MÉTODOS NUMÉRICOS PARA INTEGRAÇÃO

Evandro Pedro Alves de Mendonça[a], **Marcelino José de Lima Andrade**[a].

[a] *Núcleo de Tecnologia (NTI), Universidade Federal de Pernambuco (UFPE), Campus Acadêmico do Agreste (CAA), Rodovia BR-104, km 59, S/N, Nova Caruaru, CEP. 55.014-900, Caruaru-PE, Brasil, http://www.ufpe.br/caa*

Palavras Chave: Integração numérica, fórmulas, erro, truncamento, aproximação, funções.

Resumo: Este trabalho visa analisar os diversos métodos de integração numérica e discernir qual o melhor método a ser empregado em diversas situações. Serão levados em conta vários parâmetros para a escolha do melhor método, fatores como tempo computacional e erro associado a cada método. No trabalho também é mostrada a implementação de cada método de integração pedido.

1 INTRODUÇÃO

A integração numérica tem importância inegável nas ciências exatas. Quando possível, a integração é feita analiticamente. Porém, existem algumas situações em que calcular uma integral analiticamente torna-se inviável. Pode ser que o integrando seja muito complicado, ou que não haja integrando realmente expresso, desta forma é quase que impossível resolver a integral analiticamente. Também existem situações em que é possível integra-se analiticamente, pois o integrando está devidamente expresso, porém o tempo para a análise do integrando é muito alto, e é mais viável optar-se pela integração numérica. Ainda pode-se ter o caso em que o integrando não está definido em todos os pontos do intervalo, tornando o cálculo analítico inviável novamente.

Alguns métodos numéricos levam em consideração o fato de que, a integral representa a área abaixo da curva na maioria dos casos. Por isso, os métodos fazem aproximações do integrando para funções mais simples, ou polígonos cuja área é facilmente calculável. Desta forma é possível ter-se uma boa aproximação da integral.

Dos métodos usados, dois bastante conhecidos são o de Newton-Cotes e da quadratura de Gauss. No método de Newton, o intervalo de integração é igualmente espaçado, e a função integrada é aproximada por polinômios conhecidos e facilmente integráveis. O método de Gauss é mais aprimorado e não necessariamente utiliza pontos igualmente espaçados, mas sim, pontos sabiamente escolhidos previamente, para que o erro seja mínimo.

O cálculo do erro deve ser feito sempre e é primordial, para que se tenha a noção da acurácia dos valores obtidos. A partir do erro, pode-se escolher e discernir qual o método deve ser utilizado em cada caso, e ajuda na quantidade de intervalos que devem ser usados.

2 EXERCÍCIOS PROPOSTOS

Segue, abaixo, a solução dos exercícios propostos sobre o tema.

2.1 1ª questão.

A resolução da questão está na figura abaixo.

```
>> f = exp(-2*x)*cos(x)
f =
exp(-2*x)*cos(x)
>> n1 = NumMinIntTrapezio(0,pi/2,f,1e-4)
n1 =
    99
>> n2 = NumMinIntSimpson13(0,pi/2,f,1e-4)
n2 =
     8
>> I1 = IntegralTrapezio(0,pi/2,f,n1)
I1 =
   0.408683835249056
>> I2 = IntegralSimpson13(0,pi/2,f,n2)
I2 =
   0.408656838641268
>> I3 = quad(inline(f),0,pi/2)
I3 =
   0.408642683050944
```

Figura 1: resolução da 1ª questão.

Primeiramente, declaramos f, que é uma função da variável simbólica x. Em seguida, chamamos a função *NumMinIntTrapezio*, que calcula o número mínimo de subintervalos para o método do Trapézio considerando a tolerância 10^{-4}. O resultado indica que precisamos de no mínimo 99 subintervalos nesse método para garantir a tolerância exigida. Depois chamamos a função *NumMinIntSimpson13*, que faz a mesma coisa, só que considerando o método de 1/3 de Simpson. O resultado mostra que precisamos de apenas 8 subintervalos para garantir a tolerância. Para garantir que teremos quatro casas decimais iguais com os dois métodos, simplesmente calculamos a integral com ambos os métodos e também com uma terceira função que é nativa do MATLAB®. Como pode-se ver, nos três casos, as primeiras 4 casas decimais são idênticas. Os algoritmos desenvolvidos para essa questão estão nos Anexos 2, 4, 7 e 8.

2.2 2ª questão

2.2.1– 2.a)

Como não há uma maneira simples de encontrar o número de pontos que garante uma precisão determinada para a Quadratura de Gauss, implementamos um algoritmo que calcula a estimativa de erro. Então calculamos empiricamente a integral pedida e vemos se o erro atingiu a tolerância exigida.

Com o código do Anexo 6, calculamos a integral da função dada.

```
>> syms x
>> f = 1 / ((x + 1)*(x^2 - 3*x + 2)^(1 / 2))
f =
1/((x + 1)*(x^2 - 3*x + 2)^(1/2))
```

Figura 2: declaração da função.

```
>> [I,e] = IntegralQuadraturaGauss(2,5,f,2)
I =
   0.504609536755298
e =
     6.607607610093504e+15
>> [I,e] = IntegralQuadraturaGauss(2,5,f,3)
I =
   0.571297480364340
e =
     4.127716457875702e+23
>> [I,e] = IntegralQuadraturaGauss(2,5,f,4)
I =
   0.605919553449633
e =
     3.142289610742580e+31
>> [I,e] = IntegralQuadraturaGauss(2,5,f,5)
I =
   0.627201203698316
e =
     2.693275975305877e+39
>> [I,e] = IntegralQuadraturaGauss(2,5,f,6)
I =
   0.641691027560178
e =
     2.498950977519263e+47
```

Figura 3: cálculo da integral utilizando a Quadratura de Gauss com 2, 3, 4, 5 e 6 pontos.

Como foi observado, o erro calculado foi muito grande em todos os casos e o valor da integral converge muito lentamente para o valor real com o aumento de pontos.

