Instituto Superior Técnico

Licenciatura em Engenharia Aeroespacial

Matemática Computacional

1º 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 Existência de zero no intervalo

Seja $f(t) = y_2(t) - y_1(t) = t^2 + sin(t) - a$, onde $y_1 = a - sin(t)$ e $y_2 = t^2$. Queremos verificar para que valores de a existe solução para a equação $f(t) = 0$, com $t \in [1; 2]$.

A função f é contínua, uma vez que é dada pela soma de funções contínuas. Logo, pelo Teorema de Bolzano, existe uma solução da equação se $f(1) \cdot f(2) \leq 0$, ou seja, se $f(1)$ e $f(2)$ tiverem sinais opostos.

$$f(1) = 1 + sin(1) - a$$
$$f(1) = 0 \Rightarrow a = 1 + sin(1)$$

$$f(2) = 4 + sin(2) - a$$
$$f(2) = 0 \Rightarrow a = 4 + sin(2)$$

- Caso 1: $f(1) \geq 0$ e $f(2) \leq 0$
  $a \leq 1 + sin(1)$ e $a \geq 4 + sin(2)$
  $a \in \varnothing$
- Caso 2: $f(1) \leq 0$ e $f(2) \geq 0$
  $a \geq 1 + sin(1)$ e $a \leq 4 + sin(2)$
  $a \in [1 + sin(1); 4 + sin(2)]$

Assim, $I_a = \varnothing \cup [1 + sin(1); 4 + sin(2)] = [1 + sin(1); 4 + sin(2)]$.

2 Convergência do método de Newton

De forma a podermos garantir a convergência do método de Newton para τ_a é necessário verificar as 4 condições suficientes de convergência.

A primeira condição foi verificada na pergunta anterior, na qual averiguamos o intervalo onde $f(1) \cdot f(2) \leq 0$ se verifica.

$$f(t) = t^2 - a + sin(t)$$

$$f'(t) = 2t + cos(t) > 0, \quad \forall t \in I = [1; 2]$$

Considerando a(t) = 2t e b(t) = cos(t) vem:

$$\begin{cases} t = 1 \begin{cases} a(1) = 2 \\ b(1) = cos(1) > 0 \end{cases} \\ \\ t = 2 \begin{cases} a(2) = 4 \\ b(2) = cos(2) < 0 \end{cases} \end{cases}$$

Uma vez que b(t) é contínua, a(t) é contínua e crescente, $-1 \leq cos(t) \leq 1$ e $2t \geq 2, \quad \forall t \in I = [1; 2]$, podemos concluir pelo teorema das funções enquadradas que:

$$-1 + 2 \leq cos(t) + 2t \leq 1 + 4 \ \Leftrightarrow 1 \leq cos(t) + 2t \leq 5 \ \Leftrightarrow 1 \leq f'(t) \leq 5$$

Ou seja, verificamos a segunda proposição:

$$f'(t) \neq 0, \quad \forall t \in I = [1;2]$$

A partir da expressão obtida anteriormente vem:

$$f''(t) = 2 - sin(t) \tag{1}$$

Uma vez que $-1 \leq sin(t) \leq 1$, então $f''(t) \geq 0, \quad \forall t \in I = [1;2]$, estando, assim, comprovada a terceira condição.

Por último, é necessário verificar:

$$\frac{|f(a)|}{|f'(a)|} < b - a \quad \wedge \quad \frac{|f(b)|}{|f'(b)|} < b - a$$

