# Vetores e Polinômios

Projeto 1 - Algoritmos e Técnicas de Programação 2

26 de setembro de 2023

## 1 Introdução

Polinômios possuem aplicações das mais diversas. Enquanto recurso de modelagem matemática, são funções simples e de fácil manipulação algébrica. Na análise de dados, podem ser usados em algoritmos de interpolação para uma primeira análise da relação funcional entre duas variáveis. Em redes sociais, o algoritmo de agrupamento espectral passa por calcular raízes de um polinômio para encontrar estruturas de grupo nessas redes, um problema conhecido como detecção de comunidades.

Um polinômio de grau $n \geq 0$ é uma função $p_n(x)$ da forma

$$p_n(x) = \sum_{i=0}^{n} a_i x^i, \tag{1}$$

onde $\boldsymbol{a} = (a_0, a_1, ..., a_n)$ é o **vetor de coeficientes** do polinômio.

Note então que a função $p_n(x)$ é especificada pelo vetor de coeficientes. Em outras palavras, para cada vetor $\boldsymbol{a}$ existe um polinômio e para cada polinômio existe um vetor $\boldsymbol{a}$. Isso permite então especificar um *struct* similar ao de um vetor, mas que representa um polinômio.

A proposta deste projeto é que você implemente funções para polinômios por meio de um *struct* o qual compreende os coeficientes e o grau do polinômio. Em analogia com o *struct* de vetor, $\boldsymbol{a}$ consiste dos dados do vetor, enquanto $n + 1$ é o número de elementos do mesmo.

Ao final desse projeto, você possuirá uma pequena biblioteca capaz de fazer algumas manipulações algébricas em polinômios. Essas operações são a um **nível simbólico**, que é a forma como humanos fazem quando manipulam estas funções, em oposição a operações a nível numérico, que são implementadas nativamente pela linguagem. Com isso, como você verá a seguir, pode-se expressar operações mais complexas como derivadas e integrais.

Você pode considerar a manipulação numérica tradicional como mais próxima do que faz uma calculadora, enquanto a proposta do projeto aproxima suas manipulações de ferramentas mais elaboradas como Wolfram Alpha.

A seguir serão descritos blocos de tarefas para você implementar. Cada um destes blocos será avaliado conforme os critérios descritos no plano de ensino, os quais são também utilizados nas atividades semanais que fizemos. **As questões de compilação e comentário são avaliadas para o código todo, constituindo a base da nota e valem 3 pontos na sua nota final.** Os demais critérios serão avaliados para cada bloco e o restante da nota será então a média das notas relativas a cada bloco.

## 2 Funções básicas do *struct* de polinômio (2 pontos)

Seu *struct* deve fazer as operações básicas de gerenciamento de memória como alocação de memória para os coeficientes e liberação desta para os mesmos. Além disso, implemente também uma função que leia os coeficientes de um arquivo e registre-os no *struct* que você implementou. Faça também uma função que imprima na tela os coeficientes do polinômio.

É necessário também ser capaz de calcular o valor de $p_n(x)$ para um dado valor de $x$. Assim, implemente uma função que receba como argumentos seu *struct* e um número $z$ e retorne o valor de $p_n(z)$.

## 3 Soma, subtração e produto (2 pontos)

Supondo $n \leq m$, considere dois polinômios

$$p_n(x) = \sum_{i=0}^{n} a_i x^i \tag{2}$$

e

$$q_m(x) = \sum_{j=0}^{m} b_j x^j. \tag{3}$$

As operações de soma, subtração e produto entre os mesmos produzem novos polinômios. Estas são dadas por

$$p_n(x) + q_n(x) = \sum_{i=0}^{n}(a_i + b_i)x^i + \sum_{j=n+1}^{m} b_j x^j \quad \text{(soma)} \tag{4}$$

$$p_n(x) - q_n(x) = \sum_{i=0}^{n}(a_i - b_i)x^i - \sum_{j=n+1}^{m} b_j x^j \quad \text{(subtração)} \tag{5}$$

e veja que o segundo somatório em cada uma destas expressões só é realizado nos casos em que $n < m$. Note também que, diferente, da soma de vetores, que demandava vetores de mesmo tamanho, você pode somar e subtrair polinômos de graus diferentes.

A operação de produto também pode ser realizada entre polinômos de graus distintos e é dada por

$$p_n(x)q_m(x) = \left(\sum_{i=0}^{n} a_i x^i\right)\left(\sum_{j=0}^{m} b_j x^j\right). \tag{6}$$

Atente-se ao grau do polinômio resultante.

## 4 Derivada e integral (2 pontos)

As operações de derivada e integral sobre polinômios também resultam em um polinômio. Aqui, você deve implementar três funções: derivada, integral definida e integral indefinida, as quais serão especificadas a seguir.

O polinômio derivada de $p_n(x)$ é dado por.

$$p'_n(x) = \sum_{i=1}^{n} i a_i x^{i-1}. \tag{7}$$

Então, sua função derivada deve receber como argumento um polinômio e retornar o polinômio derivada.

