INSTITUTO SUPERIOR TÉCNICO

LICENCIATURA EM ENGENHARIA AEROESPACIAL

MATEMÁTICA COMPUTACIONAL

2º Projeto Computacional

Alunos:
Leonor Alves - 102845
Lourenço Faria - 103354
Paulo Campos - 103042
Pedro Almeida - 103027

Professores responsáveis:
Prof.ª Isabel dos Santos
Prof.ª Margarida Baía
Prof. Pedro Lima
Prof. Rúben Silva
Prof.ª Sónia Allaei

Ano Letivo 2022/2023

Índice

1

a) Método dos coeficientes indeterminados e grau de $Q(f)$

De forma a aproximar o seguinte integral $I\left(f\right) = \int_{-1}^{1} f\left(x\right)\ dx$ iremos recorrer a uma quadratura do tipo:

$$Q\left(f\right) = A_1 f\left(-\sqrt{\frac{3}{5}}\right) + A_2 f\left(0\right) + A_3 f\left(\sqrt{\frac{3}{5}}\right)$$

Esta quadratura tem grau k se:

$$\begin{cases} Q\left(x^i\right) = I\left(x^i\right), \quad i = 0, ..., k \\ Q\left(x^{k+1}\right) \neq I\left(x^{k+1}\right) \end{cases} \tag{1}$$

Assim, de forma a que a quadratura tenha, pelo menos, grau 2 é necessário que as seguintes condições se verifiquem:

$$\begin{cases} Q\left(1\right) = I\left(1\right) \\ Q\left(x\right) = I\left(x\right) \\ Q\left(x^2\right) = I\left(x^2\right) \end{cases}$$

Deste modo, surge:

$$\begin{cases} A_1 + A_2 + A_3 = \int_{-1}^{1} 1\ dx \\ \left(-\sqrt{\frac{3}{5}}\right) \cdot A_1 + 0 \cdot A_2 + \left(\sqrt{\frac{3}{5}}\right) \cdot A_3 = \int_{-1}^{1} x\ dx \quad \Leftrightarrow \\ \left(-\sqrt{\frac{3}{5}}\right)^2 \cdot A_1 + 0 \cdot A_2 + \left(\sqrt{\frac{3}{5}}\right)^2 \cdot A_3 = \int_{-1}^{1} x^2\ dx \end{cases}$$

$$\Leftrightarrow \begin{cases} A_1 + A_2 + A_3 = 2 \\ \left(-\sqrt{\frac{3}{5}}\right) \cdot A_1 + \left(\sqrt{\frac{3}{5}}\right) \cdot A_3 = 0 \\ \left(\frac{3}{5}\right) \cdot A_1 + \left(\frac{3}{5}\right) \cdot A_3 = \frac{2}{3} \end{cases} \Leftrightarrow \quad \begin{cases} A_1 = A_3 \\ A_1 \cdot 2 + A_2 = 2 \\ \left(\frac{6}{5}\right) \cdot A_3 = \frac{2}{3} \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} A_1 = \frac{5}{9} \\ A_2 = \frac{8}{9} \\ A_3 = \frac{5}{9} \end{cases}$$

Assim, $Q\left(f\right) = \frac{5}{9} \cdot f\left(-\sqrt{\frac{3}{5}}\right) + \frac{8}{9} \cdot f\left(0\right) + \frac{5}{9} \cdot f\left(\sqrt{\frac{3}{5}}\right)$.

De seguida, para encontrarmos o grau da quadratura, temos de utilizar a equação 1. Sabemos que $i = 1$ e $i = 2$ verificam a equação, pelo que:

Para $i = 3$:

$$Q\left(x^3\right) = \frac{5}{9} \cdot \left(-\sqrt{\frac{3}{5}}\right)^3 + \frac{8}{9} \cdot 0^3 + \frac{5}{9} \cdot \left(\sqrt{\frac{3}{5}}\right)^3 = 0$$

$$I\left(x^3\right) = \int_{-1}^{1} x^3\ dx = \left.\frac{x^4}{4}\right|_{-1}^{1} = 0$$

$$Q\left(x^3\right) = I\left(x^3\right)$$

Para $i = 4$:

$$Q\left(x^4\right) = \frac{5}{9} \cdot \left(-\sqrt{\frac{3}{5}}\right)^4 + \frac{8}{9} \cdot 0^4 + \frac{5}{9} \cdot \left(\sqrt{\frac{3}{5}}\right)^4 = \frac{2}{5}$$