Para tal, é necessário encontrar o intervalo de valores de a no qual esta condição se verifica. Deste modo surge:

$$\frac{|f(1)|}{|f'(1)|} < 1 \Leftrightarrow \frac{|1 - a + sin(1)|}{|2 + cos(1)|} < 1 \Leftrightarrow |1 - a + sin(1)| < |2 + cos(1)| \Leftrightarrow$$
$$\Leftrightarrow 1 - a + sin(1) < 2 + cos(1) \quad \wedge \quad 1 - a + sin(1) > -2 - cos(1) \Leftrightarrow$$
$$\Leftrightarrow a > sin(1) - 1 - cos(1) \quad \wedge \quad a < 3 + cos(1) + sin(1)$$

$$\frac{|f(2)|}{|f'(2)|} < 1 \Leftrightarrow \frac{|4 - a + sin(2)|}{|4 + cos(2)|} < 1 \Leftrightarrow |4 - a + sin(2)| < |4 + cos(2)| \Leftrightarrow$$
$$\Leftrightarrow 4 - a + sin(2) < 4 + cos(2) \quad \wedge \quad 4 - a + sin(2) > -4 - cos(2) \Leftrightarrow$$
$$\Leftrightarrow a > sin(2) - cos(2) \quad \wedge \quad a < 8 + cos(2) + sin(2)$$

Assim, vem que

$$a \in A = [sin(2) - cos(2); 3 + cos(1) + sin(1)]$$

Uma vez que $J = A \cap I_a$ então,

$$J = [1 + sin(1); 3 + cos(1) + sin(1)]$$

3 Implementação do método de Newton

A função func_f é dada por $y_2(t) - y_1(t)$. A função devolve um valor para cada argumento a e t.

```
function y_1 = y_1(a, t)
    y_1 = a - sin(t);
end
```

Figura 1: Código MatLab da função y_1 (y_1.m)

```
function y_2 = y_2(t)
    y_2 = t.^2;
end
```

Figura 2: Código MatLab da função y_2 (y_2.m)

```
function f_t = func_f(a, t)
    f_t = y_2(t) - y_1(a, t);
end
```

Figura 3: Código MatLab da função $f(t)$ (func_f.m)

A função func_df é a derivada da função func_f. A função devolve um valor que depende apenas da variável t.

```
function df_t = func_df(t)
        df_t = 2*t + cos(t);
    end
```

Figura 4: Código MatLab da função $f'(t)$ (func_df.m)

A função func_X recebe o valor da iteração anterior do método de Newton e o valor de a, devolvendo a iteração seguinte.

```
function X_n1 = func_X(X_n,a)
        X_n1 = X_n - (func_f(a,X_n)/func_df(X_n));
    end
```

Figura 5: Código MatLab do método de Newton (func_X.m)

A função func_e verifica se para duas iterações consecutivas x_n e x_{n+1}, a desigualdade $|x_{n+1} - x_n| < \epsilon$ é verdadeira. Nesse caso, não serão feitas mais iterações.

```
function truth = func_e(t_n1, t_n,e)
        if abs(t_n1-t_n) < e
            truth = 1;
        else
            truth = 0;
        end
    end
```

Figura 6: Código MatLab da função de condição de paragem (func_e.m)

A função main recebe como argumentos o valor de a, a iterada inicial t_ini, o número máximo de iterações N e o parâmetro ϵ que define um critério de paragem do método logo que para duas iterações consecutivas x_n e x_{n+1}, a desigualdade $|x_{n+1} - x_n| < \epsilon$ é verdadeira. A função utiliza a instrução *while* que apenas termina quando o módulo da diferença entre duas iterações consecutivas for inferior a ϵ ou quando é atingido o número máximo de iterações. Em cada ciclo da instrução *while*, obtemos uma nova iteração através do método de Newton. Esse valor é acrescentado na posição n do vetor que a função devolve.

```
function vector = main(a,t_ini,N,e)

    %nargin

    truth = 0;
    X_n = t_ini;
    X_n1 = 0;
    n = 1;
    vector(n) = X_n;

    while((truth == 0) && (n <= N))
        X_n1 = func_X(X_n,a);
        truth = func_e(X_n1,X_n,e);
        X_n = X_n1;
        n = n + 1;
        vector(n) = X_n;
    end

end
```

Figura 7: Código MatLab do *loop* onde são convocadas as funções anteriores

4 Estimativa dos valores da ordem de convergência e do coeficiente assintótico

Executando o programa anterior com $a = 2$, $\epsilon < 10^{-10}$ e $t_0 = 1$ e com N um valor arbitrário (apenas para forçar ϵ como critério de paragem), obtemos a seguinte aproximação ao fim de 4 iteradas: $\tau_a = 1.061549774631384$.