2.2.1– 2.b)

Utilizando os métodos de Newton-Cotes apresentados na disciplina, apenas dois tornam o cálculo da integral viável. Isso acontece porque a função é assintótica no ponto 2, ou seja, a imagem da função nesse ponto tende a infinito. Com os métodos fechados que calculam a imagem do ponto inicial, o valor da integral também diverge. Portanto, precisamos de um método que utilize as imagens em outros pontos que não sejam o ponto inicial. Temos o método do Retângulo com o ponto final do intervalo e o método do Ponto Médio. Dentre esses, o que tem melhor exatidão é o método do ponto médio e este será o método utilizado nesta questão.

Para saber quantos subintervalos precisaremos para garantir a exatidão exigida, utilizamos o algoritmo do Anexo 9. Utilizando o algoritmo do Anexo 1, que implementa o método do ponto médio, calculamos a integral.

```
>> NumMinIntPontoMedio(2,5,f,1e-3)
ans =
     4903275
>> [I,e] = IntegralPontoMedio(2,5,f,4903275)
I =
   0.719480803186517
e =
     9.999997521267143e-04
```

Figura 4: com a primeira função, descobre-se o número de intervalos e, com a segunda função, calcula-se a integral e a estimativa de erro.

Vemos que com o método do ponto médio, garantimos um erro menor que 10-3, porém o número de subintervalos é quase 5 milhões, isto é, o método é muito ineficiente.

2.2.2– 2.c)

Utilizando funções do MATLAB® para integrar a função, temos:

```
>> I = quad(inline(f),2,5)
I =
   0.719649276295325
>> I = quadl(inline(f),2,5)
I =
   0.719638715659874
```

Figura 5: resultados utilizando as funções *quad* e *quadl*. Ambas realizam o cálculo com precisão de 10^{-6}.

Temos outras funções para calcular as integrais, que são as funções *trapz* e *dblquad*, porém a primeira é usada para pontos discretos e a segunda para integrais duplas. Portanto, não se aplicam ao escopo dessa questão.

Comparando os resultados com os dos itens anteriores, vemos que a função *IntegralPontoMedio* conseguiu um valor exato nas três primeiras casas decimais enquanto que a função *IntegralQuadraturaGauss* não conseguiu. Já era esperado, visto que o cálculo da estimativa de erro para esta função retornou um valor muito alto.

2.3 3ª questão

Primeiramente, calculamos os as integrais analiticamente.

$$(i) \int_0^{\pi} xsin(x)dx = \pi$$

$$(ii) \int_0^{2\pi} xsin(15x)dx = -\frac{2\pi}{15}$$

O gráfico das duas funções é mostrado na figura a seguir:

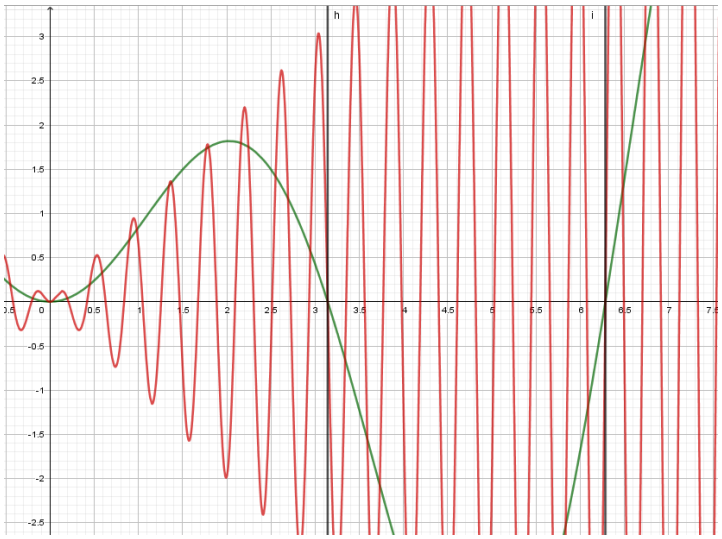

Figura 6: parte do gráfico das funções (i – em verde) e (ii – em vermelho). Também são mostradas na figura as retas x = π e x = 2 π, que são os limites superiores de integração. Gráfico gerado com o Geogebra®.

2.3.1– 3.a) a 3.d)

Para os itens 3.a) até 3.d), foi feito um algoritmo (Anexo 12) que resolve todas as integrais usando os algoritmos desenvolvidos para cada método (Anexos 2, 4, 5, 6), calcula os erros (Anexo 11) e dispõe tudo em uma tabela, que é mostrada na figura abaixo.

```
>> tic,questao3,toc
T =
  12×4 table
```

	Integral_i	Erro_Rel_i	Integral_ii	Erro_Rel_ii
Analítico	3.14159265358979	0	-0.418879020478639	0
Trapézio 6pt	3.06948875628924	2.29513833431467	-7.85844299979075e-15	99.9999999999981
Trapézio 12pt	3.12362867585603	0.571811170784204	-1.64493406684823	292.699081698725
Simpson 1/3 6pt	3.14294854875831	0.0431594836767754	6.53309891392712e-15	100.000000000002
Simpson 1/3 12pt	3.14167531571163	0.00263121705928851	-2.1932454224643	423.5987755983
Simpson 3/8 6pt	3.14474971329135	0.100492331427899	-2.19149241000623e-15	99.9999999999995
Simpson 3/8 12pt	3.1417801853323	0.00596932076129214	-2.46740110027234	489.048622548087
QG-Legendre 2pt	3.04077827835735	3.20902123059284	-9.98238949141007	2283.11994713975
QG-Legendre 3pt	3.14377445841638	0.0694490046026128	7.90816841048452	1987.93613999768
QG-Legendre 4pt	3.14156802109896	0.00078407653524361	-0.174884384305346	58.2494286523321
QG-Legendre 5pt	3.14159296178644	9.81020395873755e-06	2.84024989660614	778.059715991667
QG-Legendre 6pt	3.14159256697433	2.75705593785104e-06	3.46184825549851	926.455393144963

```
Elapsed time is 3.732023 seconds.
```

Figura 7: resultados em forma de tabela, contendo os valores das integrais e os erros relativos percentuais.

