CCI22 2018
Lab2 Zeros de Funções

27 de fevereiro de 2018

1 Tarefas

1.1 Implementação

As seguintes funções já implementadas são fornecidas:

1. `[r,n] = Bisseccao(f,a,b,epsilon,maxIteracoes)`: determina uma aproximação para um zero contido no intervalo $[a, b]$ da função $f(x)$ utilizando o Método da Bissecção. O retorno da função é a aproximação r e o número de iterações n executadas. Considera-se para critério de parada a ocorrência de uma das duas situações: $|f(x_i)| < \varepsilon$ ou $i >$ `maxIteracoes`, em que x_i é a aproximação para a raiz na i-ésima iteração.

2. `[r,n] = PontoFixo(f,g,x0,epsilon,maxIteracoes)`: determina um zero da função $f(x)$ a partir do chute inicial x_0 utilizando o Método do Ponto Fixo com função de iteração $g(x)$. O retorno da função é a aproximação r e o número de iterações n executadas. Considera-se para critério de parada a ocorrência de uma das duas situações: $|f(x_i)| < \varepsilon$ ou $i >$ `maxIteracoes`, em que x_i é a aproximação para a raiz na i-ésima iteração.

Implementar as seguintes funções em MATLAB (cada uma em um arquivo .m separado):

1. [2.5pt] `[r,n] = PosicaoFalsa(f,a,b,epsilon,maxIteracoes)`: determina uma aproximação para um zero contido no intervalo $[a, b]$ da função $f(x)$ utilizando o Método da Posição Falsa. O retorno da função é a aproximação r e o número de iterações n executadas. Considera-se para critério de parada a ocorrência de uma das duas situações: $|f(x_i)| < \varepsilon$ ou $i >$ `maxIteracoes`, em que x_i é a aproximação para a raiz na i-ésima iteração.

2. [2.5pt] `[r,n] = NewtonRaphson(f,df,x0,epsilon,maxIteracoes)`: determina um zero da função $f(x)$ a partir do chute inicial x_0 utilizando o Método de Newton-Raphson com função derivada $f'(x) = df(x)$. O retorno da função é a aproximação r e o número de iterações n executadas. Considera-se para critério de parada a ocorrência de uma das duas situações: $|f(x_i)| < \varepsilon$ ou $i >$ `maxIteracoes`, em que x_i é a aproximação para a raiz na i-ésima iteração.

3. [2.5pt] `[r,n] = Secante(f,x0,x1,epsilon,maxIteracoes)`: determina um zero da função $f(x)$ a partir dos chutes iniciais x_0 e x_1 utilizando o Método da Secante. O retorno da função é a aproximação r e o número de iterações n executadas. Considera-se para critério de parada a ocorrência de uma das duas situações: $|f(x_i)| < \varepsilon$ ou $i >$ `maxIteracoes`, em que x_i é a aproximação para a raiz na i-ésima iteração.

1.2 Análise

Analise os comportamentos dos métodos (fornecidos e implementados) para os seguintes casos de teste:

1. Raiz da função $f(x) = x^3 - x^2 + 10x - 5$ no intervalo $[0, 1]$. Usar a função de iteração $g(x) = (5 - x^3 + x^2)/10$ no caso do Método do Ponto Fixo.
2. Raiz da função $f(x) = e^{-x^2} - \cos(x)$ no intervalo $[1, 2]$. Usar a função de iteração $g(x) = \cos(x) - e^{-x^2} + x$ no caso do Método do Ponto Fixo.

Em todos os casos, utilize $\varepsilon = 10^{-4}$ e `maxIteracoes` $= 1000$. Além disso, configure os métodos da seguinte forma:

- Bissecção e Posição Falsa: utilize a e b dados no caso de teste.
- Ponto Fixo: utilize $x_0 = (a + b)/2$.
- Newton-Raphson: utilize $x_0 = (a + b)/2$.
- Secante: utilize $x_0 = a$ e $x_1 = b$.
- função `fzero` do proprio matlab. O script `rootprob.m` configura a função para imprimir informações adicionais. Note que a tolerância desta função é definida no eixo x, não em f(x), portanto a comparação com as outras não é exata.