	1	2	3	4	5
1	1	1.0624	1.0615	1.0615	1.0615

Figura 8: Tabela onde aparecem os valores das iteradas no MatLab - maior precisão no MatLab

```
values = main(2, 1, 1000, 10^(-10));
```

Figura 9: Linha de código que permite guardar o valor das iteradas e de τ_a, guardando-as em "values"

Sabendo que uma sucessão $X_n, n \in \mathbb{N}$ converge para z com ordem de convergência $p \geq 1$, se existir a constante positiva K_∞ (coeficiente assintótico) de tal modo que:

$$\lim_{n \to \infty} \frac{|z - x_{n+1}|}{|z - x_n|^p} = K_\infty \Leftrightarrow$$

$$\Leftrightarrow \lim_{n \to \infty} \frac{|e_{n+1}|}{|e_n|^p} = K_\infty \tag{2}$$

podemos utilizar a equação 2 para estimar p e K_∞, uma vez que e_n e e_{n+1} podem ser calculados com base nas iterações dadas pelo método.

Deste modo uma abordagem possível será realizar uma regressão linear baseada na seguinte relação:

$$\frac{|e_{n+1}|}{|e_n|^p} = K_\infty \Leftrightarrow |e_{n+1}| = |e_n|^p \, K_\infty \Leftrightarrow$$

$$\Leftrightarrow |e_{n+1}| = |e_n|^p \, K_\infty \Leftrightarrow$$
$$\Leftrightarrow \ln |e_{n+1}| = \ln \left(|e_n|^p \, K_\infty\right) \Leftrightarrow$$
$$\Leftrightarrow \ln |e_{n+1}| = \ln \left(|e_n|^p\right) + \ln \left(K_\infty\right) \Leftrightarrow$$
$$\Leftrightarrow \ln |e_{n+1}| = p \ln \left(|e_n|\right) + \ln \left(K_\infty\right)$$

Isto é:

$$y = px + \ln K_\infty$$

onde p corresponde ao declive (m) da reta e $\ln(K_\infty)$ à ordenada na origem (b). De forma, a obter os valores pretendidos foram desenvolvidas as seguintes funções em MatLab:

```
%Cálculo do valor de z através da função da alinea 3) e atribuição do valor à variavel "z"
z_vec = func(2, 1, 1000, 10^(-36));
z = z_vec(size(z_vec,2));

%Cálculo das iteradas e de tau_a
values = func(2, 1, 1000, 10^(-10));

%loop para o calculo do erro absoluto
for i=1:size(values, 2)-1
    dif_answer(i) = abs(z - values(i));
end

%loop para calcular o ln do erro
for i=1:size(dif_answer,2)-1
    vec_1(i) = log(dif_answer(i+1));
    vec_2(i) = log(dif_answer(i));
end

%Regressão linear
exec_pol = polyfit(vec_2, vec_1, 1);

exec_fit = polyval(exec_pol, vec_2);

%Calculo do coeficiente assintótico
K = exp(exec_pol(2));

%Gráfico corresponde à regressão linear
plot(vec_2, exec_fit);
hold on;
grid on;
plot(vec_2, vec_1, 'o');
```

Para os valores fornecidos no enunciado foi obtida a seguinte regressão linear:

