**Numérico M9**

**Q1 - Aproxime \int_2^3 cos(2 x) dx utilizando somas de Riemann a esquerda com n=40 intervalos.**

clear

/\*\* Somas de riemmann a esquerda

\* a: limite esquerdo

\* b: limite direito

\* n: numero de iteracoes

\*

\* S: area apos integrar a funcao

\*/

function **S**=riemmann(**a**, **b**, **n**)

h=(**b**-**a**)/**n**;

x=linspace(**a**,**b**,**n**+1);

**S**=0;

for i=1:**n**

A=f(x(i))\*h;

**S**=**S**+A;

end

endfunction

*// Funcao a ser integrada e' dada por y*

function **y**=f(**x**)

**y**=cos(2\***x**);

endfunction

*// Devolve a integral definida da funcao para comparacao com metodos iterativos*

function **v**=integral(**limiteEsquerda**, **limiteDireita**, **funcao**)

**v** = intg(**limiteEsquerda**, **limiteDireita**, **funcao**);

endfunction

*// valor da integral fica em riemmann(inicio, fim, numero\_intervalos)*

*// valor da funcao fica em f(valor\_x)*

disp(riemmann(2,3,40))

**Q2 - Aproxime \int_0^3 sin(x+33) dx utilizando o método de Simspon com h=0.5.**

function **S**=simpson(**a**, **b**, **n**)

h=(**b**-**a**)/**n** *//n numero de intervalos*

x=linspace(**a**,**b**,**n**+1) *//cria um vetor que vai de a até b, com n+1 pontos*

**S**=0

for i=1:**n**

x1=x(i) *// extremo esquerdo*

x3=x(i+1) *//extremo direito*

x2=x1+h/2 *//ponto médio*

A1=1/6; A2=4/6; A3=1/6

ds=(A1\*f(x1)+A2\*f(x2)+A3\*f(x3))\*h

**S**=**S**+ds

end

endfunction

function **y**=f(**x**)

**y**=sin(**x**+33)

endfunction

*//no scilab, entra com o comando "simpson(a,b,n)", sendo a o limite inferior da integral,*

*//b o limite superior da integral, e n o número de intervalos que deseja (o problema não dará*

*//o numero de intervalos)*

*//calcula o numero de intervalos com h fixo*

a=0

b=3

h=0.5 *//h fixo*

n=abs((a-b)/h)

disp(n)

disp(simpson(a,b,n))

**Q3 - Aproxime \int_3^4 cos(x/6) dx utilizando o método do trapézio com n=6 intervalos.**

clear

/\*\*

\* a: limite esquerdo

\* b: limite direito

\* n: numero de iteracoes/intervalos

\* S: area apos integrar a funcao

\*

\* >>> n=abs((a-b)/h) caso n nao seja dado

\*/

function **S**=trapezio(**a**, **b**, **n**)

h=(**b**-**a**)/**n**;

x=linspace(**a**,**b**,**n**+1); *// cria vetor de a ate' b com n+1 pontos*

**S**=0;

for i=1:**n** *// percorre todos intervalos*

x1 = x(i); *// esquerda do intervalo*

x2 = x(i+1); *// direita do intervalo*

A1 =1/2;

A2 =1/2;

dS =(A1\*f(x1)+A2\*f(x2))\*h; *// aprox. da area ds*

**S**=**S**+dS;

end

endfunction

*// Funcao a ser integrada e' dada por y*

function **y**=f(**x**)

**y**=cos(**x**/6);

endfunction

*// Devolve a integral definida da funcao para comparacao com metodos iterativos*

function **v**=integral(**limiteEsquerda**, **limiteDireita**, **funcao**)

**v** = intg(**limiteEsquerda**, **limiteDireita**, **funcao**);

endfunction

*// valor da integral fica em trapezio(inicio, fim, numero\_intervalos)*

*// valor da funcao fica em f(valor\_x)*

disp(trapezio(3,4,6))

**Q4 - Aproxime \int_5^6 \cos(x)+\cos(3*x) dx utilizando o método de Simpson com com n=79 intervalos.**

clear

/\*\*

\* a: limite esquerdo

\* b: limite direito

\* n: numero de iteracoes

\*

\* S: area apos integrar a funcao

\*/

function **S**=simpson(**a**, **b**, **n**)

h=(**b**-**a**)/**n**; *// tamanho do intervalo*

x=linspace(**a**,**b**,**n**+1); *// vetor de a ate' b com n+1 elementos*

**S**=0;

for i=1:**n**

x1=x(i); *// extremo esquerdo*

x3=x(i+1); *// extremo direito*

x2=x1+h/2; *// ponto medio*

A1 =1/6; A2 =4/6; A3=1/6;

dS =(A1\*f(x1)+A2\*f(x2)+A3\*f(x3))\*h;

