CCI-22
Lab7 Integração Numérica

22 de maio de 2018

1 Implementação (AUTOTEST):

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

1.1 [1 pt]: `I = IntegracaoTrapezio(h, y)`

Usando a regra dos trapézios, retorna o valor da integração numérica de uma função dados o espaçamento h entre os pontos x_i onde a função é amostrada com espaçamento constante; e o vetor $\mathbf{y}$ onde $y_i = f(x_i)$.

1.2 [1 pt] `I = IntegracaoSimpson(h, y)`

Usando a regra composta de Simpson, retorna o valor da integração numérica de uma função dados o espaçamento h entre os pontos x_i onde a função é amostrada com espaçamento constante; e o vetor $\mathbf{y}$ onde $y_i = f(x_i)$. Note que para a regra composta de simpson, o número de elementos de $\mathbf{y}$ deve ser ímpar.

note que para 1.1 e 1.2 *não* é necessário saber os valores de x_i, nem os extremos do intervalo de integração.

1.3 [3 pt] `[I, x] = IntegracaoQuadraturaAdaptativa(f, a, b, epsilon)`

usando o Método da Quadratura Adaptativa com Regra de Simpson, calcula uma aproximacao para a integral:

$$I = \int_a^b f(x)dx$$

As entradas são o ponteiro de função `f` representando a função $f(x)$, os extremos do intervalo de integração $[a, b]$ e o erro total na integração `epsilon`. As saídas são o valor da aproximação da integral I e o vetor coluna $\mathbf{x} = [x_0, x_1, ..., x_n]^T$ de pontos que foram usados para calcular a integral.

- Espera-se que seja definida uma função auxiliar para efetuar a recursão.

- Dica de organização: O MATLAB permite definir várias funções no mesmo arquivo .m. Nesse caso, apenas a primeira função definida (topo do arquivo) é acessível externamente e deve ter o mesmo nome do arquivo. As outras funções tem escopo local ao arquivo e podem ser chamadas apenas por outras funções no mesmo arquivo.

- Sobre a recursão interna: quando chamamos recursivamente a quadratura em um intervalo $[a, b]$, o nível de recursão anterior já calculou $f(a)$, $f(b)$, e $f(c)$ onde $c = (a+b)/2$. Portanto, $f(a)$, $f(b)$, e $f(c)$ devem ser usados como argumentos para as chamadas recursivas. Assim, nunca será necessário avaliar f mais de uma vez no mesmo ponto. Como apenas esses 3 pontos são conhecidos antes da chamada da função, não é necessário nenhuma estrutura de dados adicional.

- Também espera-se que seja chamada a função `IntegraçãoSimpson` implementada no exercício 1.2 portanto:
  - implemente este exercício por último!
  - Não utilize `IntegraçãoTrapezio` pois pode resultar em valores diferentes dos testes.
  - A fórmula da condição de parada possui uma constante 2^p onde p depende do método de integração. Certifique-se que o valor p é o correto para a regra de Simpson.

2 Análise

2.1 [2 pt] Comparar Simpson x Trapézio

Calcule a Integral

$$I = \int_1^2 x e^{x^2} dx$$

usando a regra composta dos trapézios e a regra composta de simpson, para valores de $n \in \{2 : 2 : 100\}$, e plote um gráfico onde no eixo x, está o valor de n, e no eixo y, o erro absoluto do cálculo da integral. Plote duas curvas nos mesmos eixos, uma curva para cada método, permitindo compará-los.

Para calcular o erro absoluto, deve-se calcular o valor exato da integral analiticamente.

2.2 Pontos da Quadratura Adaptativa

Dada a função
$f(x) = 0.5 + 0.02x^2 + e^{-(x-1)^2} \sin(\pi x)$

2.2.1 Compare Quadratura, Simpson e Trapézio.

Calcule a integral de f(x) no intervalo [-5 5] usando o método da Quadratura Adaptativa com $\varepsilon = 10^{-4}$. Encontre quantos pontos foram utilizados (ou seja, o tamanho do vetor **x** retornado), e denote este número de pontos de n.

Utilizando o mesmo número de pontos n, mas igualmente espaçados, calcule a integral com a regra composta de simpson, e com a regra composta dos trapézios. Preencha a tabela abaixo:

	Quadratura **[1 pt]**	Simpson **[0.25 pt]**	Trapézio **[0.25 pt]**
n (número de pontos)		o mesmo da Quad.	o mesmo da Quad.
erro			
número de chamadas de f			
número de avaliações de f			

onde número de chamadas é fornecido diretamente pelo profile, mas número de avaliações depende do número de elementos da entrada. Se `x` é um vetor de `n` elementos, o código `f(x)` representa uma chamada com `n` avaliações de `f`.

