Hélio Hideki Assakura Moreira

# MAC 0323 – EP1
# Bacias de Newton

**Instituto de Matemática e Estatística - USP**

**São Paulo – 2015**

## Objetivo do EP:

Estudar o método de Newton, usado para encontrar raízes complexas de equações. Para isso, criamos uma imagem para cada equação, e para cada raiz encontrada, definimos uma cor, com intensidade dependendo da quantidade de iterações realizadas para encontrar tal raiz.

## Novos tipos:

- **Image:**

Para usar o recurso de zoom (que será explicado posteriormente), foi criado um novo tipo, chamado Image. Ele contém os atributos:

**int maxI**: Número máximo de iterações do Método de Newton para que a intensidade da cor seja máxima.
**double x**: Parte real do centro da figura.
**double y**: Parte imaginaria do centro.
**double xsize**: Região x do plano a ser considerado.
**double ysize**: Região y do plano a ser considerado.
**int M**: Largura da imagem.
i**nt N**: Altura da imagem.
**Complex[] Roots**: vetor com as raízes encontradas (pode receber no máximo 2000 raízes).
**Color[] Colors**: vetor com as cores correspondentes às raízes.
**int Mnew**: largura considerada ao realizar zoom.
**int Nnew**: altura considerada ao realizar zoom.

Foi estabelecido o máximo de 2000 raízes, que já são suficientes para ter uma imagem clara do processo de convergência das raízes.

## Classes usadas:

- **Newton:**

    - **FindRoot:**

        Com parâmetros iguais aos descritos no enunciado, retorna uma aproximação de uma das raízes da função. Porém, algumas vezes, o complexo (z0) não converge para nenhuma raiz. Por isso, foram estabelecidas algumas condições de parada em *FindRoot*:

        1. Depois de 1000 iterações.
        2. Caso $\frac{f(z0)}{f'(z0)}$ seja menor que um $\varepsilon$, determinado na própria classe.
        3. $f'(z0)$ for extremamente pequeno, ou seja, $\frac{f(z0)}{f'(z0)}$ é um número muito alto.

- **NewtonBasins:**

    - **Draw:**

        Diferente do enunciado, **draw** recebe uma HolomorphicFunction e um objeto Image, que contém os parâmetros citados no enunciado. Para cada pixel da imagem a ser gerada, executamos FindRoot, e dependendo da raiz encontrada, pintamos esse pixel de uma cor.

        Depois da imagem ser gerada, podemos ampliar determinada região. A imagem ampliada tem mesmas dimensões da imagem original, e ela pode ser ampliada diversas vezes.

        Além do zoom, há a possibilidade de salvar a imagem no diretório atual.

- **ImgManipulation:**

    - **Zooming:**

        Ao pressionar o mouse 2 vezes, formando um retângulo, uma nova imagem é gerada na tela, com as mesmas dimensões. Como podemos ampliá-la diversas vezes, para não perder a informação da parte do plano que estamos considerando, as posições anteriores dos cliques do usuário são guardadas em variáveis do tipo Point2D.

Depois dos cliques, o novo centro e a nova região do plano são calculados, e uma nova imagem é mostrada ao usuário.

- **ScreenShot:**

  Função usada para salvar a imagem. Ela é salva apertando a tecla "**9**". O nome do arquivo gerado é da forma seed _ maxI _ xc _ yc _ xsize _ ysize _ M _N.jpg.

- **Trig:**

Classe contendo as funções Fsin, Fcos, Ftan e Trans1, que correspondem, respectivamente, a $f(z) = sin(2\pi z) - c$ , $f(z) = cos(2\pi z) - c$ , $f(z) = tan(2\pi z) - c$ e $x^{\pi}$. Os argumentos são, nessa ordem:

**seed, maxI, x, y, xsize, ysize, M, N, a, b**

Depois de inseridos, uma mensagem é mostrada:

1: Sin (2*pi*z - c)
2: Cos (2*pi*z - c)
3: Tan (2*pi*z - c)
4: x^pi

Deve-se entrar com o numero correspondente à função desejada. Exemplo de entrada:

java Trig 42 20 0 0 1 1 1000, 1000, 2, 0
1

Assim, ele irá gerar a imagem correspondente a função seno, com esses argumentos.