2.3.2– 3.e)

Os resultados da integral (i) foram bastante satisfatórios. Com um simples aumento no número de pontos usados, o erro diminuiu consideravelmente. Já para a integral (ii), não observa-se o mesmo comportamento. Não há como determinar se, com um aumento pequeno no número de pontos, o erro tende a diminuir. Para esse caso, talvez o erro só viesse a diminuir satisfatoriamente se o número de pontos fosse muito grande, o que fornece mais

exatidão ao cálculo. O motivo do comportamento dessas integrais é que, como a primeira integral tem uma taxa de variação pequena no domínio de integração, o erro também tende a ser pequeno; isso não acontece na segunda integral, que tem uma taxa de variação muito grande com relação à primeira, logo o erro também tende a variar bastante. Para calcular com mais exatidão essas integrais, pode-se usar mais pontos em cada método ou usar um método diferente. Por exemplo, o método adaptativo de Simpson discretiza com mais pontos as regiões do domínio que têm grande taxa de variação, o que confere maior precisão nas aproximações daquela região; nas demais regiões, onde a taxa de variação é pequena, a discretização também é menor, pois o erro tende a ser menor. Caso seja necessário, por exemplo, calcular a integral de uma função que modela uma onda de alta frequência, é mais vantajoso utilizar um método adaptativo.

2.4 4ª questão

Utilizando os algoritmos dos Anexos 3 e 13, encontramos os seguintes resultados.

```
>> questao4
ProfundidadeMedia =
   3.160000000000000
Area =
  63.200000000000003
VelocidadeMedia =
   0.264000000000000
Vazao =
  21.004000000000001
Elapsed time is 0.004773 seconds.
```

Figura 8: resolução da questão 4.

A seguir, discutiremos o procedimento feito para calcular cada uma das grandezas.

2.4.1 – 4.a)

Para calcular a profundidade média, a partir dos dados de profundidade, usamos o Teorema do Valor médio, que define que existe um valor H_m (profundidade média) que representa um retângulo de base Δy cuja área é igual à área sob a função H(y) (nesse caso, os dados de profundidade). Podemos então calcular a profundidade média como segue abaixo.

$$H_m \times \Delta y = \int_0^y H(y)dy \therefore H_m = \frac{1}{\Delta y}\int_0^y H(y)dy$$

$$onde\ \Delta y = y_{final} - y_{inicial} = 20 - 0 = 20$$

O valor da integral é obtido através do método do trapézio. A função *IntegralTrapezio* foi modificada para *IntTrapz*, em que ao invés de receber como parâmetros os extremos do domínio de integração, a função e o número de subintervalos, recebe apenas dois vetores de mesmo tamanho com dados discretos. Multiplicando o inverso de Δy pelo resultado da integral acima, obtivemos o valor da profundidade média.

2.4.2 – 4.b)

A área da seção foi simplesmente calculada a partir da integral:

$$A = \int_0^y H(y)dy$$

Usando a função *IntTrapz*, encontramos o valor da integral e, consequentemente, da área.

2.4.3– 4.c)

O valor da velocidade média foi encontrado da mesma forma que a profundidade média, através da seguinte expressão:

$$U_m = \frac{1}{\Delta y}\int_0^y U(y)dy$$

Novamente, fez-se uso da função *IntTrapz* para calcular o valor da integral e, assim, encontrar a velocidade média.

2.4.4– 4.d)

Para calcular a vazão do escoamento, a partir da seguinte expressão:

$$Q = \int_0^y H(y)U(y)dy$$

Encontramos o resultado calculando a integral pela função *IntTrapz*, porém passamos como parâmetro a multiplicação elemento a elemento dos vetores H e U.

2.5 5ª questão

As fórmulas de Quadratura de Gauss e suas características são:

- **Fórmula de Gauss-Legendre:**

Para utilizar a fórmula de Gauss-Legendre, a integral a ser calculada deve ter a função peso w(x) = 1, a = -1 e b = 1. Caso o intervalo de integração não coincida com o intervalo [-1,1], devemos fazer uma mudança de variável.

- **Fórmula de Gauss-Tchebyshev:**

Para utilizar as fórmulas de Gauss-Tchebyshev, a integral a ser calculada deve ter a função peso w(x) = $\frac{1}{\sqrt{1-x^2}}$, a = -1 e b = 1. Novamente, caso o intervalo de integração não coincida com o intervalo [-1,1], devemos fazer uma mudança de variável.

- **Fórmula de Gauss-Laguerre:**

Para utilizar a fórmula de Gauss-Laguerre, a integral a ser calculada deve ter a função peso w(x) = e^{-x}, a = 0 e b = ∞. Novamente, caso o intervalo de integração não coincida com o intervalo [0, ∞], devemos fazer uma mudança de variável, mas neste caso precisamos tomar mais cuidado, pois ao fazer uma mudança de variável mudamos também a função peso, que pode não ficar mais da forma requerida.

- **Fórmula de Gauss-Hermite:**

Para utilizar a fórmula de Gauss-Hermite, a integral a ser calculada deve ter a função peso w(x) = e^{-x^2}, a = -∞ e b = ∞. Neste caso, se o intervalo de integração não coincidir com o intervalo [-∞,∞], não podemos utilizar a fórmula de Gauss-Hermite.

2.5.1– 5.a) e 5.b)

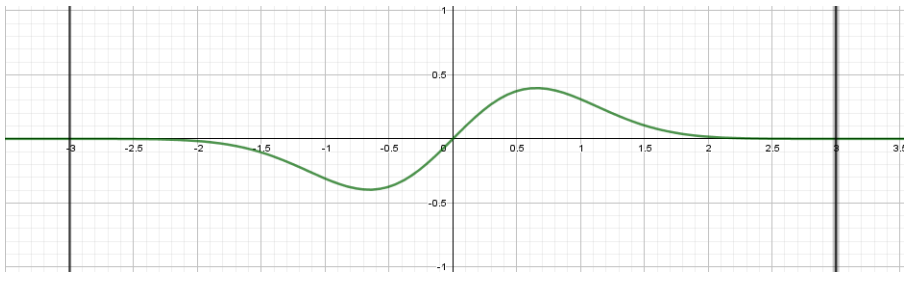

Figura 9: gráfico da função dada no enunciado da questão. O gráfico também contém as retas x=-3 e x=3, que representam os limites de integração usados nessa questão.

