CURSO: ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

DISCIPLINA: MATEMÁTICA COMPUTACIONAL

PROJETO COMPUTACIONAL

PARTE I – GRUPO II

Diogo Martins Alves Nº 86980 (diogo.m.alves@tecnico.ulisboa.pt)

João Santiago Silva Nº 84081 (joao.santiago.s@tecnico.ulisboa.pt)

André Lopes Nº 84004 (andrefplopes@hotmail.com)

Data: 15/12/2017

1.-Resolução

Sendo $h = \frac{b-a}{m}$, e $x_i = a + i * h, \ \forall_{0 \le i \le m}$, pela regra dos trapézios composta temos que:

$$T_m(f) = \frac{h}{2}\left(f(x_0) + \sum_{i=1}^{m-1} 2 * f(x_i) + f(x_m) \right)$$

e pela regra de Simpson composta:

$$S_m(f) = \frac{h}{3}\left(f(x_0) + 4 * \sum_{i=1}^{\frac{m}{2}} f(x_{2i-1}) + 2 * \sum_{i=1}^{\frac{m}{2}-1} f(x_{2i}) + f(x_m) \right)$$

Aplicando a regra de Romberg:

$$R_{j,0} = T_{m(j)}(f), j = 0, \dots, n$$

$$R_{j,k} = \frac{4^k * R_{j,k-1} - R_{j-1,k-1}}{4^k - 1}, j = k, \dots, n$$

Com $k = 1$ obtemos:

$$R_{j,1} = \frac{4^1 * R_{j,0} - R_{j-1,0}}{4^1 - 1} = \frac{4 * T_{m(j)} - T_{m(j-1)}}{3} = \frac{4 * T_{2^j} - T_{2^{j-1}}}{3}$$

$$= \frac{4}{3} * \frac{h_1}{2}\left(f(x_0) + \sum_{i=1}^{2^j-1} 2 * f(x_i) + f\left(x_{2^j}\right) \right)$$

$$- \frac{1}{3}\frac{h_2}{2}\left(f(y_0) + \sum_{i=1}^{2^{j-1}-1} 2 * f(y_i) + f\left(y_{2^{j-1}}\right) \right)$$

Onde $h_1 = \frac{b-a}{2^j}$; $x_i = a + i * h_1 \ \forall_{0 \le i \le 2^j}$; $h_2 = \frac{b-a}{2^{j-1}}$ e $y_i = a + i * h_2 \ \forall_{0 \le i \le 2^{j-1}}$, de onde se conclui que $h_2 = 2 * h_1$ e $y_i = x_{2i} \ \forall_{0 \le i \le 2^{j-1}}$, em que i é um número natural.

Assim fica:

$$R_{j,1} = \frac{4}{3} * \frac{h_1}{2}\left(f(x_0) + \sum_{i=1}^{2^j-1} 2 * f(x_i) + f(x_{2^j}) \right)$$

$$- \frac{1}{3} h_1 \left(f(x_0) + \sum_{i=1}^{2^{j-1}-1} 2 * f(x_{2i}) + f(x_{2^j}) \right)$$

$$= \frac{h_1}{3}\left(f(x_0) + 4 * f(x_1) + 2 * f(x_2) + \cdots + 2 * f(x_{2^j-2}) + 4 \right.$$

$$\left. * f(x_{2^j-1}) + f(x_{2^j}) \right)$$

$$= \frac{h_1}{3}\left(f(x_0) + 4 * \sum_{i=1}^{2^{j-1}} f(x_{2i-1}) + 2 * \sum_{i=1}^{2^{j-1}-1} f(x_{2i}) + f(x_m) \right)$$

$$= S_{2^j}(f)$$

Como queríamos mostrar.

2. – Resolução

Código do programa feito em Matlab para calcular um integral através da regra dos Trapézios:

```
display('Insira os dados de entrada (função f:[a,b]->IR e um natural n):');
str = (input('Função f(x) = ', 's'));%função a estudar (string) ex: x.^-2
f = inline(str, 'x');%converte a string para uma função do matlab
a = input('Extremo inferior do intervalo a = ');
b = input('Extremo superior do intervalo b = ');
n = input('Número natural n = ');

h = (b-a)/n;
sum = f(a) + f(b);
for i = 1:n-1
    sum = sum + 2*f(a + i*h);
end
integral = sum *(h/2); %regra dos trapézios composta
fprintf('Integral = %.20f\n', integral);
```

