



ESCUELA SUPERIOR DE CÓMPUTO

MATEMATICAS AVANZADAS PARA LA INGENIERIA

# Raices $n$ -esimas de un complejo

*por:*

*López Manríquez Ángel*

profesor

Olvera Andana Miguel

17 de junio de 2019

# 1. Introduccion

Para hacer del programa que grafique las raices de un complejo  $z$ , se hara uso de los siguientes temas.

**Teorema 1.1.** *Raices  $n$ -esimas de un complejo*

Sea  $z = (a, b)$ , entonces  $z^{\frac{1}{n}}$  tiene exactamente  $n$  raices, de la forma

$$w_k = |z|^{1/n} \text{cis}\left(\frac{\theta + 2k\pi}{n}\right) \quad k \in \{0, 1, \dots, n-1\}$$

# 2. Planteamiento del problema

Hacer un programa en cualquier lenguaje que grafique la raiz  $n$ -esima de un complejo.

# 3. Diseño e implementacion de la solucion

Para dar solucion al problema, usamos el teorema de la raiz  $n$ -esima de la unidad como se muestra en el siguiente codigo

---

```

1
2 # Programa que grafica las raices n-esimas de un complejo
3 # y las muestra en el plano complejo
4
5 # Importamos las bibliotecas para graficar en el plano
6 import numpy as np
7 import matplotlib.pyplot as plt
8 # Para el obtener la arcotangente de un angulo
9 import math
10
11 # Obtenemos la parte real e imaginaria para z = a + bi
12 a = raw_input ("re {z} = ")
13 a = float(a)
14 b = raw_input ("im {z} = ")
15 b = float(b)
16 n = raw_input ("n (z ^ (1 / n)) = ")
17 # Obtenemos a n, para la raiz n-esima
18 n = int(n)
19 # Hallamos el modulo de z
20 r = math.sqrt (a ** 2 + b ** 2)
21 # Hallamos el angulo formado por el complejo con respecto
22 # al eje real
23 theta = math.atan2 (b, a)
24 # Le calculamos la raiz n-esima a |z|
25 r = r ** (1 / n)
26 # Declaramos dos listas para guardar las coordenadas de
27 # los puntos
28 x = []
29 y = []

```

```

30 # Obtenemos los n - 1 angulos, posteriormente agregamos
31 #a las listas las coordenadas rectangulares del complejo
32 for k in range (0, n):
33     phi = float ((2 * k * math.pi + theta) / n)
34     x.append (r * math.cos (phi))
35     y.append (r * math.sin (phi))
36 # Graficamos los puntos
37 plt.scatter (x, y)
38 plt.show ()

```

---

## 4. Funcionamiento

Para  $(-1, 0)$ ,  $n = 4$

```

angel@lambda: roots >> make [22:03:40]
python root.py
re {z} = -1
im {z} = 0
n (z ^ (1 / n)) = 4

```



Para  $(3, 4)$ ,  $n = 32$

```
angel@lambda: roots » make [22:32:04]  
python root.py  
re {z} = 3  
im {z} = 4  
n (z ^ (1 / n)) = 32
```

