INE5202 - Cálculo Numérico em Computadores

Relatório EP1

Alunos: Jéssica R. dos Santos e Marcos Machado

Professor: Priscila Cardoso Calegari

Florianópolis, 2024

Tarefa 1

Implementação do Método de Newton-Horner para encontrar raízes de polinômios

Objetivo: implementar, testar e verificar a eficácia do Método de Newton-Horner para encontrar as raízes do polinômio teste $p(x) = x^4 - 3x^3 + x^2 + x + 1$.

Implementação do Método de Newton-Horner:

O método foi implementado em Python com 3 funções principais: `eval_p()`, `horner()` e `newton_horner()`.

A função `horner(`x_0`, a)` calcula o valor do polinômio $p(x_0)$ e sua derivada $p'(x_0)$ no ponto x0. Ela recebe como entrada o valor de x0 a ser avaliado e os coeficientes do polinômio a.

A função `newton_horner(`x_0`, a)` utiliza o Método de Newton para encontrar as raízes do polinômio fazendo uso da função `horner()`. Ela itera até que a raiz seja encontrada com a precisão desejada (na nossa implementação, com uma tolerância de erro de 1×10^{-16}, ou seja, com precisão nas 16 primeiras casas decimais) ou até alcançar um número máximo de iterações (100 iterações, na nossa aplicação).

A função `eval_p(a, x)` é utilizada para avaliar uma função, dado uma lista `a` com seus coeficientes, em um dado valor x. Já que, no nosso código, o polinômio é representado por uma lista de coeficientes (a fim de realizar as operações de `horner()`), ao invés de uma função convencional, foi necessário usar esse método para determinar se o chute atual já atingia a precisão requerida.

Experimentação e Análise dos Resultados

O Método de Newton-Horner foi aplicado em p(x). O processo iterativo foi iniciado com um chute inicial para cada uma das raízes do polinômio (1.3 para a primeira raiz, e 2.3 para a segunda).

Os resultados obtidos foram analisados por meio gráfico gerado pela biblioteca `matplotlib`. O gráfico foi utilizado para visualizar as raízes encontradas em relação ao polinômio modificado após a aplicação do método.

A implementação do Método de Newton-Horner resultou na determinação precisa das raízes do polinômio p(x).

O gráfico gerado permite uma análise visual do comportamento do polinômio modificado após a aplicação do método. Observou-se claramente os pontos em que o polinômio cortou o eixo x, indicando a localização das raízes.

A precisão das raízes encontradas foi verificada comparando-as com valores teóricos esperados. As raízes determinadas pelo método estavam em concordância com esses valores, confirmando a eficácia do Método de Newton-Horner na resolução do problema.

Problemas/Dificuldades Encontradas

1. Implementação a partir de pseudocódigos e teoremas;
2. A necessidade de adicionar tolerância/critério de parada;
3. Posicionar e mostrar o valor do chute inicial no local equivalente do gráfico;

Tarefa 2

Implementação do Método de Muller para encontrar raízes de polinômios

Objetivo: implementar, testar e verificar a eficácia do Método de Muller para encontrar as raízes do polinômio teste $p(x) = x^4 - 3x^3 + x^2 + x + 1$.

Implementação do Método de Muller:

De forma similar a tarefa 1, a implementação do método se deu através de um programa em Python, usando os métodos `mullers_method()` e `sqrt()`.

O método `mullers_method()`, parte principal da implementação, aplica o algoritmo fornecido no enunciado do EP1. Dados 3 chutes iniciais para a raiz, o próximo chute x3 é determinado como o zero da parábola que passa pelos pontos $((x_0,p(x_0)), (x_1,p(x_1)), (x_2,p(x_2)))$. Novos chutes são calculados até que $f(x_3)$ alcance a precisão desejada (na nossa implementação, com uma tolerância de erro de 1×10^{-14}, ou seja, com precisão nas 14 primeiras casas decimais) ou até que o número máximo de iterações seja atingida (100 iterações, na nossa aplicação).

O método `sqrt()` é utilizado dentro de `mullers_method()`, tendo o papel de avaliar qual será a função utilizada para calcular a raiz de $(b^2 - 4ac)$: aquela para raízes reais, da biblioteca math (`math.sqrt()`), ou aquela dedicada para raízes complexas não-reais, da biblioteca cmath (`cmath.sqrt()`), visto que cada função só suporta resultados dentro desses respectivos domínios. Para preservar a legibilidade do método principal, foi preferido dedicar uma função própria para essa tomada de decisão.