$$I\left(x^4\right) = \int_{-1}^{1} x^4 \; dx = \left.\frac{x^5}{5}\right|_{-1}^{1} = \frac{2}{5}$$

$$Q\left(x^4\right) = I\left(x^4\right)$$

Para $i = 5$:

$$Q\left(x^5\right) = \frac{5}{9} \cdot \left(-\sqrt{\frac{3}{5}}\right)^5 + \frac{8}{9} \cdot 0^5 + \frac{5}{9} \cdot \left(\sqrt{\frac{3}{5}}\right)^5 = 0$$

$$I\left(x^5\right) = \int_{-1}^{1} x^5 \; dx = \left.\frac{x^6}{6}\right|_{-1}^{1} = 0$$

$$Q\left(x^5\right) = I\left(x^5\right)$$

Para $i = 6$:

$$Q\left(x^6\right) = \frac{5}{9} \cdot \left(-\sqrt{\frac{3}{5}}\right)^6 + \frac{8}{9} \cdot 0^6 + \frac{5}{9} \cdot \left(\sqrt{\frac{3}{5}}\right)^6 = \frac{6}{25}$$

$$I\left(x^6\right) = \int_{-1}^{1} x^6 \; dx = \left.\frac{x^7}{7}\right|_{-1}^{1} = \frac{2}{7}$$

$$Q\left(x^6\right) \neq I\left(x^6\right)$$

Logo, a quadratura tem grau $k = 5$.

b) Cálculo da quadratura para qualquer $f(x)$

A função *composite_gauss(f, n, a, b)* calcula o integral definido de uma função f num intervalo $[a, b]$ utilizando a fórmula de quadratura de Gauss composta. A função define os pesos A_1, A_2, A_3 e $h = \frac{b-a}{n}$, e, usando esses valores, calcula as variáveis x_1, x_2, x_3. É importante ter em conta que esta implementação assume que a função f é suficientemente regular de forma a que fórmula de quadratura de Gauss composta seja precisa.

Argumentos:

- f - função de uma variável
- n - número de espaçamentos
- a - início do intervalo
- b - fim do intervalo

A função devolve o valor aproximado da integração de f no intervalo $[a, b]$.

```
% Define the composite Gauss quadrature function
function Qn = composite_gauss(f, n, a, b)
    % Define the weights A1, A2, and A3
    A1 = 5/9;
    A2 = 8/9;
    A3 = 5/9;
    % Define the step size h
    h = (b-a)/n;
    x = a:h:b-h;
    % Define the nodes x1, x2, and x3
    x1 = x + h*(1-sqrt(3/5))/2;
    x2 = x + h/2;
    x3 = x + h*(1+sqrt(3/5))/2;
    % Calculate Qn using the composite Gauss quadrature formula
    Qn = (h/2)*(A1*sum(arrayfun(f,x1)) + A2*sum(arrayfun(f,x2)) + A3*sum(arrayfun(f,x3)));
end
```

2

a) Cálculo de $L(T)$ recorrendo a $Q_n(f)$

Este programa utiliza o método de quadratura de Gauss composta para calcular o espaço percorrido por um ponto em movimento num plano. A lei de movimento do ponto é definida pelas funções $x(t)$ e $y(t)$, que determinam a sua posição num dado momento. O integral $L(T) = \int_0^T \sqrt{x'(t)^2 + y'(t)^2}\, dt$ é utilizado para calcular o espaço percorrido.

O programa usa um loop *for* para iterar sobre um intervalo de valores de T (de 0 a 15 com um incremento de 0.5) e um loop *while* para iterar até que o erro seja menor que 10^{-6}. Deste modo, obtemos uma matriz com o valor de T, o valor calculado de $L(T)$ e o número de iterações utilizadas n.

```
% Define the derivative dx/dt
function out_dx = dx(t)
    out_dx = 1;
end
```

```
% Define the derivative dy/dt
function out_dy = dy(t)
    out_dy = (4 - t)/(4*exp(t/4));
end
```

```
% Define the vector of T values
T = 0:0.5:15;

% Define the function handle dL
dL = @(t) sqrt(dx(t)^2+dy(t)^2);

% Loop over all T values
for i=1:size(T,2)
    % Initialize n and L
    n = 1;
    L = composite_gauss(dL, n, 0, T(i));
    % Iterate until the error is less than 10^(-6)
    while abs(L - composite_gauss(dL, n*2, 0, T(i))) > 1e-6
        n = n+1;
        L = composite_gauss(dL, n, 0, T(i));
    end
    % Save the results in the res_a matrix
    res_a(i,1) = T(i);
    res_a(i,2) = L;
    res_a(i,3) = n;
end
```