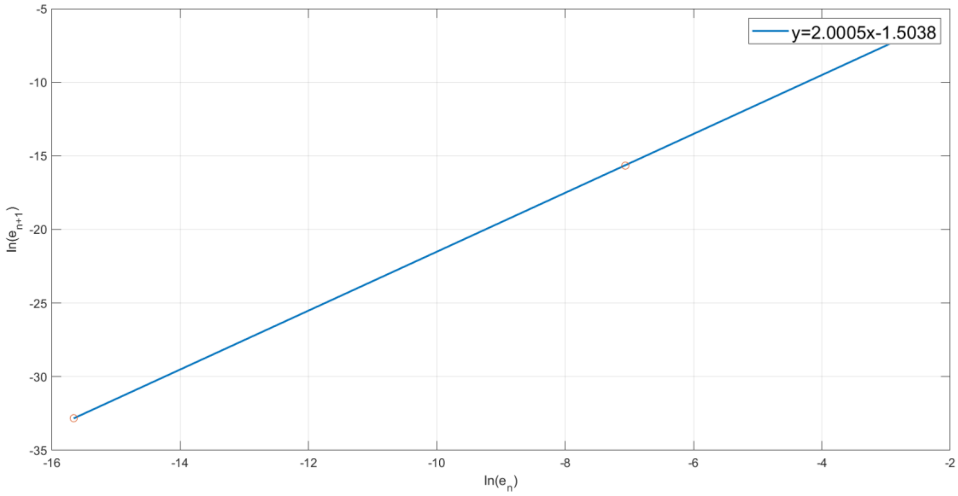

Figura 10: Gráfico da regressão linear obtida, $y = 2.0005x - 1.5038$.

Uma vez obtidos os valores do declive ($m = 2.0005$) e da ordenada da origem ($b = -1.5038$), podemos finalmente, estimar os valores de p e de K_∞:

- $p = m = 2.0005 \approx 2$, ordem de convergência 2.
- $\ln(K_\infty) = b \Leftrightarrow K_\infty = e^b = e^{-1.5038} \Leftrightarrow K_\infty \approx 0.222$.

5 Ordem de convergência esperada

Sabemos que, se $f \in C^2$ em I e $f'(z) \neq 0$, então o método de Newton, quando converge, tem ordem de convergência igual ou superior a 2 ($p \geq 2$). Como anteriormente (cf. 2) verificámos que as duas condições referidas se verificam e que o método converge, será de esperar que o método tenha convergência pelo menos quadrática. Além disso, tendo em conta a seguinte equação:

$$\lim_{n\to\infty} \frac{|z - x_{n+1}|}{|z - x_n|^p} = \frac{|f''(z)|}{2\,|f'(z)|}$$

também podemos verificar que para o limite ser > 0, ou seja $\neq 0$, será necessária $f''(z) \neq 0$. Tendo em conta a equação 1, e sabendo que $-1 \leq sin(t) \leq 1$, torna- se evidente que a condição $f''(z) \neq 0$ também se verifica no nosso caso. Assim será expectável que a ordem de convergência seja $p = 2$, o que corresponde ao valor obtido em 4.

Conhecendo o valor de p é ainda possível estimar K_∞ através de outro método. Na realidade, visto tratar-se de um limite, será de esperar que utilizando apenas o valor das ultimas iteradas para o calculo dos erros, obtenhamos uma boa aproximação do coeficiente assintótico com a seguinte expressão:

$$K_\infty = \frac{|e_{n+1}|}{|e_n|^p} \Leftrightarrow \frac{|5.5511e(-15)|}{|1.5797e(-7)|^2} \approx 0.222$$

Tal como esperado, K_∞ é uma constante positiva, o que corrobora $p = 2$.

6 Interseção das funções y_1 e y_2 com dependência de a

Queremos agora determinar os valores de τ_a para $a \in J$, isto é, $g(a) = \tau_a$, com um espaçamento nas abcissas de 0.01 e um erro $\epsilon < 10^{-5}$.

À medida que esta variável se altera, a função y_1 vai sofrer uma translação vertical linearmente dependente de a.

Ao nível computacional, optou-se por realizar um *loop* que vai utilizar a função desenvolvida na alínea 3) para calcular uma aproximação de τ_a para cada valor de a. Basta escolher um valor para $t_0 \in I$ e armazenar o último valor do vetor solução da função da alínea 3).

Para visualizarmos onde é que estas interseções ocorrem na função y_1, basta obter a imagem de τ_a, $y_1(\tau_a)$, e representar num gráfico.