O gráfico da função é mostrado acima. Em integrais onde os intervalos tendem a infinito, pode-se plotar o gráfico da função e analisar em que pontos o integrando é próximo de 0. A partir daí, são escolhidos novos extremos de integração a partir dos pontos onde suas imagens são muito aproximadas de 0. No caso da função pedida na questão, o integrando é próximo de 0 nos pontos -3 e +3. Portanto, tomam-se esses pontos como novos limites de integração. Abaixo temos uma figura que mostra o cálculo da integral utilizando a Quadratura de Gauss-Legendre com 3 pontos. Em adição, para confirmar o resultado, foi calculada a integral pelos métodos do Trapézio, Simpson 1/3, Simpson 3/8, Quadratura de Gauss-Legendre de 6 pontos, com intervalo de -3 a +3, e também pelas funções do MATLAB® *quad*, *quadl* e *vpaintegral* (esta última do *Symbolic Math Toolbox*), obtêm-se os resultados da seguinte figura.

Também foi chamada a função *rsums*, do *Symbolic Math Toolbox*, que analisa a função dada, no intervalo fornecido, a partir de somas de Riemann. O gráfico permite selecionar o número de retângulos da discretização. Para essa questão, foram selecionados 60 retângulos, como pode ser visto na imagem posterior aos resultados.

```
>> f = exp(-x^2)*sin(x)
f =
exp(-x^2)*sin(x)
>> IntegralQuadraturaGauss(-3,3,f,3)
ans =
     0
```

Figura 10: resultado da integração pelo método da Quadratura de Gauss-Legendre com 3 pontos.

```
>> syms x
>> f = exp(-x^2)*sin(x)
f =
exp(-x^2)*sin(x)
>> IntegralTrapezio(-3,3,f,1000)
ans =
  -5.7585e-17
>> IntegralSimpson13(-3,3,f,1000)
ans =
   2.8948e-17
>> IntegralSimpson38(-3,3,f,900)
ans =
   4.4335e-17
>> IntegralQuadraturaGauss(-3,3,f,6)
ans =
  -2.9341e-17
>> quad(inline(f),-3,3)
ans =
   1.3878e-17
>> quadl(inline(f),-3,3)
ans =
  -2.2235e-19
>> vpaintegral(f,-3,3)
ans =
0.0
>> rsums(f,-3,3)
```

Figura 11: resultado da integração a partir de diferentes funções.

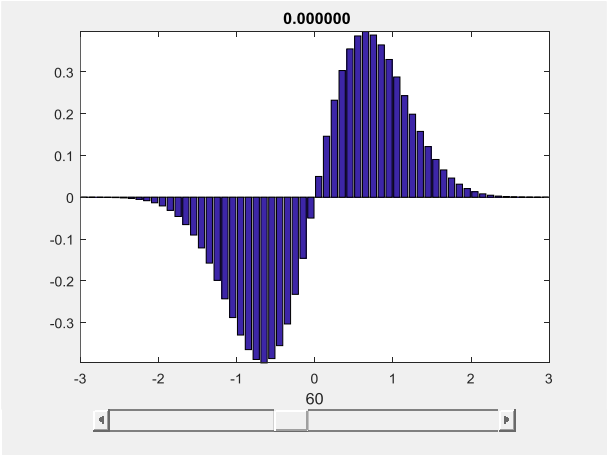

Figura 12: gráfico gerado pela função *rsums* da figura anterior, que mostra a função avaliada em somas de Riemann.

Como pode ser observado nas figuras acima, os resultados, em sua maioria, tendem a zero (os resultados para as funções *vpaintegral* e *IntegralQuadraturaGauss* com 3 pontos foram exatamente zero). Isso está de acordo com a teoria, pois como a função do enunciado é uma função ímpar, a integral definida (ou imprópria) com extremos de integração simétricos de uma função ímpar é igual a zero. Faz sentido também ao analisar o gráfico, pois todos os valores à direita da origem são idênticos aos valores à esquerda da origem, mas com sinal trocado. Ou seja, a soma infinita desses valores deve tender a zero.

Os algoritmos usados nessa questão estão nos Anexos 2, 4, 5, 6.

3 CONCLUSÃO

Conclui-se que os métodos para integração são bastante úteis no ramo da engenharia. Por diversas vezes não será possível realizar uma integração de maneira analítica. Por outras, será possível, porém, não será trivial. Por meio da utilização de métodos numéricos, é possível economizar tempo, e ter resultados com boa acurácia. Porém, é preciso escolher o método sabiamente, pois desta forma, tem-se o controle do erro obtido, e também, pode-se saber com a maior acurácia possível a dimensão do erro.

REFERÊNCIAS

Chapra, S. C., e Canale, R. P. *Métodos Numéricos para Engenharia*. 5ª edição. Porto Alegre: AMGH, 2011.

Gilat, A., e Subramaniam, V. *Métodos Numéricos para Engenheiros e Cientistas: uma introdução com aplicações usando o MATLAB*. Porto Alegre: Bookman, 2008.