- A quadratura deve ser implementada de forma que a recursão reaproveite valores de $f(x_i)$ já calculados anteriormente. Ou seja, nunca devemos reavaliar f no mesmo ponto durante a recursão.
- No entanto, ***não vale*** executar a quadratura até o fim previamente para descobrir os pontos $(x_i, f(x_i))$ necessários e pré-calculá-los, eles devem ser calculados conforme a recursão avança.
- Sempre que possível, devemos agrupar em um vetor os valores de x a serem avaliados de forma a utilizar a notação e a otimização vetorial do MATLAB. Ou seja, **não queremos apenas reduzir o número de avaliações, mas também o número de chamadas.** Por exemplo, digamos que desejamos calcular `f(2)` e `f(4)`. É mais rápido fazer `resultado = f([2 4])` do que `resultado = [f(2) f(4)]`
- Isso implica que a função deve ser implementada no MATLAB de forma a aceitar vetores como entrada. Mostrei isso no código demo disponibilizado na aula.
- Sem otimizar o número de chamadas e avaliações, apenas com o resultado da integral correto, a quadratura não recebe nota nesta questão, sem prejuízo à nota da questão 1.3.

Aviso aos Navegantes: As funções gabarito ***não*** estão otimizadas em relação ao número de chamadas e/ou avaliações.

2.2.2 [0.5 pt] Visualise os pontos escolhidos

- Plote os pontos x_i escolhidos pelo método da quadratura adaptativa sobre o gráfico da própria função $f(x)$.

  - Ao plotar os pontos x_i, utilize apenas o marcador, e.g. `'*'`, e não ligue os pontos (e.g, não use `'*-'`).

- Na mesma figura, além de plotar os pontos x_i com um marcador em cada um, plote uma linha vertical passando por cada ponto, para permitir visualizar a diferença entre eles mais facilmente.

2.3 [1 pt] Comente os resultados dos itens 2.2.1 e 2.2.2.

3 Instruções:

- Valem as mesmas regras anteriores sobre o testador automático. Já que o testador está com vocês, qualquer erro, mesmo trivial, que não passe pela correção automática leva a princípio a zero no exercício.

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

4 Dicas:

- `line([ x x ],[ low high ])` traça uma linha vertical, na posição x no eixo horizontal, e de low até high no eixo vertical. É uma função de plot, então para desenhar na mesma figura é necessário chamar `hold on`.

4.1 Profile

Para calcular o número de chamadas de uma função use a ferramenta profile. Esta ferramenta é normalmente utilizada para se otimizar código, identificando em quais funções se gasta mais tempo. Mas aqui estamos interessados apenas no número de chamadas.

```
profile on; % turns it on
my_script_which_calls_many_functions_to_do_amazing_things; % runs your stuff!
p = profile('info'); % p is a struct, has info about all function calls.
                     % it also profiles inline functions defined with @(x) (...)
profview(0,p); % opens graphical interface for profile info.
profile clear; % zeroes all statistics.
profile off;   % dont let it on, it makes everything slower
```

O código acima é um resumo de:
`http://matlab.izmiran.ru/help/techdoc/ref/profile.html`
E note:

1. Provavelmente foram chamadas centenas de funções. Mas queremos saber quantas vezes foi chamada a função f a ser integrada. Ora, como ela deve ser a função mais chamada de todas, ou pelo menos uma das mais chamadas, basta abrir a interface gráfica `profview` e clicar na coluna "Calls" e ordenar a tabela pelo número

de chamadas, e ela deve aparecer no topo da tabela. Fácil! Depois disso, basta clicar no nome da função, e ainda é discriminado em outra tabela, quais funções chamaram a função escolhida (parent function), e quantas vezes cada parent funcion chamou a função escolhida. Por exemplo, lista quantas vezes f foi chamada pela QuadraturaAdaptivaRecursiva especificamente.

2. Não esqueça de zerar as estatísticas antes de cada execução (`profile clear`). Se não zerar as estatísticas, continua somando os novos valores com os anteriores.

3. Também é possível usar uma variável global para contar chamadas, mas nesse caso a função f deve ser um m file separado, e não pode estar na forma inline `@(x)`. Mas pra que se preocupar se a ferramenta profile já faz isso?

4. Não deixe o profile ligado sem necessidade, o matlab se tornará mais lento. Depois de obter os resultados, use `profile off`.