3. – Resolução

Código do programa feito em Matlab para calcular um integral através da regra de Romberg:

```
display('Insira os dados de entrada (função f:[a,b]->IR, um natural n e uma estimativa
para o erro):');
str = (input('Função f(x) = ', 's'));%função a estudar (string) ex: x.^-2
f = inline(str, 'x');%converte a string para uma função do matlab
a = input('Extremo inferior do intervalo a = ');
b = input('Extremo superior do intervalo b = ');
n = input('Número natural n = ');
epsilon = input('Tolerância de erro = ');

k = 1;
while( abs(Romberg( k , k , f , a , b )-Romberg( k-1 , k-1 , f , a , b )) >= epsilon )
%enquanto a condição não for verificada, incrementar k
    k = k+1;
end

integral = Romberg( log2(n) , k , f , a , b ); %regra de Romberg
fprintf('Integral = %.20f\n', integral);
```

Função que calcula os valores da função de Romberg:

```
function r = Romberg( j , k , f , a , b )

    if k == 0 %condição de paragem
        m = 2.^j;
        h = (b-a)/m;
        sum = f(a) + f(b);
        for i = 1:m-1
            sum = sum + 2*f(a + i*h);
        end
        r = sum *(h/2); %regra dos trapézios composta
    else
        r = (4.^k * Romberg( j , k-1 , f , a , b ) - Romberg( j-1 , k-1 , f , a , b ))
/(4.^k - 1); %chamadas recursivas à função
    end


end
```

4(a). - Resolução

Sendo $I = \int_1^2 f(x)\,dx$ em que $f(x) = \frac{1}{x^2}$

Resolvendo o integral fica $\int_1^2 \frac{1}{x^2} dx = \left(-\frac{1}{x}\right)_1^2 = -\frac{1}{2} + 1 = \frac{1}{2} = 0{,}5$

Aplicando os programas anteriores obtemos os seguintes resultados:

	Regra dos Trapézios		Regra de Romberg (k = 1)	
Nós de integração (n)	$T_n(f)$	$\lvert E_n^T \rvert$	$R_{log_2(n),1}(f)$	$\lvert E_n^R \rvert$
1	**0,**625000000000000	0,208333333333334	**0,**416666666666666	0,083333333333334
2	**0,5**34722222222222	0,030092592592593	**0,50**4629629629629	0,004629629629629
4	**0,5**08993764172335	0,008576152683295	**0,500**417611489040	0,000417611489040
8	**0,50**2270850326336	0,002240971282000	**0,5000**29879044336	0,000029879044336
16	**0,50**0569170126996	0,000567226733113	**0,50000**1943393883	0,000001943393883
32	**0,500**142384590821	0,000142261845392	**0,500000**122745429	0,000000122745429
64	**0,5000**35601916756	0,000035594224689	**0,5000000**07692067	0,000000007692067
128	**0,5000**08900839995	0,000008900358920	**0,500000000**481075	0,000000000481075
256	**0,50000**2225232553	0,000002225202481	**0,500000000**30072	0,000000000030072
512	**0,50000**0556309547	0,000000556307668	**0,5000000000**1879	0,000000000001879
1024	**0,500000**139077475	0,000000139077358	**0,5000000000**117	0,000000000000117

(A negrito encontram-se os algarismos significativos das aproximações obtidas.)

4.(b). – Resolução

Como $f \in C^2([1,2])$, a seguinte fórmula de erro relativa ao método dos trapézios é aplicável:

$$E_m^T(f) = -\frac{b-a}{2} * h^2 * f''(\xi)$$

Se $m = m(j) = 2^j$, então $h = h(j) = \frac{b-a}{2^j}$, de onde resulta que $\frac{h(j)}{h(j+1)} = 2$

Fazendo o quociente

$$\frac{E_{2j}^T(f)}{E_{2j+1}^T(f)} = \frac{-\frac{b-a}{2} * h(j)^2 * f''(\xi_1)}{-\frac{b-a}{2} * h(j+1)^2 * f''(\xi_2)} = 4 * \frac{f''(\xi_1)}{f''(\xi_2)}$$

Se j for suficiente grande, $f''(\xi_1) \approx f''(\xi_2)$, então $E_{2^j}^T(f) \approx 4 * E_{2^{j+1}}^T(f)$.