A segunda função a ser implementada é a integral indefinida de $p_n(x)$, que deve receber como argumento um polinômio e uma constante de integração $C$ e retornar outro polinômio. A integral indefinida de um polinômio é dada por

$$P_{n+1}(x) = \int p_n(x)dx = C + \sum_{i=0}^{n} \frac{a_i}{i+1} x^{i+1}. \tag{8}$$

Em seguida, implemente a integral definida de um polinômio, que recebe como argumento um polinômio e os limites de integração $x_0$ e $x_1$ e retorna um número, que é o resultado da integral definida:

$$\int_{x_0}^{x_1} p_n(x)dx = P_{n+1}(x_1) - P_{n+1}(x_0). \tag{9}$$

Portanto, o cálculo da integral definida (IDEF) passa pelo cálculo da integral indefinida (INDEF). Assim, na implementação de IDEF há uma chamada para INDEF. Nessa chamada de INDEF, a constante de integração pode ser escolhida para um valor arbitrário, pois não afetará o valor da integral definida. Você pode verificar isso vendo que quando faz-se $P_{n+1}(x_1) - P_{n+1}(x_0)$ as constantes de integração de cada termo se anulam.

# 5 O método de Newton-Raphson (1 ponto)

Para estimar uma raíz de um polinômio, pode-se utilizar o método de Newton-Raphson. Este é um procedimento iterativo no qual faz-se, a cada iteração $t$,

$$x_{t+1} = x_t - \frac{p_n(x_t)}{p'_n(x_t)} \tag{10}$$

até que atinja-se um limite de iterações $t_{max}$ ou uma tolerância

$$|x_{t+1} - x_t| < \epsilon, \tag{11}$$

com $\epsilon > 0$. O que ocorrer primeiro. Certifique-se sempre que uma divisão por zero não será realizada e informe o usuário caso isso venha a ocorrer.

Assim, sua função deve receber um polinômio, um valor inicial $x_0$, o número máximo de iterações $t_{max}$ e uma tolerância $\epsilon > 0$. Como retorno, a função fornece o valor final estimado para a raíz.

**Atenção:** polinômios podem possuir raízes complexas. Você **NÃO** precisa considerar esse caso. Assim, durante seus testes, certifique-se que o polinômio de entrada possui uma raíz real para ser encontrada (pode utilizar ferramentas como WolframAlpha para checar a corretude do seu método e existência de raízes reais).

Você, no entanto, pode tomar o seguinte desafio: ao invés de implementar os coeficientes como *floats*, você pode criar um *struct* de número complexo que armazena dois *floats*, correspondentes às partes real e imaginária do número complexo. Note que isso vai demandar de você escrever as operações básicas de soma, subtração, produto e divisão para este *struct* e reescrever todas as funções discutidas considerando essa álgebra.

**Não há qualquer recompensa ou exigência para a discussão feita no último parágrafo. Foi apenas um comentário sobre a questão das possíveis raízes complexas e como seria o tratamento desse tipo de problema.** É um trabalho realmente difícil, o qual nem eu tentei fazer. Recomendo fortemente que tentem uma aventura desse tipo apenas nas férias.

# 6 A interação com o usuário (1 ponto)

Essa parte é absolutamente fundamental. Sem nenhuma interação com o usuário, seu trabalho receberá nota 0. Isso se dá porque todo programa de computador é feito para ser usado por alguém que potencialmente não sabe o que está acontecendo no código. Dessa forma, é importante notificar os erros (ainda que não sejam plenamente inteligíveis para uma pessoa totalmente leiga) e assegurar uma boa interação com quem for utilizar seu programa.

Destaco ainda que não é necessário criar uma interação com o usuário para uma função não implementada. Caso você não consiga implementar algo que foi pedido acima, basta ignorar a interação correspondente.

O usuário do seu programa deve fornecer dois arquivos contendo coeficientes para dois polinômios. Nesses arquivos, em cada linha haverá um único coeficiente. Você pode implementar sua função de leitura considerando esta propriedade. Uma vez que os polinômios forem carregados, imprima na tela os coeficientes de cada um deles e também os de suas derivadas.

Em seguida, imprima na tela os coeficientes dos polinômios que são obtidos das operações de soma, subtração e produto entre os dois polinômios fornecidos como entrada.

Peça então para que o usuário forneça dois números, os quais serão utilizados como limites de integração no cálculo da integral definida desses polinômios. Você pode fazer essa operação para ambos os polinômios utilizando um único limite fornecido pelo usuário.

Por fim, peça para o usuário inserir uma condição inicial, um número de iterações e uma tolerância $\epsilon > 0$ e faça o método de Newton-Raphson para ambos os polinômios usando os dados fornecidos (novamente, pode usar os mesmos parâmetros para ambos).

Lembre-se de utilizar as funções de gerenciamento de memória que você escreveu.