- **PolyRoots:**

Essa classe gera uma imagem a partir da entrada das raízes de um polinômio. Os argumentos são:

**seed, maxI, x, y, xsize, ysize, M, N**

Então, deve-se entrar com as raízes, inserindo primeiro a parte real, depois a parte imaginária. Exemplo:

```
java PolyRoots 42 20 0 0 2 2 1000 1000
1 0
-1 0
0 1
0 -1
```

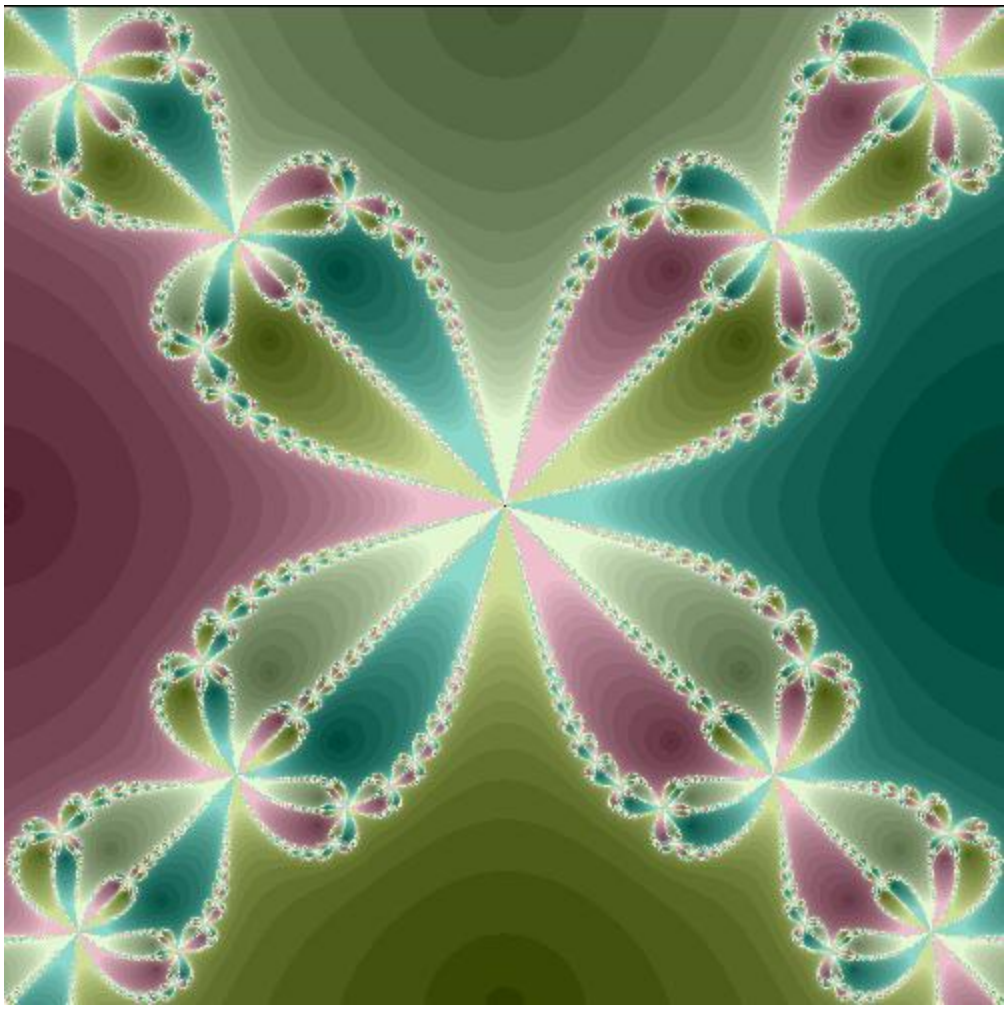
No exemplo, as raízes são: (1, 0), (-1, 0), (0, 1) e (0, -1), ou 1, -1, i, -i.

# Imagens Geradas

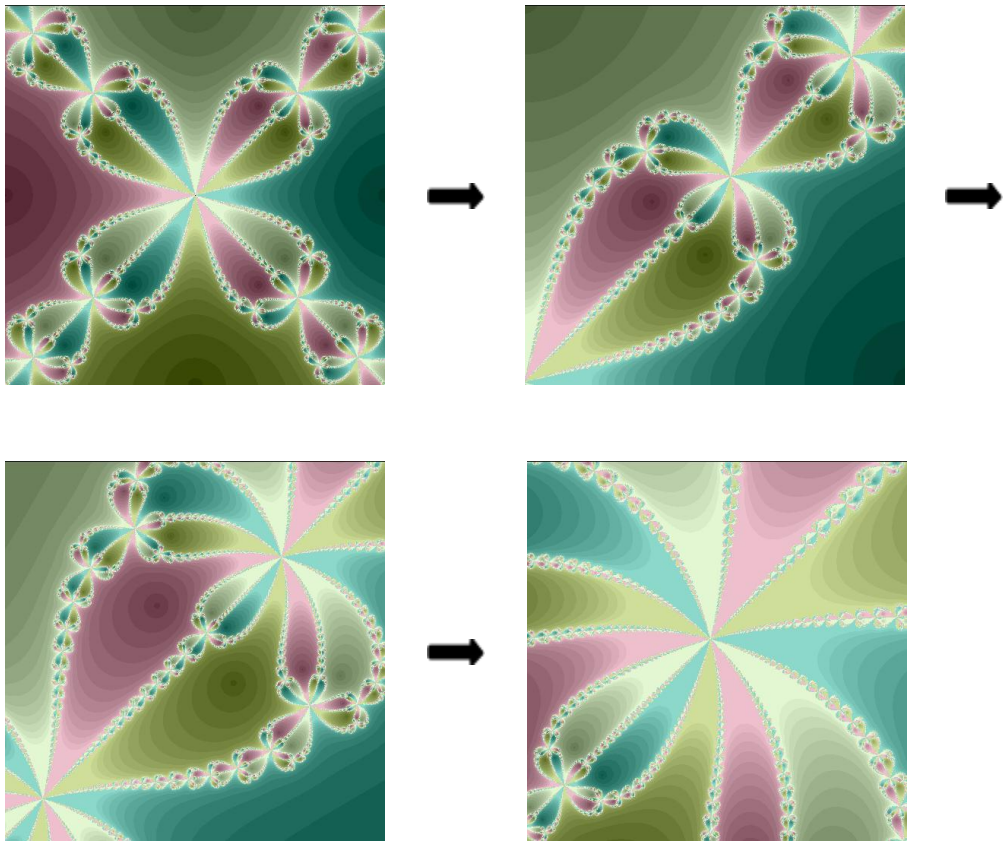
As imagens foram salvas com o uso da função ScreenShot
Link com as outras fotos:

