Métodos Numéricos
Exercícios

Solução de equações

1. Implemente o algoritmo abaixo e responda as perguntas a seguir:

```
double aux = 1.0;
while 1 + aux > 1 {
    print aux;
    aux = aux / 2;
}
```

   (a) O algoritmo termina? Por que?

   (b) O que significa o valor impresso na última linha do algoritmo (se um dia ele terminar?)

2. Você tem um polinômio dado por $p(x) = 6 * x^5 + 18x^3 - 34x^2 - 493x + 1431$. Escreva um programa que calcule as cotas de Cauchy e Lagrange para as raizes de $p(x)$. Se você estiver em um dia mais corajoso, pode calcular a cota de Fujiwara também, que é mais complicada mas costuma ser mais precisa do que as outras.

3. Crie um polinômio que tem 2, 3 e 4 como raizes. Depois disso use o método da bissecção para encontrar uma raiz no intervalo [1, 5]. Ele encontra raizes? Uma delas? Todas? Quais? Sempre?

4. Agora crie um polinômio que tem 2, 3, 4 e 5 como raizes e procure uma raiz no intervalo [1, 6]. O que aconteceu? Você pode adaptar seu algoritmo para se ajustar a esta situação?

5. Implemente uma função baseada no método de Newton para encontrar a raiz quadrada de um número p.

6. Implemente um método para encontrar a raiz cúbica de um número p.

7. Seu sistema de plotagem de funções tem um *bug* e só consegue plotar funções para x variando entre -1 e 1. Com esta restrição, você quer achar as quatro soluções reais de $f(x) = 0$ para
$$f(x) = 8x^4 - 238x^3 + 1047x^2 - 953x + 154$$
Se você não pode olhar para a função fora do intervalo $[-1, 1]$, descubra como plotar $f(1/x)$ pode ajudar a resolver o problema.

8. Se você tem um polinômio dado por $p(x) = x^5 + 18x^3 + 34x^2 - 493x + 1431$, analise o algoritmo abaixo e determine o que significam os valores que ele irá imprimir. Implemente-o em sua linguagem preferida e teste-o para vários valores de `x`, se for necessário.

```
void metodo( double x ) :
  double a[] = {1, 0, 18, 34, -493, 1431};
  double p = 0;
  para i = 0 a tamanho(a)-1 faça
    p = x * p + a[i];
  imprima x, p;
```

9. Use o pedaço de código do problema 8 para implementar o método da secante para encontrar as raizes reais do polinômio. Talvez você deva começar perguntando quantas raizes reais podem existir.

10. O algoritmo do problema 8 foi mudado misteriosamente. Analise o novo algoritmo e determine o que significam os valores que ele irá imprimir. Implemente-o em sua linguagem preferida e teste-o para vários valores de `x`, se for necessário.

```
void metodo( double x ) :
  double a[] = {1, 0, 18, 34, -493, 1431};
  double p = 0;
  double q = 0;
  para i = 0 a tamanho(a)-1 faça
      q = x * q + p;
      p = x * p + a[i];
  imprima x, p, q;
```

11. Se sua linguagem preferida tem uma classe para números complexos, use o pedaço de código do problema 10 para implementar o método de Newton usando números complexos para encontrar todas as raizes do polinômio.

12. Se sua linguagem preferida tem uma classe para números complexos, comece criando um polinômio $p(x)$ com raizes que você já conhece (você pode escolher as raizes que quiser e quantas quiser!). Depois disso use o pedaço de código do problema 10 para implementar o método de Newton usando números complexos para encontrar todas as raizes do polinômio, agora fazendo um *plot twist*:

    (a) Escolha uma região do plano complexo e largue pontinhos iniciais do método de Newton;

    (b) Cada pontinho inicial deve levar até uma das raizes;

    (c) Como você já sabe quais são as raizes, pode identificar qual delas foi encontrada;

    (d) De acordo com a raiz encontrada, você pode pintar o pontinho inicial com a "cor" da raiz;

    (e) Plote tudo e você pode terminar com imagens como estas: Paul Bourke, Mitch Richling e John Whitehouse