**S**=**S**+dS;

end

endfunction

*// Funcao para ser integrada*

function **y**=f(**x**)

**y**=cos(**x**)+cos(3\***x**);

endfunction

*// Devolve a integral definida da funcao para comparacao com metodos iterativos*

function **v**=integral(**limiteEsquerda**, **limiteDireita**, **funcao**)

**v** = intg(**limiteEsquerda**, **limiteDireita**, **funcao**);

endfunction

*// resposta pode ser obtida em: simpson(inicio,final,numero\_intervalos)*

disp(simpson(5,6,79))

**Q5 - Aproxime \int_0^1  \cos(4x+11) dx com 7 dígitos significativos.**

*function* ***S****=trapezio(****a****,* ***b****,* ***n****)*

*h=(****b****-****a****)/****n*** *//n numero de intervalos*

*x=linspace(****a****,****b****,****n****+1) //cria um vetor que vai de a até b, com n+1 pontos*

***S****=0*

*for i=1:****n***

*x1=x(i) // extremo esquerdo*

*x2=x(i+1) //extremo direito*

*A1=1/2; A2=1/2*

*ds=(A1\*f(x1)+A2\*f(x2))\*h*

***S****=****S****+ds*

*end*

*endfunction*

*function* ***y****=f(****x****)*

***y****=cos(4\*****x****+11)*

*endfunction*

*//no scilab, entra com o comando "trapezio(a,b,n)", sendo a o limite inferior da integral,*

*//b o limite superior da integral, e n o número de intervalos que deseja (o problema não dará*

*//o numero de intervalos)*

*//para essa questão, trocar o valor de n e avaliar a partir de qual o n a resposta fica*

*//com 3 dígitos após a vírgula sem alterar*

*disp(trapezio(0,1,2987)) //tentativa e erro.*

**Q6 - Ao utilizar o método do trapézio para calcular uma integral com 30 intervalos temos um erro de aproximadamente 0.001. Se utilizarmos 90 intervalos, de quando será o erro aproximadamente?**

clear

/\*\*

\* Ao utilizar o método do trapézio para calcular uma integral com

\* intervalos\_depois intervalos temos um erro de aproximadamente

\* erro\_antes. Se utilizarmos intervalos\_depoisintervalos, de quando

\* será o erro aproximadamente?

\*/

function **n**=erro\_trapezio(**erro\_antes**, **intervalos\_antes**, **intervalos\_depois**)

vezes\_mais = **intervalos\_depois**/**intervalos\_antes**

**n** = **erro\_antes**/(vezes\_mais^2)

endfunction

disp(erro\_trapezio(0.001,30,90))

**Q7 - Ao utilizar o método de Simpson para calcular uma integral com h=0.02 temos um erro de aproximadamente 0.005. Se utilizarmos h=0.005, de quando será o erro aproximadamente?**

**Q8 - Aproxime \int_2^3 1/(6 x) dx utilizando somas de Riemann a esquerda com h=0.0025.**

function **S**=riemann(**a**, **b**, **n**)

*// n numero de intervalos*

h=(**b**-**a**)/**n**

x=linspace(**a**,**b**,**n**+1)

**S**=0

for i=1:**n**

A=f(x(i))\*h

*//m=(x(i)+x(i+1))/2;*

*//A=f(m)\*h*

**S**=**S**+A

end

endfunction

function **y**=f(**x**)

**y**=1/(6\***x**)

endfunction

*//no scilab, entra com o comando "riemann(a,b,n)", sendo a o limite inferior da integral,*

*//b o limite superior da integral, e n o número de intervalos que deseja (o problema não dará*

*//o numero de intervalos)*

*//calcula o numero de intervalos com h fixo*

a=2

b=3

h=0.0025 *//h fixo*

n=abs((a-b)/h)

disp(n)

disp(riemann(a,b,n))

**Q9 - Aproxime \int_1^{3.5} x^3-15*x dx utilizando o método do trapézio com h=0.1.**

function **S**=trapezio(**a**, **b**, **n**)

h=(**b**-**a**)/**n** *//n numero de intervalos*

x=linspace(**a**,**b**,**n**+1) *//cria um vetor que vai de a até b, com n+1 pontos*

**S**=0

for i=1:**n**

x1=x(i) *// extremo esquerdo*

x2=x(i+1) *//extremo direito*

A1=1/2; A2=1/2

ds=(A1\*f(x1)+A2\*f(x2))\*h

**S**=**S**+ds

end

endfunction

function **y**=f(**x**)