Assim, fazendo o gráfico de $g(a) = \tau_a$ e de $h(a) = y_1(\tau_a)$ num mesmo *plot*, obtemos o seguinte resultado:

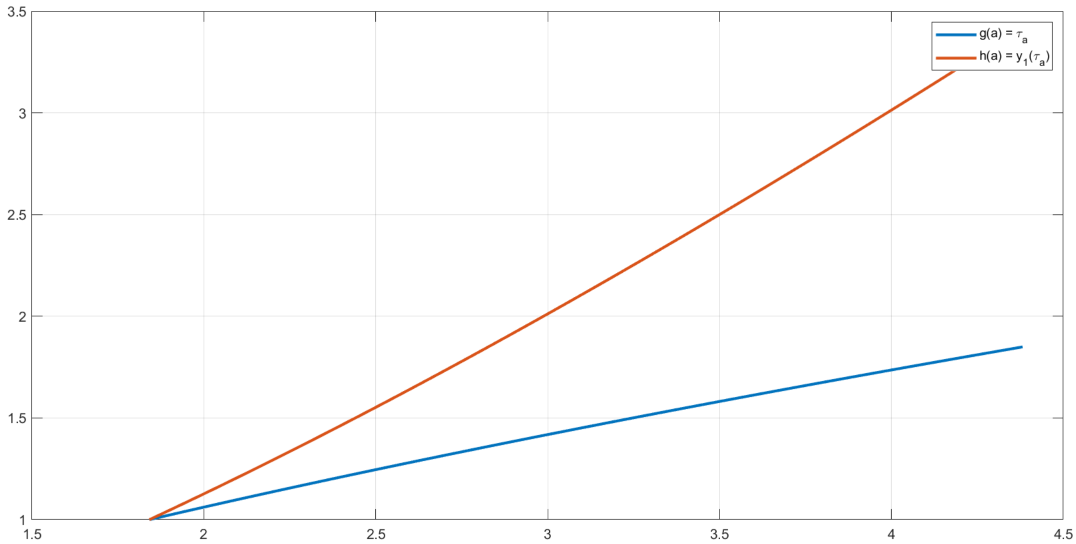

Figura 11: Gráficos de $g(a) = \tau_a$ e de $h(a) = y_1(\tau_a)$

```
a = 1+sin(1):0.01:3+sin(1)+cos(1);

for i=1:size(a, 2)
    vector = main(a(i), 1.5, 10, 10^(-5));
    res(i) = vector(size(vector, 2));
end

plot(a, res, 'LineWidth', 2);
hold on;
grid on;
plot(a, y_1(a, res), 'LineWidth', 2);
legend('g(a) = \tau_{a}', 'h(a) = y_1(\tau_a)');
```

Figura 12: Código MatLab para obtenção dos *plots* de $g(a) = \tau_a$ e de $h(a) = y_1(\tau_a)$ (inter_a.m)

7 Análise da função da alínea 3) para valores $t_0 \notin I$

Determinámos na alínea 2) um intervalo para os valores de a para os quais se pode garantir convergência pelo método de Newton para uma aproximação de τ_a, que corresponde à interseção entre as funções y_1 e y_2 no intervalo $I = [1, 2]$.

Se utilizarmos a função desenvolvida na alínea 3), com $a = 2$, $\epsilon < 10^{-5}$ e t_0 = -0.5, ou seja, com $t_0 \notin I$, obtemos alguns resultados interessantes:

$\epsilon < 10^{-5}$	
n	x_n
0	-0.500000000000000
1	-18.7116663526949
2	-9.15259657766104
3	-4.92280708232712
4	-2.51411282311604
5	-1.87454001750679
6	-1.73628589923619
7	-1.72849142384347
8	-1.72846631925768
9	-1.72846631899718

Tabela 1: Aproximação do valor da interseção das funções y_1 e y_2 para $t_0 = -0.5$

```
>> main(2, -0.5, 10, 10^(-5))

ans =

   -0.5000  -18.7117   -9.1526   -4.9228   -2.5141   -1.8745   -1.7363   -1.7285   -1.7285   -1.7285
```

Figura 13: Código MatLab para obtenção de uma raíz da função $f(x)$ com $t_0 = -0.5$

No entanto, como a função $f(t)$ não foi analisada analiticamente fora do intervalo I, não podemos garantir a partir deste resultado que vai existir sempre convergência.