T	$L(T)$	n	T	$L(T)$	n
0.0	0.0	1	8.0	8.40954579963394	4
0.5	0.667313617022792	1	8.5	8.91408174306375	4
1.0	1.27100820668118	1	9.0	9.41849636038670	4
1.5	1.83133586355048	1	9.5	9.92269857651331	4
2.0	2.36374844388501	1	10.0	10.4266267030801	4
2.5	2.87935915785145	2	10.5	10.9302409169718	5
3.0	3.38569024113166	2	11.0	11.4335268872697	5
3.5	3.88753390672483	2	11.5	11.9364807339842	5
4.0	4.38774659300006	2	12.0	12.4391108678484	5
4.5	4.88789303496646	2	12.5	12.9414333865288	5
5.0	5.38871934879341	3	13.0	13.4434670137774	6
5.5	5.89048339120279	3	13.5	13.9452388104207	6
6.0	6.39316891232765	3	14.0	14.4467725258221	6
6.5	6.89662360833617	3	14.5	14.9480932162592	6
7.0	7.40064214294924	3	15.0	15.4492251373774	6
7.5	7.90501406333529	3			

Tabela 1: Cálculo de $L(T)$, com intervalos de 0.5 entre os sucessivos valores de $T \in [0, 15]$

b) Utilização de $Q_n(f)$ para aproximar o valor de $L(15)$

```
% Define the vector of n values
n = [10, 20, 40, 80, 160];

% Define the function handle dL
dL = @(t) sqrt(dx(t)^2+dy(t)^2);

% Loop over all n values
for i=1:size(n, 2)
    % Calculate the composite Gauss quadrature for the current value of n
    res_b(i, 1) = composite_gauss(dL, n(i), 0, 15);
end
```

n	$L(15)$
10	15.4492241656482
20	15.4492242251562
40	15.4492242257210
80	15.4492242257292
160	15.4492242257293

Tabela 2: Cálculo de valores aproximados de $L(15)$, utilizando a fórmula $Q_n\left(f\right)$

c) Cálculo da relação entre quadraturas com diferentes valores de n

```
% Define the vector of n values
n = [10, 20, 40, 80, 160];

% Define the function handle dL
dL = @(t) sqrt(dx(t)^2+dy(t)^2);

% Loop over all n values
for i=1:size(n, 2)
    % Calculate L using composite_gauss method
    res_c(i,1) = n(i);
    res_c(i,2) = composite_gauss(dL, n(i), 0, 15);
    % Calculate the absolute error of L
    res_c(i,3) = abs(composite_gauss(dL, 2*n(i), 0, 15) - composite_gauss(dL, n(i), 0, 15));
    % Calculate the relative error of L
    res_c(i,4) = abs(composite_gauss(dL, 2*n(i), 0, 15) - composite_gauss(dL, n(i), 0, 15))/abs(composite_gauss(dL, n(i), 0, 15) - composite_gauss(dL, n(i)*0.5, 0, 15));
end
```

n	Q_n	$Q_{2n} - Q_n$	$(Q_{2n} - Q_n) \,/\, (Q_n - Q_{n/2})$
10	15.4492241656482	5.95079772125473e-08	0.00441555110965764
20	15.4492242251562	5.64833513294616e-10	0.00949172765992658
40	15.4492242257210	8.17479417491995e-12	0.0144729269466275
80	15.4492242257292	1.22568621918617e-13	0.0149934810951760
160	15.4492242257293	5.32907051820075e-15	0.0434782608695652

Tabela 3: Relação entre quadraturas com valores de n diferentes

d) Erro da fórmula de quadratura

Uma vez que a fórmula de quadratura satisfaz a desigualdade $|E_n(f)| \leq Ch^{\alpha}$, onde C e α são constantes independentes de h, é possível determinar o valor de α através da expressão seguinte:

$$\frac{E_{2n}}{E_n} = \frac{C\left(\frac{1}{2}h\right)^{\alpha}}{Ch^{\alpha}} = \left(\frac{1}{2}\right)^{\alpha}$$

$$\Rightarrow \alpha = \log_{\frac{1}{2}} \frac{E_{2n}}{E_n} \approx \log_{\frac{1}{2}} \frac{Q_{2n} - Q_n}{Q_n - Q_{\frac{n}{2}}}$$

n	α
10	7.82319077123899
20	6.71911357786284
40	6.11049947354672
80	6.05952081144592
160	4.52356195605701

Tabela 4: Valores de α para diferentes valores de n

Contudo, é de notar que os valores obtidos para o α na tabela 4 não são constantes à medida que o n aumenta, ao contrário do valor esperado, $\alpha = 6 = \text{grau } +1 = 5 + 1 = 6$.