ANEXO 1

```
function [ I,erro ] = IntegralPontoMedio(a,b,f,n)
%Calcula integral numericamente pelo método do Ponto Médio
%Parâmetros: [ I,erro ] = IntegralTrapezio(a,b,f,n)
%a: ponto inicial do domínio de integração
%b: ponto final do domínio de integração
%f: função do integrando
%n: número de intervalos
%I: valor da integral (saída)
%erro: estimativa de erro de truncamento (saída)

    %%%VALIDAÇÃO%%%
    if isa(f,'sym')==false
        f = sym(f); %transforma string em uma função simbólica
    end
    if isnumeric(a)==false||isnumeric(b)==false||isnumeric(n)==false
        disp('Erro. Parâmetros a, b e n devem ser numéricos.');
        I = 'erro'; erro = I;
        return;
    end
    if a>b % a deve ser menor que b
        disp('Erro. Parâmetro a deve ser menor que b.');
        I = 'erro'; erro = I;
        return;
    elseif a==b % a deve ser diferente de b
        disp('Erro. Parâmetros a e b devem ser diferentes.');
        I = 'erro'; erro = I;
        return;
    end
    if mod(fix(n),n)~=0||n<=0
        disp('Erro. Número de intervalos deve ser um inteiro positivo.');
        I = 'erro'; erro = I;
        return;
    end

    %%%PROCESSAMENTO%%%
    h = (b-a)/n; %define tamanho do intervalo
    x = a:h:b; %discretização do domínio
    g = inline(f); %cria função inline a partir de f
    soma = 0; %declaração da variável do somatório
    for i = 1:n
        %calcula imagem de cada ponto médio e acrescenta à soma
        soma = soma + g((x(i)+x(i+1))/2);
    end

    %%%SAÍDA%%%
    I = h*soma; %cálculo da integral

    %cálculo do erro
    erro = ((b-a)/24)*maximo(diff(f,2),a,b)*h^2;
end
```

Figura 13: função em MATLAB® que implementa o método do Ponto Médio para integração numérica.

ANEXO 2

```
function [ I ] = IntegralTrapezio(a,b,f,n)
%Calcula integral numericamente pelo método do trapézio.
%Parâmetros: [ I ] = IntegralTrapezio(a,b,f,n)
%a: ponto inicial do domínio de integração
%b: ponto final do domínio de integração
%f: função do integrando
%n: número de intervalos
%I: valor da integral (saída)

    %%%VALIDAÇÃO%%%
    if isa(f,'inline')==false
        f = inline(f); %transforma string em uma função inline
    end
    if isnumeric(a)==false||isnumeric(b)==false||isnumeric(n)==false
        disp('Erro. Parâmetros a, b e n devem ser numéricos.');
        I = 'erro';
        return;
    end
    if a>b % a deve ser menor que b
        disp('Erro. Parâmetro a deve ser menor que b.');
        I = 'erro';
        return;
    elseif a==b % a deve ser diferente de b
        disp('Erro. Parâmetros a e b devem ser diferentes.');
        I = 'erro';
        return;
    end
    if mod(fix(n),n)~=0||n<=0
        disp('Erro. Número de intervalos deve ser um inteiro positivo.');
        I = 'erro';
        return;
    end

    %%%PROCESSAMENTO%%%
    h = (b-a)/n; %define tamanho do intervalo
    x = a:h:b; %discretização do domínio
    y = f(x); %imagens dos pontos do domínio discretizado
    soma = 0; %declaração da variável do somatório
    for i = 1:n
        soma = soma + (y(i)+y(i+1));
    end

    %%%SAÍDA%%%
    I = (h/2)*soma;
end
```

Figura 14: função em MATLAB® que implementa o método do Trapézio (contínuo) para integração numérica.

ANEXO 3

```
function [ I ] = IntTrapz(X,Y)
%Calcula integral numericamente pelo método do Trapézio
%Parâmetros: [ I ] = IntTrapz(X,Y)
%X: vetor do domínio
%Y: vetor de imagens
%I: valor da integral (saída)

    %%%VALIDAÇÃO%%%
    if isnumeric(X)==false||isnumeric(Y)==false
        disp('Erro. Parâmetros devem ser numéricos.');
        I = 'erro'; return;
    end
    if(((size(X,1)>1)&&(size(X,2)>1))||((size(Y,1)>1)&&(size(Y,2)>1)))
        disp('Erro! X ou Y não é um vetor.');
        I = 'erro'; return;
    elseif(length(X)~=length(Y))%x e y devem ter mesmo número de elementos
        disp('Erro! Dimensões de X e Y não são equivalentes.');
        I = 'erro'; return;
    end
    if length(X)<=1
        disp('Erro. Vetores devem ter ao menos 2 elementos.');
        I = 'erro'; return;
    end

    %%%PROCESSAMENTO E SAÍDA%%%
    I = 0;
    for i = 1:length(X)-1
        %calcula a interpolação do ponto médio e multiplica pelo tamanho do
        %intervalo
        I = I + ((Y(i+1)+Y(i))*(X(i+1)-X(i)))/2;
    end
end
```

Figura 15: função em MATLAB® que implementa o método do Trapézio (discreto) para integração numérica.

ANEXO 4

```
function [ I ] = IntegralSimpson13(a,b,f,n)
%Calcula integral numericamente pelo método de 1/3 de Simpson.
%Parâmetros: [ I ] = IntegralTrapezio(a,b,f,n)
%a: ponto inicial do domínio de integração
%b: ponto final do domínio de integração
%f: função do integrando
%n: número de intervalos
%I: valor da integral (saída)

    %%%VALIDAÇÃO%%%
    if isa(f,'inline')==false
        f = inline(f); %transforma string em uma função inline
    end
    if isnumeric(a)==false||isnumeric(b)==false||isnumeric(n)==false
        disp('Erro. Parâmetros a, b e tol devem ser numéricos.');
        I = 'erro';
        return;
    end
    if a>b % a deve ser menor que b
        disp('Erro. Parâmetro a deve ser menor que b.');
        I = 'erro';
        return;
    elseif a==b % a deve ser diferente de b
        disp('Erro. Parâmetros a e b devem ser diferentes.');
        I = 'erro';
        return;
    end
    if mod(fix(n),n)~=0||n<=0||mod(n,2)~=0
        disp('Erro. Número de intervalos deve ser inteiro, positivo e par.');
        I = 'erro';
        return;
    end

    %%%PROCESSAMENTO%%%
    h = (b-a)/n; %define tamanho do intervalo
    x = a:h:b; %discretização do domínio
    y = f(x); %imagens dos pontos do domínio discretizado
    soma = 0; %declaração da variável do somatório
    for i = 2:n %pontos internos
        if mod(i,2)==0
            soma = soma + 4*y(i);
        else
            soma = soma + 2*y(i);
        end
    end
    soma = (h/3)*soma; %fator externo ao somatório

    %%%SAÍDA%%%
    I = soma + (h/3)*(y(1)+y(n+1)); %adiciona pontos das extremidades
end
```

Figura 16: função em MATLAB® que implementa o método de 1/3 de Simpson para integração numérica.