https://drive.google.com/folderview?id=0B1ZLb5r4r2MVfmVuVFl5RU9TQmZXWjljQUU5a0VGdWRRcVh0aGxtcHFGWmY5REZJVC1wZmM&usp=sharing

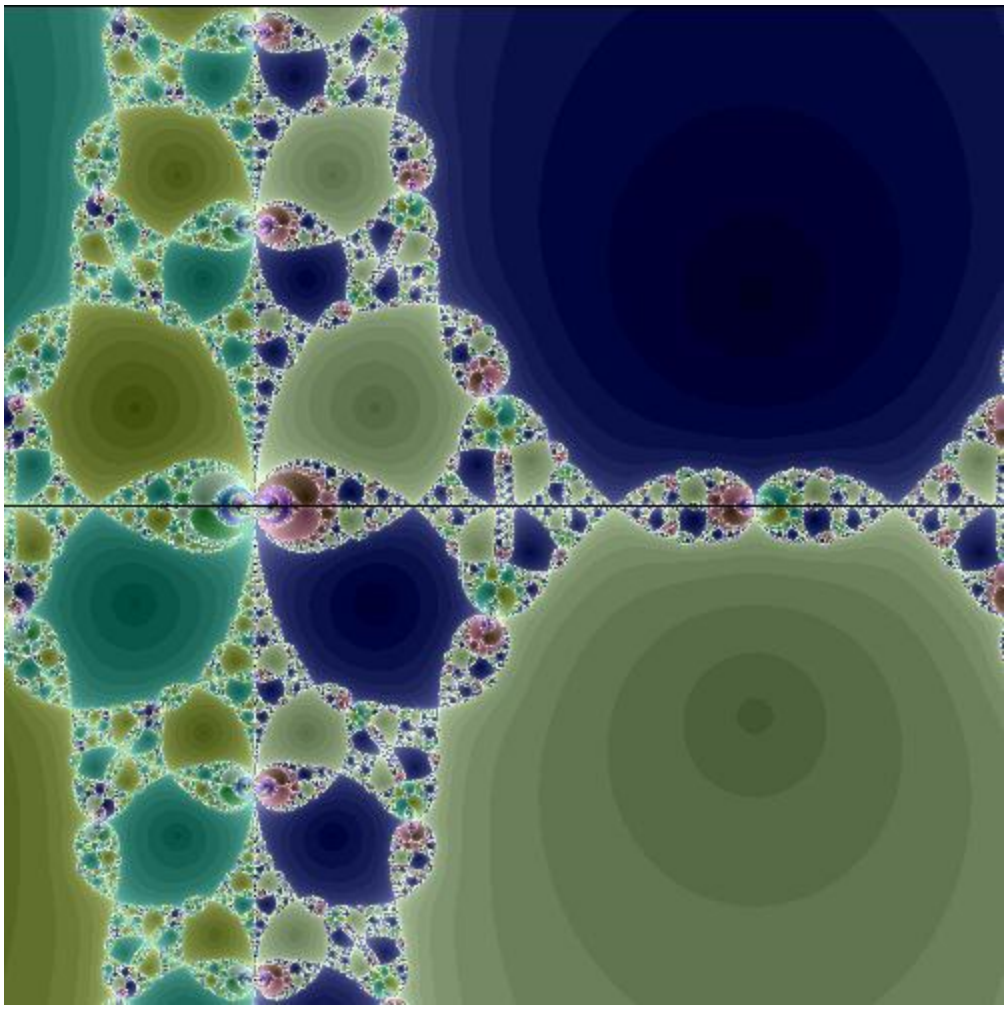
java PolyRoots 2024 20 0 0 2 2 500 500
1 0
-1 0
0 1
0 -1



***Zoom:***



java Trig 2024 20 0 0 1 1 500 500 2 0
1



***Zoom:***