Se $t_0 = -0.3$, o método de Newton volta a convergir para os valores obtidos na alínea 3):

$\epsilon < 10^{-5}$	
n	x_n
0	-0.300000000000000
1	5.90684977241873
2	3.35474045615803
3	1.77718099044896
4	1.13911884757442
5	1.06278033498414
6	1.06155010116542
7	1.06154977463141

Tabela 2: Aproximação do valor da interseção das funções y_1 e y_2 para $t_0 = -0.3$

```
>> main(2, -0.3, 10, 10^(-5))

ans =

   -0.3000    5.9068    3.3547    1.7772    1.1391    1.0628    1.0616    1.0615
```

Figura 14: Código MatLab para obtenção de uma raíz da função $f(t)$ com $t_0 = -0.3$

Se traçarmos um gráfico da função apresentada na alínea 1), $f(t) = y_2(t) - y_1(t)$ e para $a = 2$, é possível notar um mínimo relativo incluído no intervalo $]-0.5, -0.3[$. Caso o método se aplicasse exatamente nesse ponto onde a derivada de $g(t)$ fosse nula, o método de Newton não iria convergir para nenhum valor dado que a reta tangente ao gráfico não intersetaria o eixo das abcissas, uma vez que seria paralela ao mesmo.

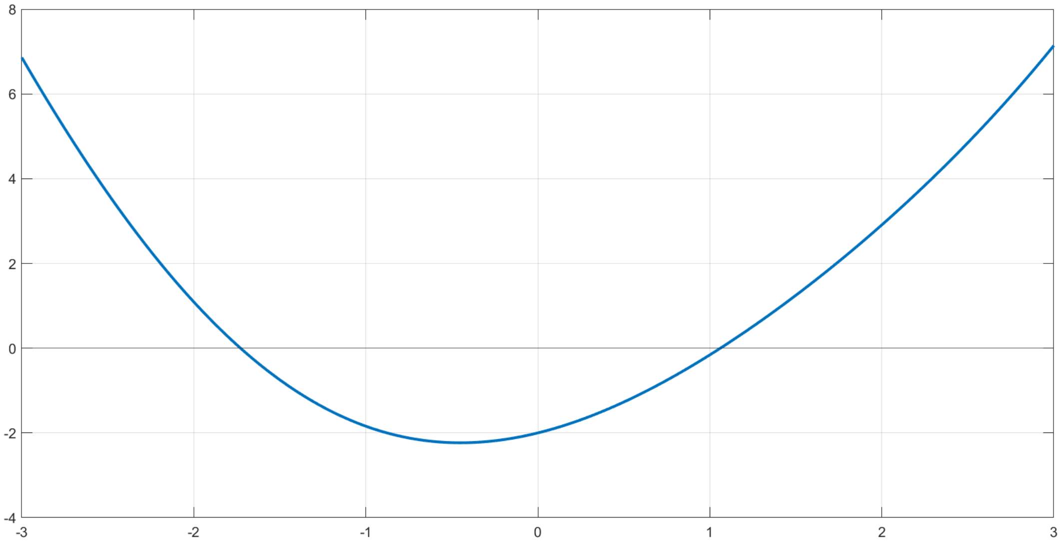

Figura 15: Gráfico de $y_2(t) - y_1(t)$ com $a = 2$

```
x = -3:0.01:3;

plot(x, func_f(2,x), 'LineWidth', 2);
hold on;
grid on;
yline(0);
```

Figura 16: Código MatLab para obtenção do *plot* do gráfico $f(t)$(t0_not_belong.m)