ANEXO 5

```
function [ I ] = IntegralSimpson38(a,b,f,n)
%Calcula integral numericamente pelo método de 3/8 de Simpson.
%Parâmetros: [ I ] = IntegralTrapezio(a,b,f,n)
%a: ponto inicial do domínio de integração
%b: ponto final do domínio de integração
%f: função do integrando
%n: número de intervalos
%I: valor da integral (saída)

    %%%VALIDAÇÃO%%%
    if isa(f,'inline')==false
        f = inline(f); %transforma string em uma função inline
    end
    if isnumeric(a)==false||isnumeric(b)==false||isnumeric(n)==false
        disp('Erro. Parâmetros a, b e tol devem ser numéricos.');
        I = 'erro'; return;
    end
    if a>b % a deve ser menor que b
        disp('Erro. Parâmetro a deve ser menor que b.');
        I = 'erro'; return;
    elseif a==b % a deve ser diferente de b
        disp('Erro. Parâmetros a e b devem ser diferentes.');
        I = 'erro'; return;
    end
    if mod(fix(n),n)~=0||n<=0||mod(n,3)~=0
        disp('Erro. Número de intervalos deve ser inteiro, positivo e divisível por 3.');
        I = 'erro'; return;
    end
    %%%PROCESSAMENTO%%%
    h = (b-a)/n; %define tamanho do intervalo
    x = a:h:b; %discretização do domínio
    y = f(x); %imagens dos pontos do domínio discretizado
    soma = 0; %declaração da variável do somatório
    j = 1; %variável auxiliar
    for i = 2:n %pontos internos
        if mod(j,3)==0 %entra nessa condição a cada 3 iterações
            soma = soma + 2*y(i);
            j = 0; %restaura valor de j
        else
            soma = soma + 3*y(i);
        end
        j = j + 1; %incrementa j
    end
    soma = (3*h/8)*soma; %fator externo ao somatório
    %%%SAÍDA%%%
    I = soma + (3*h/8)*(y(1)+y(n+1)); %adiciona pontos das extremidades
end
```

Figura 17: função em MATLAB® que implementa o método de 3/8 de Simpson para integração numérica.

ANEXO 6

```
function [ I,erro ] = IntegralQuadraturaGauss(a,b,f,n)
%Calcula integral numericamente pelo método da Quadratura de Gauss
%utilizando de 2 até 6 pontos no domínio.
%Parâmetros: [ I,erro ] = IntegralTrapezio(a,b,f,n)
%a: ponto inicial do domínio de integração
%b: ponto final do domínio de integração
%f: função do integrando
%n: número de pontos
%I: valor da integral (saída)
%erro: estimativa de erro de truncamento (saída)

    %%%VALIDAÇÃO%%%
    if isa(f,'sym')==false
        f = sym(f); %transforma string em uma função simbólica
    end
    if isnumeric(a)==false||isnumeric(b)==false||isnumeric(n)==false
        disp('Erro. Parâmetros a, b e tol devem ser numéricos.');
        I = 'erro'; erro = I;
        return;
    end
    if a>b % a deve ser menor que b
        disp('Erro. Parâmetro a deve ser menor que b.');
        I = 'erro'; erro = I;
        return;
    elseif a==b % a deve ser diferente de b
        disp('Erro. Parâmetros a e b devem ser diferentes.');
        I = 'erro'; erro = I;
        return;
    end
    if mod(fix(n),n)~=0||n<2||n>6
        disp('Erro. Número de pontos deve ser um inteiro entre 2 e 6.');
        I = 'erro'; erro = I;
        return;
    end

    %%%PROCESSAMENTO%%%
    %definindo pontos e pesos
    if n==2
        %coeficientes (pesos)
        c = [1 1];
        %pontos do domínio
        x = [-0.577350269 0.577350269];
        df = diff(f,4); %calcula derivada de 4ª ordem de f
    elseif n==3
        %coeficientes (pesos)
        c = [0.5555556 0.8888889 0.5555556];
        %pontos do domínio
        x = [-0.774596669 0 0.774596669];
        df = diff(f,6); %calcula derivada de 6ª ordem de f
    elseif n==4
        %coeficientes (pesos)
        c = [0.3478548 0.6521452 0.6521452 0.3478548];
        %pontos do domínio
```

```
        x = [-0.861136312 -0.339981044 0.339981044 0.861136312];
        df = diff(f,8); %calcula derivada de 8ª ordem de f
    elseif n==5
        %coeficientes (pesos)
        c = [0.2369269 0.4786287 0.5688889 0.4786287 0.2369269];
        %pontos do domínio
        x = [-0.906179846 -0.538469310 0 0.538469310 0.906179846];
        df = diff(f,10); %calcula derivada de 10ª ordem de f
    elseif n==6
        %coeficientes (pesos)
        c = [0.1713245 0.3607616 0.4679139 0.4679139 0.3607616 0.1713245];
        %pontos do domínio
        x = [-0.932469514 -0.661209386 -0.238619186 0.238619186];
        x = [x  0.661209386 0.932469514];
        df = diff(f,12); %calcula derivada de 12ª ordem de f
    end

    %mudança de variável
    g = inline('(t*(b-a)+a+b)/2'); %cria função de mudança de variável
    dt = (b-a)/2; %termo jacobiano da mudança de variável
    f = inline(f); %transforma f de simbólica para inline

    %%%SAÍDA%%%
    %cálculo da integral
    I = 0; %declaração de variável de saída inicialmente nula
    for i = 1:n
        I = I + c(i)*f(g(a,b,x(i)))*dt;
    end
    %cálculo do erro
    erro = ((2^(2*n+3))*(factorial(n+1))^4)/((2*n+3)*(factorial(2*n+2))^3);
    erro = erro*(maximo(df,a,b));
end
```

Figura 18: função em MATLAB® que implementa o método da Quadratura de Gauss-Legendre para integração numérica.