A partir da análise do resultado obtido é possível comparar a eficiência da quadratura utilizada com a de outras regras de integração numéricas, como, por exemplo, o método do ponto médio, a fórmula dos trapézios e a fórmula de Simpson.

$$\left|E^{PM}(f)\right| \leq \frac{h^3}{24} \max_{x\in[a,b]} \left|f''(x)\right|$$

$$\left|E^{T}(f)\right| \leq \frac{h^3}{12} \max_{x\in[a,b]} \left|f''(x)\right|$$

$$\left|E^{S}(f)\right| \leq \frac{h^5}{90} \max_{x\in[a,b]} \left|f^{(4)}(x)\right|$$

Ao analisar as expressões dos erros, é possível retirar que são proporcionais a h^3 (para a fórmula do ponto médio e dos trapézios) e h^5 (para a fórmula de Simpson). Comparando estes valores com o obtido através da quadratura utilizada, e tendo em conta que quanto maior o α, maior a eficiência da quadratura, verifica-se que o método utilizado é o mais eficiente.

É importante notar que o programa aplica a transformação logarítmica do erro relativo para analisar a eficiência do método de quadratura de Gauss composta para diferentes valores de n, o que permite uma comparação mais precisa da eficiência do método.

É necessário ter em conta que o seguinte *script* depende das variáveis saída do *script* correspondente à alínea c). Logo, este último deve ser executado previamente ao da alínea d).

```
% Define the symbolic variable x
syms x;

% Define the vector of n values
n = [10, 20, 40, 80, 160];

% Define the function handle dL
dL = @(t) sqrt(dx(t)^2+dy(t)^2);

% Initialize the vector res_d
res_d = zeros(size(n,2),1);

% Define the function handle new_log
new_log(x) = log(x) / log(0.5);

% Loop over all n values
for i=1:size(n,2)
    % Calculate the logarithmic transformation of the relative error
    res_d(i,1) = new_log(res_c(i, 4));
end
```

e) Método dos mínimos quadrados e regressão linear

Esta função *sse(g, T, res)* - sum of squared errors - foi projetada para aplicar o método dos mínimos quadrados, de acordo com as funções de entrada (input), a um conjunto de pontos dados. As funções de entrada são passadas através de um vetor que será percorrido de forma a avaliá-las em cada ponto do domínio, armazenando o resultado numa matriz. De seguida a função *sse(g, T, res)* calcula uma matriz com os produtos internos das funções do input (ϕ), bem como um vetor com os produtos internos das imagens e das funções dadas (f). Finalmente, esta função resolve o sistema linear das matriz ϕ e f (ver figura 1), a partir do qual os coeficientes das funções do input são obtidos.

$$
\begin{array}{ccc}
\phi & & f \\
\begin{bmatrix}
\langle \phi_0, \phi_0 \rangle & \langle \phi_0, \phi_1 \rangle & \cdots & \langle \phi_0, \phi_m \rangle \\
\langle \phi_1, \phi_0 \rangle & \langle \phi_1, \phi_1 \rangle & \cdots & \langle \phi_1, \phi_m \rangle \\
\vdots & \vdots & \ddots & \vdots \\
\langle \phi_m, \phi_0 \rangle & \langle \phi_m, \phi_1 \rangle & \cdots & \langle \phi_m, \phi_m \rangle
\end{bmatrix}
\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix}
& = &
\begin{bmatrix} \langle \phi_0, f \rangle \\ \langle \phi_1, f \rangle \\ \vdots \\ \langle \phi_m, f \rangle \end{bmatrix}
\end{array}
$$

$$
\langle \phi_i, \phi_j \rangle = \sum_{k=0}^{n} \phi_i(x_k)\phi_j(x_k), \qquad \langle \phi_i, f \rangle = \sum_{k=0}^{n} \phi_i(x_k) f_k
$$