Como $R_{j,0} = T_{2^j}(f)$, então fazendo a regressão linear $f(j) = k * \varphi(j)$, em que $f(j) = |E_{2^j}^T|$ e $g(j) = |E_{2^{j+1}}^T|$, se obtermos um valor próximo de 4 para k, significa que os valores de $R_{j,0}$ confirmam a ordem de precisão da regra dos trapézios. Para isto, utilizaremos apenas os valores de $j \geq 5$, para que a aproximação $f''(\xi_1) \approx f''(\xi_2)$ faça sentido:

$\vec{f} = (0{,}002240971282000;\ 0{,}000567226733113;\ 0{,}000142261845392;\ 0{,}000035594224689;\ 0{,}000008900358920;\ 0{,}000002225202481;0,\ 000000556307668)$

$\vec{\varphi} = (0{,}000567226733113;\ 0{,}000142261845392;\ 0{,}000035594224689;\ 0{,}000008900358920;\ 0{,}000002225202481;\ 0,\ 000000556307668;\ 0{,}000000139077358)$

$$< \vec{\varphi},\ \vec{\varphi} > * k = < \vec{f},\ \vec{\varphi} > \Leftrightarrow\ 3{,}43336 * 10^{-7} * k =\ 0{,}0000013572351629$$
$$\Leftrightarrow k\ \approx 3{,}95$$

Conclui-se que os valores de $R_{j,0}$ confirmam a ordem de precisão da regra dos trapézios.

Analogamente para $R_{j,1}$ com a regra de Simpson:

Como $f \in C^4([1,2])$, a seguinte fórmula de erro relativa ao método de Simpson é aplicável:

$$E_m^S(f) = -\frac{b-a}{180} * h^4 * f^{(4)}(\xi)$$

Se $m = m(j) = 2^j$, então $h = h(j) = \frac{b-a}{2^j}$, de onde resulta que $\frac{h(j)}{h(j+1)} = 2$

Fazendo o quociente

$$\frac{E_{2^j}^S(f)}{E_{2^{j+1}}^S(f)} = \frac{-\frac{b-a}{180} * h(j)^4 * f^{(4)}(\xi_1)}{-\frac{b-a}{180} * h(j+1)^4 * f^{(4)}(\xi_2)} = 16 * \frac{f^{(4)}(\xi_1)}{f^{(4)}(\xi_2)}$$

Se j for suficiente grande, $f^{(4)}(\xi_1) \approx f^{(4)}(\xi_2)$, então $E_{2j}^{S}(f) \approx 16 * E_{2j+1}^{S}(f)$.

Como $R_{j,1} = S_{2^j}(f)$, então fazendo a regressão linear $f(j) = k * \varphi(j)$, em que $f(j) = |E_{2j}^{S}|$ e $g(j) = |E_{2j+1}^{S}|$, se obtermos um valor próximo de 16 para k, significa que os valores de $R_{j,1}$ confirmam a ordem de precisão da regra de Simpson. Para isto, utilizaremos apenas os valores de $j \geq 5$, para que a aproximação $f''(\xi_1) \approx f''(\xi_2)$ faça sentido:

$\vec{f} =$ (0,000029879044336; 0,000001943393883; 0,000000122745429; 0,000000007692067; 0,000000000481075; 0,000000000030072; 0,000000000001879)

$\vec{\varphi} =$ (0,000001943393883; 0,000000122745429; 0,000000007692067; 0,000000000481075; 0,000000000030072; 0,000000000001879; 0,000000000000117)

$$< \vec{\varphi}, \quad \vec{\varphi} > * k = < \vec{f}, \quad \vec{\varphi} > \Leftrightarrow 0{,}000000000003791906 * k = 0{,}5{,}83062 * 10^{-11} \Leftrightarrow k \approx 15{,}38$$

Conclui-se que os valores de $R_{j,1}$ confirmam a ordem de precisão da regra de Simpson.

5.(a). – Resolução

Código da função em Matlab $J(m,x)$ que calcula os valores da função de Bessel usando o método de Romberg:

```
function b= J(m,x)
    %Função de Bessel
    g = @(t) cos(x*sin(t)-m*t); %função a integrar
    b=(1/pi)*Romberg(10,1,g,0,pi); %aproximação do integral pelo método de Simpson↙
(k=1) com 2^10 = 1024 sub-intervalos
end
```

Programa auxiliar desenvolvido para traçar os gráficos das funções $J_0, J_1 e\, J_2$:

```
x_max = 20; %valor máximo do eixo das abcissas do gráfico da função
step = 0.01; %distância entre os pontos onde irá ser calculado o valor da função
x = zeros(x_max/step,1); %inicialização do vetor de abcissas
y = zeros(x_max/step,1); %inicialização do vetor de ordenadas

for m=0:2 %3 gráficos: J_0, J_1, J_2

    x_i = 0;
    i = 1; %variável de iteração
    while(x_i <= x_max)
        x(i) = x_i; %construção do vetor de abcissas
        y(i) = J(m,x_i); %construção do vetor de oredenadas
        x_i = x_i + step;
        i = i+1;
    end
    figure
    plot(x,y) %desenho do gráfico J_m(x)
end
```