ANEXO 7

```
function [ n ] = NumMinIntTrapezio(a,b,f,tol)
%Esta função calcula o número mínimo de intervalos para garantir uma
%exatidão definida no processo de integração numérica pelo método do
%trapézio.
%Parâmetros: [ n ] = NumMinIntTrapezio(a,b,f,tol)
%a: extremo esquerdo do domínio de integração
%b: extremo direito do domínio de integração
%f: função do integrando
%tol: tolerância de exatidão.

    %%%VALIDAÇÃO%%%
    if isa(f,'sym')==false
        f = sym(f); %transforma string em uma função simbólica
    end
    if isnumeric(a)==false||isnumeric(b)==false||isnumeric(tol)==false
        disp('Erro. Parâmetros a, b e tol devem ser numéricos.');
        n = 'erro';
        return;
    end
    if a>b % a deve ser menor que b
        disp('Erro. Parâmetro a deve ser menor que b.');
        n = 'erro';
        return;
    elseif a==b % a deve ser diferente de b
        disp('Erro. Parâmetros a e b devem ser diferentes.');
        n = 'erro';
        return;
    end

    %%%PROCESSAMENTO%%%
    d2f = diff(f,2); %calcula segunda derivada de f
    ymax = maximo(d2f,a,b); %calcula ponto máximo de d2f

    %calcula tamanho do passo de acordo com fórmula de estimativa de erro
    %do método do trapézio
    h = sqrt((12*abs(tol))/(abs(b-a)*abs(ymax)));

    %%%SAÍDA%%%
    n = ceil((b-a)/h); %calcula o número de passos no intervalo e arredonda
                       %para cima
end
```

Figura 19: função em MATLAB® que calcula o número mínimo de subintervalos para garantir uma tolerância determinada para o método do Trapézio.

ANEXO 8

```
function [ n ] = NumMinIntSimpson13(a,b,f,tol)
%Esta função calcula o número mínimo de intervalos para garantir uma
%exatidão definida no processo de integração numérica pelo método de
%Simpson de 1/3.
%Parâmetros: [ n ] = NumMinIntSimpson13(a,b,f,tol)
%a: extremo esquerdo do domínio de integração
%b: extremo direito do domínio de integração
%f: função do integrando
%tol: tolerância de exatidão.

    %%%VALIDAÇÃO%%%
    if isa(f,'sym')==false
        f = sym(f); %transforma string em uma função simbólica
    end
    if isnumeric(a)==false||isnumeric(b)==false||isnumeric(tol)==false
        disp('Erro. Parâmetros a, b e tol devem ser numéricos.');
        n = 'erro';
        return;
    end
    if a>b % a deve ser menor que b
        disp('Erro. Parâmetro a deve ser menor que b.');
        n = 'erro';
        return;
    elseif a==b % a deve ser diferente de b
        disp('Erro. Parâmetros a e b devem ser diferentes.');
        n = 'erro';
        return;
    end

    %%%PROCESSAMENTO%%%
    d4f = diff(f,4); %calcula segunda derivada de f
    ymax = maximo(d4f,a,b); %calcula ponto máximo de d4f

    %calcula tamanho do passo de acordo com fórmula de estimativa de erro
    %do método do trapézio
    h = ((180*abs(tol))/(abs(b-a)*abs(ymax)))^(1/4);

    %%%SAÍDA%%%
    n = ceil((b-a)/h); %calcula o número de passos no intervalo
    if mod(n,2)~=0 %se n for impar, acrescenta mais uma unidade
        n = n+1;
    end
end
```

Figura 20: função em MATLAB® que calcula o número mínimo de subintervalos para garantir uma tolerância determinada para o método de 1/3 de Simpson.

ANEXO 9

```
function [ n ] = NumMinIntPontoMedio(a,b,f,tol)
%Esta função calcula o número mínimo de intervalos para garantir uma
%exatidão definida no processo de integração numérica pelo método do
%ponto médio.
%Parâmetros: [ n ] = NumMinIntTrapezio(a,b,f,tol)
%a: extremo inferior do domínio de integração
%b: extremo superior do domínio de integração
%f: função do integrando
%tol: tolerância de exatidão.

    %%%VALIDAÇÃO%%%
    if isa(f,'sym')==false
        f = sym(f); %transforma string em uma função simbólica
    end
    if isnumeric(a)==false||isnumeric(b)==false||isnumeric(tol)==false
        disp('Erro. Parâmetros a, b e tol devem ser numéricos.');
        n = 'erro';
        return;
    end
    if a>b % a deve ser menor que b
        disp('Erro. Parâmetro a deve ser menor que b.');
        n = 'erro';
        return;
    elseif a==b % a deve ser diferente de b
        disp('Erro. Parâmetros a e b devem ser diferentes.');
        n = 'erro';
        return;
    end

    %%%PROCESSAMENTO%%%
    d2f = diff(f,2); %calcula segunda derivada de f
    ymax = maximo(d2f,a,b); %calcula ponto máximo de d2f

    %calcula tamanho do passo de acordo com fórmula de estimativa de erro
    %do método do trapézio
    h = sqrt((24*abs(tol))/(abs(b-a)*abs(ymax)));

    %%%SAÍDA%%%
    n = ceil((b-a)/h); %calcula o número de passos no intervalo e arredonda
                       %para cima
end
```

Figura 21: função em MATLAB® que calcula o número mínimo de subintervalos para garantir uma tolerância determinada para o método do Ponto Médio.