Figura 1: Sistema linear utilizado para o calculo dos coeficientes de ajuste a_0, a_1, ..., a_m com base no método dos mínimos quadrados

Argumentos:

- g - célula (em linha) com funções
- T - vetor (em linha) que representa o domínio a analisar das funções de g
- res - vetor (em linha) que representa as imagens da função a ser aproximada pelo método dos mínimos quadrados

O output da função *sse(g, T, res)* serão os coeficientes das funções de entrada, que poderão ser usados para ajustar uma função aos pontos dados.

```
function output = sse(g, T, res)
    % Loop through the input functions
    for i=1:length(g)
        func = g{i};
        % Evaluate the input functions at each T value
        for j=1:length(T)
            vec(i,j) = arrayfun(func,T(j));
        end
    end

    % Calculate the phi matrix
    for i=1:length(g)
        for j=1:length(g)
            sum = 0;
            for k=1:length(T)
                sum = sum + vec(i,k)*vec(j,k);
            end
            phi(i,j)=sum;
        end
    end

    % Calculate the f vector
    for i=1:length(g)
        sum = 0;
        for j=1:length(T)
            sum = sum + res(j)*vec(i,j);
        end
        f(i, 1) = sum;
    end

    % Solve the linear system of equations
    output = linsolve(phi,f);
end
```

Este último *script* usa o método dos mínimos quadrados (função *sse(g, T, res)*) para ajustar uma regressão linear aos dados do exercício em causa $(T, L(T))$.

É necessário ter em conta que o seguinte *script* depende das variáveis saída do *script* correspondente à alínea a). Logo, este último deve ser executado previamente ao da alínea e).

```
% Define the vector of T values
T = 0:0.5:15;

% Define the input vector for the sse function
input = {@(x) x, @(x) 1};

% Calculate the sum of squared errors for the input function and the T values
res_e = sse(input, T, res_a(:, 2));

% Calculate the fit of the input function using the polynomial val method
exec_fit = polyval(res_e, T);

% Create a new figure and plot the fit of the input function
plot(T, exec_fit, 'LineWidth', 1.5);
hold on;
grid on;

% Plot the T values and the corresponding L(T) values
plot(T, res_a(:,2), 'o');
legend('1.0143t + 0.0.2781');
xlabel('T');
ylabel('L(T)');
```

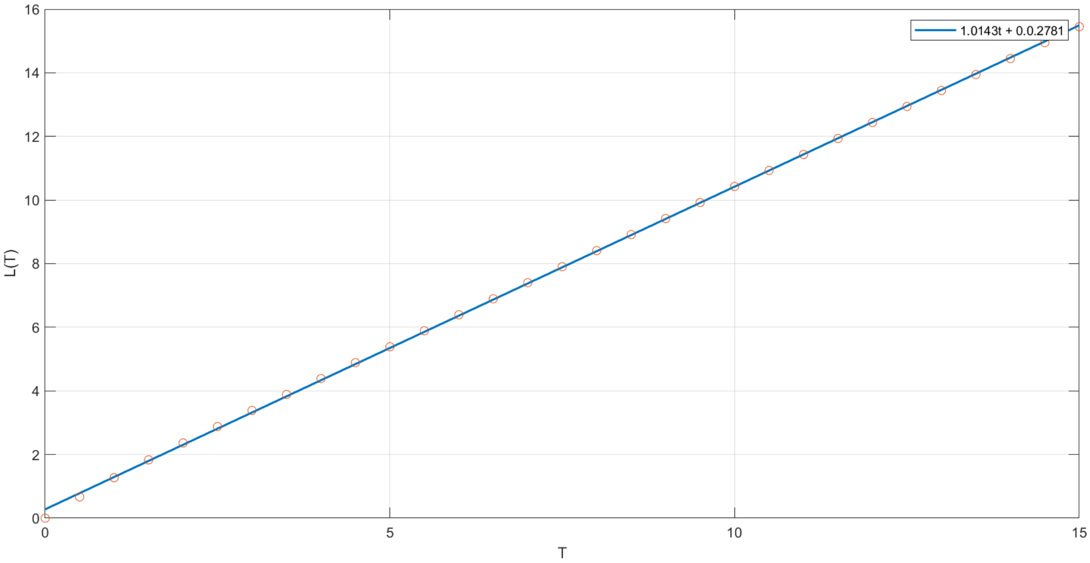

Figura 2: Método dos mínimos quadrados para aproximar a função $L(T)$ a uma função linear