**y**=**x**^3-15\***x**

endfunction

*//no scilab, entra com o comando "trapezio(a,b,n)", sendo a o limite inferior da integral,*

*//b o limite superior da integral, e n o número de intervalos que deseja (o problema não dará*

*//o numero de intervalos)*

*//calcula o numero de intervalos com h fixo*

a=1

b=3.5

h=0.1 *//h fixo*

n=abs((a-b)/h)

disp(n)

disp(trapezio(a,b,n))

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**M10**

**Q1 - Aproxime \int_1^{10} sin(8*x) dx utilizando quadratura gaussiana com 2 nós e 5 intervalos.**

clear

/\*\* Divide o intervalo de a até b em n\_intervalos, e em cada intervalo e

\* utiliza uma quadratura gaussiana com n\_nodes (nos) para calcular a

\* integral de f(x), a qual e' definida apos essa funcao.

\* a: limite esquerdo

\* b: limite direito

\* n\_intervalos: numero de iteracoes/intervalos

\* n\_nodes: numero de nos (1 - 5 estao mapeados)

\*

\* S: area apos integrar a funcao

\*/

function **S**=gaussiana(**a**, **b**, **n\_intervalos**, **n\_nodes**)

h=(**b**-**a**)/**n\_intervalos**

x=linspace(**a**,**b**,**n\_intervalos**+1) *//cria um vetor que vai de a até b, com n+1 pontos*

*// Mapeamento de mudancas de intervalo (t: nodes, w: pesos)//////////*

*// n=1*

t1=[0];

w1=[2];

*// n=2*

t2=[-sqrt(3)/3 sqrt(3)/3];

w2=[1 1];

*// n=3*

t3=[0 sqrt(3/5) -sqrt(3/5)];

w3=[8/9 5/9 5/9];

*// n=4*

t4=[sqrt((3-2\*sqrt(6/5))/7) -sqrt((3-2\*sqrt(6/5))/7) sqrt((3+2\*sqrt(6/5))/7) -sqrt((3+2\*sqrt(6/5))/7)];

w4=[(18+sqrt(30))/36 (18+sqrt(30))/36 (18-sqrt(30))/36 (18-sqrt(30))/36];

*//n=5*

t5=[0 (1/3)\*sqrt(5-2\*sqrt(10/7)) -(1/3)\*sqrt(5-2\*sqrt(10/7)) (1/3)\*sqrt(5+2\*sqrt(10/7)) -(1/3)\*sqrt(5+2\*sqrt(10/7))];

w5=[128/225 (322+13\*sqrt(70))/900 (322+13\*sqrt(70))/900 (322-13\*sqrt(70))/900 (322-13\*sqrt(70))/900];

*/////////////////////////////////////////////////////////////////////*

**S**=0 *// valor inicial da integral*

*// Percorre todos os intervalos para calcular a quadratura em cada uma*

for i=1:**n\_intervalos**

*// Obtem alfa e beta*

alpha=(x(i+1)-x(i))/2

betha=(x(i+1)+x(i))/2

*// Escolhe o alfa e beta de acordo com o numero de nos*

if (**n\_nodes** == 1) then

x1=alpha\*t1(1)+betha;

elseif (**n\_nodes** == 2) then

alpha=(x(i+1)-x(i))/2

betha=(x(i+1)+x(i))/2

x1=alpha\*t2(1)+betha;

x2=alpha\*t2(2)+betha;

A=(w2(1)\*f(x1)+w2(2)\*f(x2))\*h/2

elseif (**n\_nodes** == 3) then

x1=alpha\*t3(1)+betha;

x2=alpha\*t3(2)+betha;

x3=alpha\*t3(3)+betha;

A=(w3(1)\*f(x1)+w3(2)\*f(x2)+w3(3)\*f(x3))\*h/2

elseif (**n\_nodes** == 4) then

x1=alpha\*t4(1)+betha;

x2=alpha\*t4(2)+betha;

x3=alpha\*t4(3)+betha;

x4=alpha\*t4(4)+betha;

A=(w4(1)\*f(x1)+w4(2)\*f(x2)+w4(3)\*f(x3)+w4(4)\*f(x4))\*h/2

elseif (**n\_nodes** == 5) then

x1=alpha\*t5(1)+betha;

x2=alpha\*t5(2)+betha;

x3=alpha\*t5(3)+betha;

x4=alpha\*t5(4)+betha;

x5=alpha\*t5(5)+betha;

A=(w5(1)\*f(x1)+w5(2)\*f(x2)+w5(3)\*f(x3)+w5(4)\*f(x4)+w5(5)\*f(x5))\*h/2

else

error("--> Numero de nodes nao mapeado (ultimo paramatero).")

