865803962796], [(-50+10j), -483.1113419603721]]

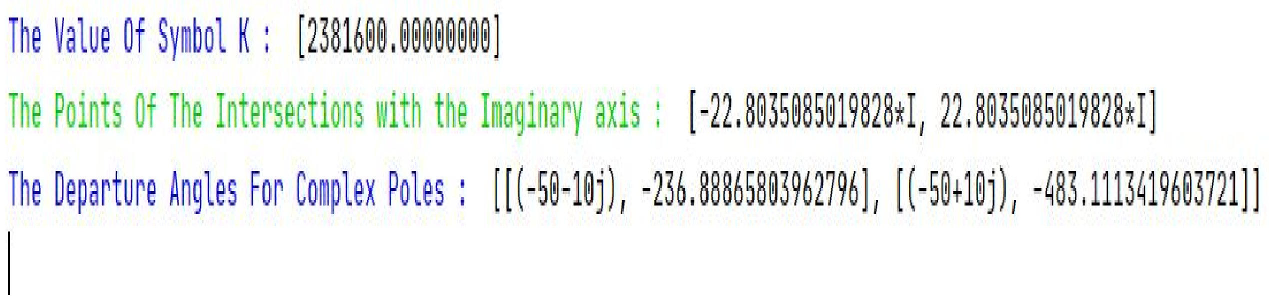


```
The Poles : [-50.-10.j -50.+10.j -25. +0.j   0. +0.j]
The Angles Of Asymptotes line : [45.0, 135.0, 225.0, 315.0]
The Centre of Asymptotes line : (-31.25+0j)
The characterstic Equation After Simplify : s**4 + 125.0*s**3 + 5100.0*s**2 + 65000.0*s
The characterstic Equation After Differentiation : 4*s**3 + 375.0*s**2 + 10200.0*s + 65000.0
Root Of The Derivative Of The Characterstic Equation :  [-45.959440558427644, -38.640169304759425, -9.150390136812927]
Break Away Points :  [-9.150390136812927]
The Routh Table :  [[1, 5100.00000000000, k], [125.000000000000, 65000.0000000000, 0], [4580.00000000000, 1.0*k, 0], [65000.0 - 0.027292576419214*k, 0, 0], [1.0*k, 0, 0]]
The Value Of Symbol K :  [2381600.00000000]
The Points Of The Intersections with the Imaginary axis :  [-22.8035085019828*I, 22.8035085019828*I]
The Departure Angles For Complex Poles :  [[(-50-10j), -236.88865803962796], [(-50+10j), -483.1113419603721]]
```

➢ The photo may be obvious data in those.



```
The Poles : [-50.-10.j -50.+10.j -25. +0.j   0. +0.j]

The Angles Of Asymptotes line : [45.0, 135.0, 225.0, 315.0]

The Centre of Asymptotes line : (-31.25+0j)

The characterstic Equation After Simplify : s**4 + 125.0*s**3 + 5100.0*s**2 + 65000.0*s

The characterstic Equation After Differentiation : 4*s**3 + 375.0*s**2 + 10200.0*s + 65000.0

Root Of The Derivative Of The Characterstic Equation :  [-45.959440558427644, -38.640169304759425, -9.150390136812927]

Break Away Points :  [-9.150390136812927]
```



```
The Value Of Symbol K :  [2381600.00000000]

The Points Of The Intersections with the Imaginary axis :  [-22.8035085019828*I, 22.8035085019828*I]

The Departure Angles For Complex Poles :  [[(-50-10j), -236.88865803962796], [(-50+10j), -483.1113419603721]]
```