Experimentação e Análise dos Resultados

Para encontrar ambas as raízes reais, foram realizados chutes iniciais baseados na análise visual do gráfico.

Para encontrar a primeira raiz, foram usados os chutes {1.3, 1.4, 1.5}, e para a segunda raiz, os números {2.2, 2.3, 2.4}. Já para encontrar uma das raízes imaginárias, foram usados os chutes concedidos no enunciado do EP {-0.5, 0, 0.5}.

Os resultados obtidos foram comparados com resultados teóricos obtidos por ferramentas on-line, além de também comparados com os resultados obtidos por meio da tarefa 1. As saídas do programa foram compatíveis com o esperado, mostrando a eficácia do Método de Muller para encontrá-las.

Problemas/Dificuldades Encontradas

1. Interpretação do algoritmo fornecido no EP (uma vez entendido como o pseudocódigo operava, foi relativamente fácil implementar em Python).
2. Encontrar uma forma de conciliar operações de números reais e complexos (visto que os métodos `sqrt()` das bibliotecas math e cmath retornam exclusivamente resultados reais e complexos não-reais, respectivamente).

3. Truncar/arredondar números complexos não-reais a fim de exibi-los no terminal (estes números não cooperam com certos métodos convencionais de formatação de números reais no Python, como, por exemplo, usando o operador %).
4. Decidir o nível de precisão/tolerância de erro (houve tentativas de tentar obter o mesmo nível de precisão na tarefa 1, mas ocorreu erros de execução do programa. Aparentemente, ao calcular q_0 com uma divisão de denominador (x_0-x_2), a diferença entre os chutes era tão pequena que era considerada zero pelo programa, que finalizava com ZeroDivisionError).

Gráficos

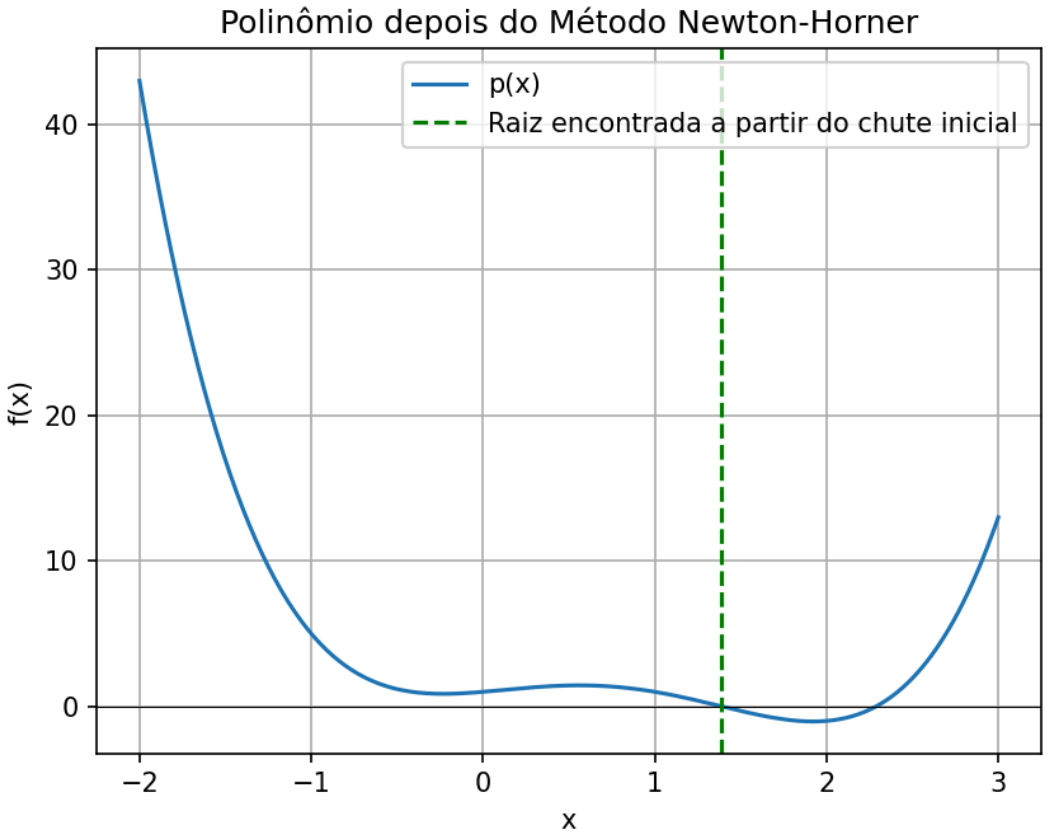

Figura 1. Análise visual de p(x) teste modificado após a aplicação do Método Newton-Horner. Chute inicial x0 = 0.