end

**S**=**S**+A

end

endfunction

*//definição da função*

function **y**=f(**x**)

**y**=sin(8\***x**)

endfunction

disp(gaussiana(1,10,5,2))

**Q2 - Aproxime \int_1^{10} sin(8*x) dx utilizando quadratura gaussiana com 3 nós e 5 intervalos.**

MESMO CÓDIGO DA 1 MAS NO FINAL:

disp(gaussiana(1,10,5,3))

**Q3 - Aproxime \int_1^{10} sin(8*x) dx utilizando quadratura gaussiana com 4 nós e 9 intervalos.**

MESMO CÓDIGO DA 1 MAS NO FINAL:

disp(gaussiana(1,10,9,4))

**Q4 - Aproxime \int_2^{4} sin(7*x+1) dx utilizando quadratura gaussiana com 3 nós e 8 intervalos.**

**Q5 - Aproxime \int_2^{4} sin(7*x+1) dx utilizando quadratura gaussiana com 3 nós e 90 intervalos.**

MESMO CÓDIGO DA 4 MAS NO FINAL:

disp(gaussiana(2,4,90,3))

**Q6 - Sejam os nós x=[0.1, 0.8, 1]. Encontre os pesos A_i da quadratura I=A_1f(x_1)+A_2f(x_2)+A_3f(x_3) tal que o erro seja o menor possível para aproximar a integral de f no intervalo 0 a 1. Forneça como resposta A_2**

clear

/\*\*

\* Para encontrar pesos wi (ou Ai) da quadratura

\* I = w1\*f(x1) + ... + wn\*f(xn) com o menor error possivel

\*

\* vetor\_nodes: vetor x de nos

\* lim\_inicial: limite inicial da integral

\* lim\_final: limite final da integral

\*

\* w: vetor de pesos (retorno)

\*/

function **w**=pesos(**vetor\_nodes**, **lim\_inicial**, **lim\_final**)

len\_nodes = length(**vetor\_nodes**);

*// Monta matriz A*

for i=1:len\_nodes

for j=1:len\_nodes

if (i == 1) then

A(i,j) = i; *// 1a linha so' tem 1s*

else

A(i,j) = **vetor\_nodes**(j)^(i-1);

end

end

end

*//disp('A:')*

*//disp(A)*

*// Monta matriz resultado b*

for i=1:len\_nodes

B(i) = (**lim\_final**^i - **lim\_inicial**^i)/i;

end

*//disp('B:')*

*//disp(B)*

*// Obtem pesos*

**w**=inv(A)\*B;

endfunction

disp(pesos([0.1,0.8,1],0,1)) *//pegar o valor pedido na questão.*

**Q7 - Sejam os nós x=[0.1, 0.7, 1]. Encontre os pesos w_i da quadratura I=w_1f(x_1)+w_2f(x_2)+w_3f(x_3) tal que o erro seja o menor possível para aproximar a integral de f no intervalo 0 a 1. Forneça como resposta w_1.**

MESMO CÓDIGO DA 6 MAS NO FINAL:

disp(pesos([0.1,0.7,1],0,1)) *//pegar o valor pedido na questão.*

**Q8 -Sejam os nós x=[0, 3/5, 5/5, 2]. Encontre os pesos A_i da quadratura I=A_1f(x_1)+...+A_4f(x_4)tal que o erro seja o menor possível para aproximar a integral de f no intervalo 0 a 2. Forneça como resposta A_2.**

MESMO CÓDIGO DA 6 MAS NO FINAL:

disp(pesos([0,3/5,5/5,2],0,2)) *//pegar o valor pedido na questão.*

**Q9 -Sejam os nós x=[0, 6/5, 8/5, 2]. Encontre os pesos A_i da quadratura I=A_1f(x_1)+...+A_4f(x_4)tal que o erro seja o menor possível para aproximar a integral de f no intervalo 0 a 2. Forneça como resposta A_4.**

MESMO CÓDIGO DA 6 MAS NO FINAL:

disp(pesos([0,6/5,8/5,2],0,2)) *//pegar o valor pedido na questão.*

**Q10 -Sejam os nós x=[0, 6/5, 8/5, 2]. Encontre os pesos A_i da quadratura I=A_1f(x_1)+...+A_4f(x_4)tal que o erro seja o menor possível para aproximar a integral de f no intervalo 0 a 2. Forneça como resposta \|w\|_2.**

MESMO CÓDIGO DA 9 MAS NO FINAL:

disp(norm(pesos([0,6/5,8/5,2],0,2))) *//pegar o valor pedido na questão.*