Gráficos obtidos:

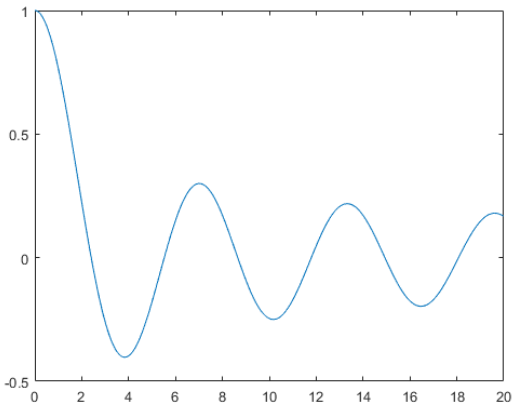

Gráfico de $J_0(x)$

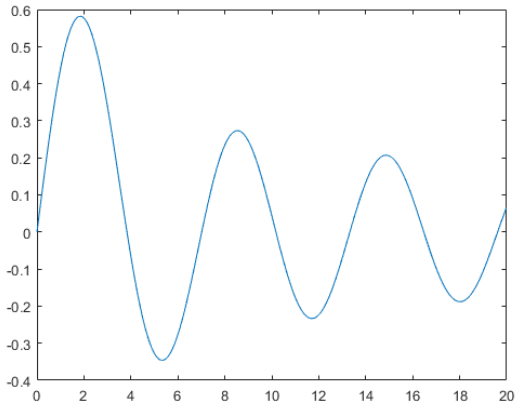

Gráfico de $J_1(x)$

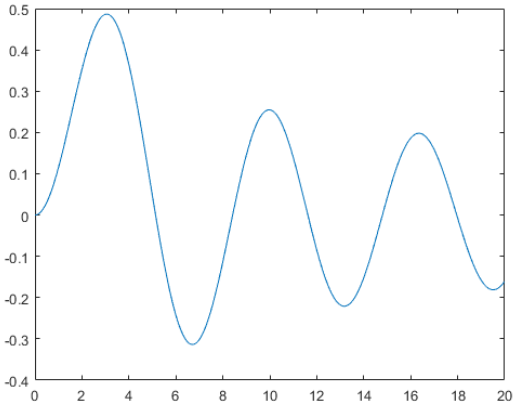

Gráfico de $J_2(x)$

5.(b). – Resolução

Código do programa feito em Matlab para produzir o "density plot":

```
%A imagem é constituida por quatro quadrantes, todos eles simétricos entre si, pelo↙
que só é preciso calcular valores para um
r_max = 10E-6; %distância do centro da imagem até às bordas
lambda = 500E-9; %comprimento de onda
step = 200; %número de pixels em horizontal/verticalmente num quadrante
i_m = 0.0000015; %variável para a função de escala
i_M = 0.1; %variável para a função de escala
K = 2*pi/lambda;
l = @(r)(J(1, K * r)/(K*r)).^2; %função de difração da luz
y = zeros(2*step, 2*step, 3); %inicialização da matriz rgb

for i=0:step
    for j=0:step
        %quandrante superior esquerdo
        r = sqrt((r_max*(step-i)/step).^2 + (r_max*(step-j)/step).^2); %distância ao↙
centro da imagem
        y(i+1,j+1,1) = log(l(r)/i_m)/log(i_M/i_m); %função de escala
        y(i+1,j+1,2) =  y(i+1,j+1,1); %green = red
        y(i+1,j+1,3) =  y(i+1,j+1,1); %blue = red
        %quadrante superior direito
        y(2*step - (i+1), j+1,1) = y(i+1,j+1,1);
        y(2*step - (i+1), j+1,2) = y(i+1,j+1,1);
        y(2*step - (i+1), j+1,3) = y(i+1,j+1,1);
        %quadrante inferior esquerdo
        y((i+1),2*step -  (j+1),1) = y(i+1,j+1,1);
        y((i+1),2*step -  (j+1),2) = y(i+1,j+1,1);
        y((i+1),2*step -  (j+1),3) = y(i+1,j+1,1);
        %quadrante inferior direito
        y(2*step - (i+1),2*step -  (j+1),1) = y(i+1,j+1,1);
        y(2*step - (i+1),2*step -  (j+1),2) = y(i+1,j+1,1);
        y(2*step - (i+1),2*step -  (j+1),3) = y(i+1,j+1,1);
    end
end
figure
imshow(y); %plot da imagem
```

Imagem obtida: