MAC0210 - Exercício Programa 1

Professor: Ernesto G. Birgin
Monitor: Gustavo Estrela

1 Parte 1: Aritmética de Ponto Flutuante

Essa parte do EP consiste em resolver quatro exercícios sobre aritmética de ponto flutuante. Para solucionar os problemas é recomendado a leitura do livro "Numerical Computing with IEEE Floating Point Arithmetic", presente na bibliografia do curso. Todos os exercícios da parte 1 desse EP foram inspirados em exercícios do livro, indicados entre parênteses.

Você deve justificar todas as suas respostas. As soluções dos exercícios devem estar em um pdf que contém também os relatórios das partes 2 e 3 desse EP. Esse pdf deve ser impresso e entregue em sala de aula no dia 20 de setembro.

Questão 1 (3.11): Suponha que temos um sistema de representação de ponto flutuante com base 2 e,

$$\begin{aligned} & x = \pm S \times 2^E, \\ \text{com} \quad & S = (0.1 b_2 b_3 b_4 ... b_{24}), \\ \text{i.e,} \quad & \tfrac{1}{2} \leq S < 1 \end{aligned}$$

onde o expoente $-128 < E < 127$.
(a) Qual é o maior número de ponto flutuante desse sistema?
(b) Qual é o menor número de ponto flutuante positivo desse sistema?
(c) Qual é o menor inteiro positivo que não é exatamente representável nesse sistema?

Questão 2 (5.1): Qual é a representação do número $1/10$ no formato IEEE single para cada um dos quatro modos de arredondamento? E para os números $1 + 2^{-25}$ e 2^{130}?

Questão 3 (6.4): Qual é o maior número de ponto flutuante x tal que $1 \oplus x$ é exatamente 1, assumindo que o formato usado é IEEE single e modo de arredondamento para o mais próximo? E se o formato for IEEE double?

Questão 4 (6.8): Em aritmética exata, a soma é um operador comutativo e associativo. O operador de soma de ponto flutuante é commutativo? E associativo? Explique.

2 Parte 2: Método de Newton

Quando se aplica o Método de Newton a uma função com mais de uma raíz, temos que a raíz que será obtida pelo método depende do ponto inicial escolhido. Nesta parte do EP vamos extender o Método de Newton para um domínio complexo e estudar suas **bacias de convergência**, o conjunto de pontos iniciais que convergem para uma mesma raíz da função estudada.

Você deve implementar o Método de Newton em Octave e usar o script disponível para gerar uma imagem que mostra as bacias de convergência de uma função escolhida. Você vai notar que as imagens geradas formam fractais! Explicar o motivo da formação desses fractais não é necessário. Veja abaixo alguns exemplos.

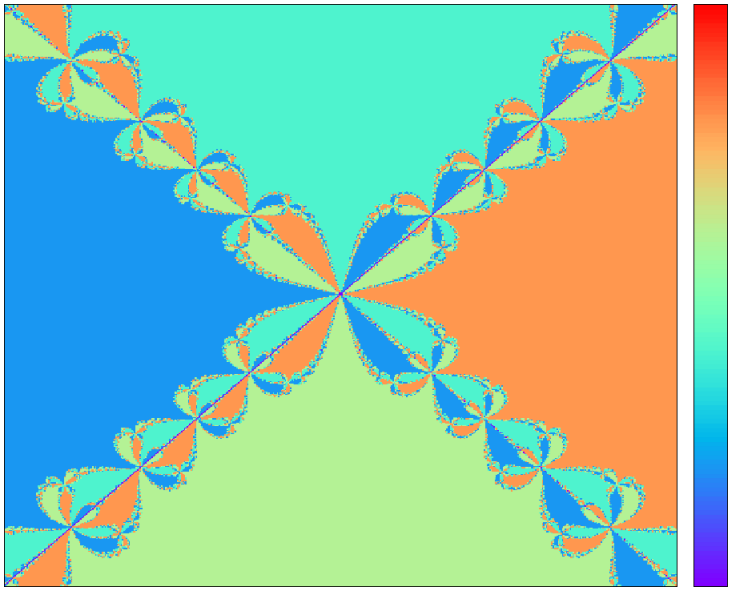

Figura 1: Bacias de convergência do polinômio $x^4 - 1$. O plano visto varia de -2 a 2 no eixo x e de $-2i$ a $2i$ no eixo y

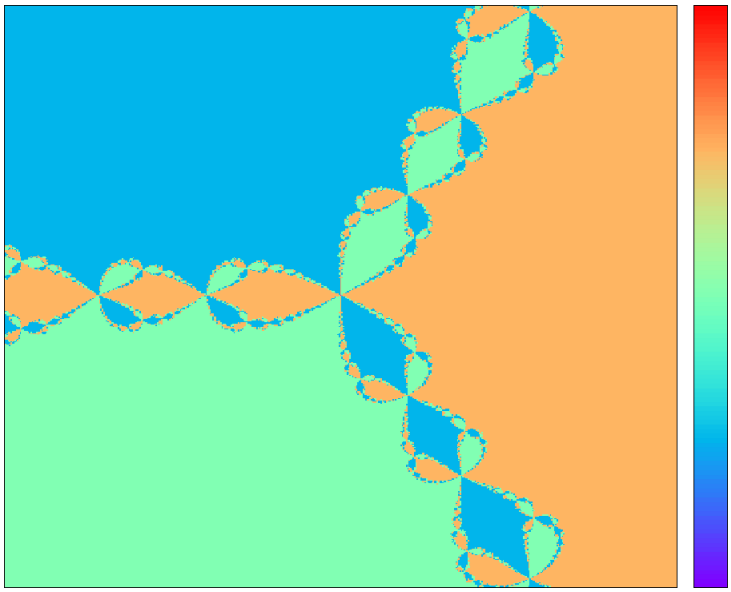

Figura 2: Bacias de convergência do polinômio $x^3 - 1$. O plano visto varia de -2 a 2 no eixo x e de $-2i$ a $2i$ no eixo y

2.1 Como gerar imagens das bacias de convergência

O primeiro passo que você deve tomar para gerar as imagens é escolher um subconjunto do plano complexo. Agora, associe a cada um dos n^2 (n é um parâmetro de sua função) pixels da imagem a um ponto do plano, rode o método de newton usando como início esse

ponto e descubra a raíz encontrada. Cada raíz do polinomio deve estar associada a uma cor, que será pintada em todos os pontos iniciais que convergirem para essa raíz.

As imagens das bacias de convergência devem ser geradas pelo programa *gnuplot* usando o script disponibilizado, *plot_basins*. Esse script lê as linhas de um arquivo chamado *output.txt* que devem conter uma tripla $x\ y\ r$ que representam respectivamente: a coordenada x do pixel na **imagem gerada**; a coordenada y do pixel na **imagem gerada**; e um inteiro r que representa a raíz encontrada quando rodamos o Método de Newton usando o ponto associado a (x, y) como partida do algoritmo.

A escolha de criar um mapeamento entre as raízes de sua função para um número inteiro r se dá para facilitar a escolha de uma cor para cada raíz. O script *plot_basins* possui uma paleta de cores que são usadas de acordo com o *range* definido no script em:

$$\text{set cbrange } [a : b]$$

Na paleta usada na figura 1, por exemplo, uma raíz associada ao número a teria cor roxa enquanto que uma raíz associada ao número b teria cor vermelha. Note que você também precisa associar os pontos que não convergem a nenhuma raíz a uma cor. Você pode modificar o script que faz o plot ao seu gosto (sem mudar o formato dos arquivos de entrada e saída) para criar imagens.

2.2 Funções a serem implementadas

Você deve implementar suas funções em um arquivo chamado *newton_basins.m* que deve conter pelo menos as seguintes funções:

- newton_basins (f, n): acha as bacias de convergência da função f e gera um arquivo *output.txt* que contém os dados para geração da imagem das bacias pelo script *plot_basins*. Os dados gerados preenchem uma imagem com $n \times n$ pixels.

- newton (f, f', x_0): aplica o método de newton para achar uma raíz da função f (com primeira derivada f'), partindo do ponto x_0.

As funções f e f' devem ser polinômios e devem ser representados por um vetor com seus respectivos coeficientes, i.e

$$\begin{aligned} \text{se } f \text{ representa} \quad & p(x) = c_n x^n + c_{n-1} x^{n-1} + ... + c_0 \\ \text{então} \quad & f = [c_n, c_{n-1}, ..., c_0] \end{aligned}$$

2.3 O que deve ser entregue

Você deve entregar ao paca seu programa em Octave que implementa o Método de Newton como foi detalhado anteriormente e também um relatório sobre o problema. O relatório deve explicar como você aplicou o Método de Newton e também deve conter exemplos interessantes com imagens geradas pelo seu programa (de preferência diferentes das que foram apresentadas neste enunciado como exemplo). Seu relatório deve estar no mesmo arquivo que as soluções da parte 1 e relatório da parte 3.

3 Parte 3 - Encontrando Todas as Raízes de Funções

Escreva um programa em Octave para achar todas as raízes de uma função $f \in C^2[a,b]$. Seu programa dividir $[a,b]$ em *ninter* sub-intervalos equidistantes e descobrir em quais deles a função f troca de sinal.

Para cada sub-intervalo $[a_i, b_i]$ sobre o qual $f(x)$ troca de sinal, seu programa deve então achar as raízes da seguinte maneira. Use o método de Newton ou secante para achar a raíz do intervalo, monitorando o decrescimento de $|f(x_k)|$. Se uma iteração é alcançada tal que não há decrescimento suficiente de $|f(x_k)|$, i.e $|f(x_k)| > 0.5f(x_{k-1})$, então volte ao intervalo $[a_i, b_i]$, aplique bisecção três vezes e recomece o método de Newton ou secante.

A i-ésima raíz x_i é considerada achada quando:

$$\begin{aligned} |x_k - x_{k-1}| &< tol(1 + |x_k|) \\ |f(x_k)| &< tol \end{aligned}$$

Observação: Você deve utilizar funções anônimas para representar f.

Você deve enviar ao paca o código do seu programa Octave e também um relatório, em pdf junto a parte 1 e 2 do EP.

Verifique seu programa achando as raízes das funções:

- $f(x) = 2\cosh(x/4) - x$
- $f(x) = \begin{cases} \frac{\sin(x)}{x}, \text{ if } x \neq 0 \\ 1, \text{ if } x = 0 \end{cases}$

Este problema é baseado no exercício 13 do capítulo 3 do livro "A First Course in Numerical Methods"presente na bibliografia do curso.

4 Entrega

Segue abaixo as regras de entrega deste EP:

- Este EP poderá ser feito em dupla.
- A entrega será aceita até as 23:55 do dia **18 de setembro de 2016** (domingo).
- Apenas um aluno da dupla deve submeter o EP no paca.
- Você deve submeter ao paca:

  - Dois programas em Octave, referentes as partes 2 e 3 do EP.

  - Um arquivo pdf que contenha as soluções da parte 1 e também relatórios da parte 2 e 3.

- A cópia impressa da part escrita do EP deverá ser entregue em sala de aula, no dia 20 de setembro, e não pode diferir da versão entregue ao paca.