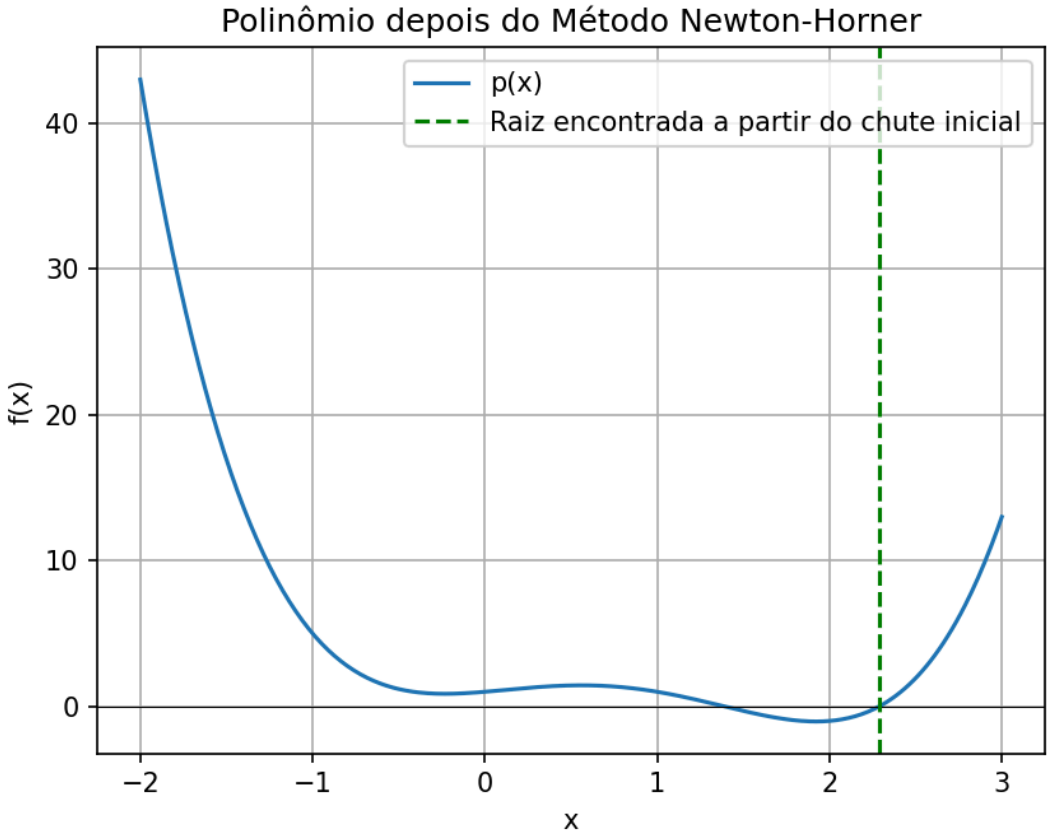

Figura 2. Análise visual de p(x) teste modificado após a aplicação do Método Newton-Horner. Chute inicial x0 = 3.

Tabelas

- Discussão e Análise dos Resultados: Como citado anteriormente no relatório, encontrar as raízes exatas para o polinômio p(x) foi um dos nossos principais desafios. As raízes x1 = 1.39 e x2 = 2.29 (arredondamento) são encontradas dependendo do chute inicial (gráficos acima). Abaixo, estão alguns exemplos de resultados de iterações feitas pelo Método de Newton-Horner.

Valor de x	Valor f(x)
0.988	1.11
1.017	1.20
1.202	0.41
1.335	0.59
1.380	0.15

1.964	-1.07
2.163	-0.29
2.222	-0.11
2.274	-0.06

Após discussões, em dupla, concluímos que quanto mais próximo da raiz verdadeira for o chute inicial, mais fácil seria para o algoritmo calcular valores próximos a mesma.
Parece uma conclusão óbvia, o maior problema seria: e se não soubéssemos as raízes verdadeiras desde o início? Quanto a escolha de um chute inicial apropriado influência no correto funcionamento do método e número de iterações necessárias?