Construa uma tabela para cada caso de teste conforme o modelo mostrado na Tabela 1 e discuta os resultados obtidos:

1. [1.5pt] Considere quais métodos apresentam resultados melhores ou piores e discuta porque, considerando as vantagens e desvantagens discutidas em aula sobre cada método. Considere o número de iterações como uma aproximação ao tempo de execução. Não precisa discutir a função matlab.
2. [1pt] Se um (alguns) determinado(s) método(s) apresenta(m) resultados melhores, porque existem os outros? Há limitações do melhor método?

	Bissecção	Posição Falsa	Ponto Fixo	Newton-Raphson	Secante	matlab
n r $f(r)$						

Tabela 1: Modelo de tabela para comparação entre os métodos.

2 Instruções

- A primeira etapa do processo de correção consistirá em submeter as funções implementadas a vários casos de teste de forma automatizada. Assim, os cabeçalhos das funções devem ser seguidos **rigorosamente**. Arquivos .m com os nomes destas funções e os cabeçalhos já implementados foram fornecidos juntamente com este roteiro. Dê preferência a implementar seu laboratório a partir destes arquivos .m fornecidos para evitar erros.

- **Não** é permitido o uso de funções ou comandos prontos do MATLAB que realizem toda a funcionalidade atribuída a uma certa função: `r = fzero(f,x0)`, `r = roots(p)` etc. Entretanto, o uso destas funções para verificação das implementações realizadas é encorajado. Em caso de dúvida quanto à permissão de uso de alguma função ou comando, recomenda-se consultar o professor.

- Não é necessário se preocupar com verificação dos dados de entrada: a implementação dos métodos pode assumir que há uma raiz no intervalo $[a, b]$ passado como argumento, que $g(x)$ seja uma função de iteração para $f(x)$, que $df(x)$ seja a função derivada de $f(x)$ etc.

- Os arquivos .m implementados devem ser entregues juntamente com um relatório.

- No relatório, não é necessário demonstrar que as funções implementadas funcionam corretamente (para isto há a correção automática). Basta incluir resultados e conclusões relativa a **Análise**.

- Perde-se ponto com gráficos ilegíveis, ou sem algum tipo de legenda que o explique. Só um bando de pontinhos espalhados sem nome não é um gráfico, certo?

- Perde-se ponto se os números não estiverem apresentados em uma representação coerente com dígitos significativos suficientes para ilustrar o resultado. Por exemplo, se não usar notação científica, números como 0.6042e-4 e 1.4234e-4 poderiam ambos serem apresentados como 0.0001! Ficaram iguais, mas são diferentes! Cuidado ao preencher as tabelas.

3 Dicas:

- Para definir uma função no MATLAB, utilize o operador @, por exemplo: `f = @(x) (x3 - x2 + 10*x - 5)`. O tipo da variável `f` é ponteiro de função, que o matlab chama de *function handle*. Se precisar saber se uma variável é um funcion handle, existe a função `isa(f, 'function_handle')`

- A implementação do Método da Posição Falsa é muito semelhante à do Método da Bissecção, que é fornecido.
- Analogamente, as implementações dos métodos de Newton-Raphson e da Secante são parecidas com a do Método do Ponto Fixo, que é fornecido.

4 FAQ

4.1 Podemos colocar no lab outros critérios de parada além dos listados? Por exemplo, caso em que f(a)*f(b)$>$0.

Respondendo em partes

$>$**Podemos colocar no lab outros casos de parada além dos listados?** É possivel que haja exemplos que funcionem melhor com outros critérios de parada. O problema do nosso exercicio, eh que o gabarito do testador usa apenas f(x) $<$ epsilon. Portanto, se usarem outro critério, pode terminar com resultados diferentes do gabarito e portanto não passar nos testes. Se precisarem de outro critério por algum motivo, podem colocar e explicar, mas deveria passar nos testes apenas com f(x) $<$ epsilon. Pelo menos consegui resolver os testes sem precisar de mais criterios de parada.

$>$**Por exemplo, caso em que f(a)*f(b)$>$0.** Sobre esse exemplo: Tanto o Newton quanto o Secante não precisam terminar se f(x_i) e f(x_{i+1}) possuem sinais diferentes. Ate mostramos casos onde isso acontece e ainda o metodo converge.