ANEXO 10

```
function [ Ymax ] = maximo(f,a,b)
%Esta função retorna o ponto y máximo de uma função num intervalo [a,b]
%Parâmetros: [ Xmax,Ymax ] = maximo(f,a,b)
%f: função (simbólica)
%a: extremo inferior do intervalo
%b: extremo superior do intervalo
%Ymax: coordenada y do ponto máximo (saída)

    %%%VALIDAÇÃO%%%
    if ischar(f)==true
        f = sym(f); %transforma string em uma função simbólica
    end
    if isnumeric(a)==false||isnumeric(b)==false
        disp('Erro. Parâmetros a e b devem ser numéricos.'); Ymax = 'erro'; return;
    end
    if a>b % a deve ser menor que b
        disp('Erro. Parâmetro a deve ser menor que b.'); Ymax = 'erro'; return;
    elseif a==b % a deve ser diferente de b
        disp('Erro. Parâmetros a e b devem ser diferentes.'); Ymax = 'erro'; return;
    end
    %%%PROCESSAMENTO%%%
    g = -f;%cria função que é a negativa de f
    f = inline(f); %transforma f em inline
    g = inline(g); %transforma g em inline
    Xmax = fminbnd(g,a,b); %calcula o mínimo de g, que é o máximo de f
    %%%SAÍDA%%%
    Ymax = f(Xmax); %avalia f no ponto x máximo
end
```

Figura 22: função em MATLAB® que calcula valor máximo de uma função em um intervalo dado.

ANEXO 11

```
function [ erro ] = ErroRelativoPercentual(Vexato,Vaproximado)
%Calcula o erro relativo percentual de um valor aproximado
%Parâmetros: [ erro ] = ErroRelativoPercentual(Vexato,Vaproximado)
%Vexato: valor exato
%Vaproximado: valor aproximado
%erro: erro relativo percentual (saída)

    %%%VALIDAÇÃO%%%
    if isnumeric(Vexato)==false||isnumeric(Vaproximado)==false
        disp('Erro. Todos os parâmetros devem ser numéricos.'); erro = 'erro'; return;
    end
    %%%PROCESSAMENTO E SAÍDA%%%
    erro = (abs(Vexato - Vaproximado)./abs(Vexato))*100;
end
```

Figura 23: função em MATLAB® que calcula o erro relativo percentual entre um vetor de valores exatos e outro de valor aproximado.

ANEXO 12

```
%valores calculados analiticamente
for i = 1:12
    Vexato1(i,1) = pi;
    Vexato2(i,1) = -(2*pi)/15;
end
%declaração das funções
syms x; f1 = x*sin(x); f2 = x*sin(15*x);
%vetor com nomes dos métodos
metodos = string({'Analítico';'Trapézio 6pt';'Trapézio 12pt';...
    'Simpson 1/3 6pt';'Simpson 1/3 12pt';'Simpson 3/8 6pt';...
    'Simpson 3/8 12pt';'QG-Legendre 2pt';'QG-Legendre 3pt';...
    'QG-Legendre 4pt';'QG-Legendre 5pt';'QG-Legendre 6pt'});
metodos = cellstr(metodos);
%cálculo das integrais
%método do trapézio
I1(1,1) = IntegralTrapezio(0,pi,f1,6);
I1(2,1) = IntegralTrapezio(0,pi,f1,12);
I2(1,1) = IntegralTrapezio(0,2*pi,f2,6);
I2(2,1) = IntegralTrapezio(0,2*pi,f2,12);
%método de 1/3 de Simpson
I1(3,1) = IntegralSimpson13(0,pi,f1,6);
I1(4,1) = IntegralSimpson13(0,pi,f1,12);
I2(3,1) = IntegralSimpson13(0,2*pi,f2,6);
I2(4,1) = IntegralSimpson13(0,2*pi,f2,12);
%método de 3/8 de Simpson
I1(5,1) = IntegralSimpson38(0,pi,f1,6);
I1(6,1) = IntegralSimpson38(0,pi,f1,12);
I2(5,1) = IntegralSimpson38(0,2*pi,f2,6);
I2(6,1) = IntegralSimpson38(0,2*pi,f2,12);
%método da Quadratura de Gauss-Legendre
I1(7,1) = IntegralQuadraturaGauss(0,pi,f1,2);
I1(8,1) = IntegralQuadraturaGauss(0,pi,f1,3);
I1(9,1) = IntegralQuadraturaGauss(0,pi,f1,4);
I1(10,1) = IntegralQuadraturaGauss(0,pi,f1,5);
I1(11,1) = IntegralQuadraturaGauss(0,pi,f1,6);
I2(7,1) = IntegralQuadraturaGauss(0,2*pi,f2,2);
I2(8,1) = IntegralQuadraturaGauss(0,2*pi,f2,3);
I2(9,1) = IntegralQuadraturaGauss(0,2*pi,f2,4);
I2(10,1) = IntegralQuadraturaGauss(0,2*pi,f2,5);
I2(11,1) = IntegralQuadraturaGauss(0,2*pi,f2,6);
%inserindo valores exatos nos vetores dos resultados
I1 = [Vexato1(1);I1]; I2 = [Vexato2(1);I2];
%calculando erro relativo percentual
erro1 = ErroRelativoPercentual(Vexato1,I1);
erro2 = ErroRelativoPercentual(Vexato2,I2);
%criando tabela de saída
nomes = string({'Integral_i' 'Erro_Rel_i' 'Integral_ii','Erro_Rel_ii'});
nomes = cellstr(nomes);
T = table(I1,erro1,I2,erro2,'RowNames',metodos,'VariableNames',nomes)
```

Figura 24: script em MATLAB® que resolve a questão 3.

ANEXO 13

```
%Questão 4
tic;
H = [0;1.8;2;4;4;6;4;3.6;3.4;2.8;0];
U = [0;0.1;0.14;0.36;0.38;0.47;0.43;0.33;0.25;0.18;0];
y = 0:2:20;
delta_y = y(length(y))-y(1);

%4.a) Calcular profundidade média
ProfundidadeMedia = (1/(delta_y))*IntTrapz(y,H)

%4.b) Calcular área da seção transversal
Area = IntTrapz(y,H)

%4.c) Calcular velocidade média
VelocidadeMedia = (1/(delta_y))*IntTrapz(y,U)

%4.d) Calcular vazão do escoamento
Vazao = IntTrapz(y,H.*U)
toc;
```

Figura 25: script em MATLAB® que resolve a questão 4.