Instituto Superior Técnico

Licenciatura em Matemática Aplicada e Computação

Laboratório de Matemática Computacional – 2º Sem. 20/21

Trabalho Computacional

Realizado por:
José Santos - 85323
Leandro Duarte - 93112
Margarida Ribeiro - 95753
Vasco Pearson - 97015

Professor:
Pedro Lima

6 de junho de 2021

Conteúdo

Parte I

Aproximação de funções por interpolação polinomial

Pretende-se aproximar no intervalo $[0,1]$ a função $r(t)$, ao ser dada uma tabela, designada por suporte de integração, constituída por $n+1$ valores dessa função. Esses pontos são designados nós de interpolação que, neste caso, são da forma $t_i = i/n$, com $i = 0,\ldots,n$. Para tal aproximação, realizar-se-á uma interpolação polinomial, ou seja, serão consideradas apenas funções interpoladoras do tipo polinomial, cujos gráficos devem passar por todos os pontos da tabela.

Definição 0.1 (Polinómio interpolador)**.** Fixado o número inteiro $n \geq 0$, chama-se polinómio interpolador no suporte

$$\{(t_0, r_0), \ldots, (t_n, r_n)\},$$

ao polinómio P_n, de grau menor ou igual a n, que satisfaz as relações

$$P_n(t_i) = r_i, \qquad i = 0, 1, \ldots, n$$

Para a resolução das perguntas seguintes, considera-se esta definição de polinómio interpolador e o facto de este ser único e existir sempre, dado um determinado suporte com nós de interpolação todos distintos.

Tomam-se os polinómios $\left\{\phi_0(t), \phi_1(t), \ldots, \phi_n(t)\right\}$, que formam uma base no conjunto dos polinómios de grau $\leq n$, procura-se o polinómio interpolador na forma

$$P_n(t) = \sum_{i=0}^{n} a_i \phi_i(t)$$

e obtêm-se os coeficientes a_i resolvendo um sistema de equações lineares.

Pergunta 1

Nesta primeira pergunta, onde se considera a base canónica $\{1, t, t^2, ..., t^n\}$, procura-se o polinómio interpolador na forma:

$$P_n(t) = a_0 + a_1 t + a_2 t^2 + ... + a_n t^n$$

e quer-se determinar os coeficientes a_i, números reais, na referida base, resolvendo um sistema de equações lineares. Os números $a_0, ..., a_n$ são as coordenadas do polinómio P_n na base canónica do espaço linear dos polinómios de grau $\leq n$. Recorrendo à definição de polinómio interpolador, verificamos que se forma um sistema linear de $n+1$ equações a $n+1$ incógnitas. Na forma matricial, escreve-se da seguinte maneira:

$$\begin{bmatrix} 1 & t_0 & t_0^2 & ... & t_0^n \\ 1 & t_1 & t_1^2 & ... & t_1^n \\ ... & ... & ... & ... & ... \\ 1 & t_n & t_n^2 & ... & t_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ ... \\ a_n \end{bmatrix} = \begin{bmatrix} r_0 \\ r_1 \\ ... \\ r_n \end{bmatrix}$$

Como neste caso se considera a base canónica, a matriz deste sistema é conhecida como matriz de Vandermonde, representada por $V(t_0, ..., t_n)$. Tendo em conta as suas características,

- Para $n = 1$, $\det V(x_0, x_1) = x_1 - x_0 \neq 0$
- Para $n = 2$, $\det V(x_0, x_1, x_2) = (x_1 - x_0)(x_2 - x_1)(x_2 - x_0) \neq 0$
- Prova-se que
$$\det V(t_0, t_1, ..., t_n) = \prod_{i,j=0, i>j}^{n} (t_i - t_j) \neq 0$$

Conclui-se que, para qualquer n, o determinante da matriz de Vandermonde é diferente de zero, ou seja, a matriz é invertível. Por conseguinte, o sistema tem sempre solução única, para nós de interpolação todos distintos. Desta forma, para cada tabela de valores da função r num conjunto de $n+1$ nós distintos, existe um único polinómio interpolador.

Sendo a matriz de Vandermonde necessária para a construção do polinómio interpolador, é necessário desenvolver um programa no MATLAB que a calcule. Assim sendo, desenvolveu-se o seguinte código, cujo único dado de entrada é o valor de n e que devolve a matriz de Vandermonde, para os nós considerados.

```
function V = vandermonde(n)
% calcula a matriz de Vandermonde, base canonica
% primeiro cria um vetor coluna com os nos de
% interpolacao t=i/n
t = zeros(n,1);
for j = 1:n+1
    t(j) = (j-1)/n;
end
% matriz de Vandermonde, criada atraves de multiplicacoes
% sucessivas, uma vez que o comando .^ seria mais lento
V = ones(length(t), n+1);
for i = 2:n+1
  V(:, i) = t .* V(:, i-1);
end
end
```

Problema 2

Nesta pergunta, a par da base canónica, considera-se a base de Bernstein.

A base de Bernstein é constituída pelos polinómios

$$B_{0,n}, B_{1,n}, \ldots, B_{n,n}$$

os quais são definidos pelas expressões:

$$B_{k,n}(t) = \binom{n}{k} t^k (1-t)^{n-k}, \qquad k = 0, 1, \ldots, n$$

onde

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

designa um coeficiente binomial.

Portanto, procura-se o polinómio interpolador na forma:

$$P_n(t) = \sum_{k=0}^{n} a_k B_{k,n}(t) = \sum_{k=0}^{n} a_k \binom{n}{k} t^k (1-t)^{n-k}$$

e quer-se determinar os coeficientes a_i, números reais, na referida base de Bernstein, resolvendo um sistema de equações lineares. O sistema linear é de $n+1$ equações a $n+1$ incógnitas e, na forma matricial, escreve-se da seguinte maneira:

$$\begin{bmatrix} (1-t_0)^n & n t_0 (1-t_0)^{n-1} & \binom{n}{2} t_0^2 (1-t_0)^{n-2} & \ldots & t_0^n \\ (1-t_1)^n & n t_1 (1-t_1)^{n-1} & \binom{n}{2} t_1^2 (1-t_1)^{n-2} & \ldots & t_1^n \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ (1-t_n)^n & n t_n (1-t_n)^{n-1} & \binom{n}{2} t_n^2 (1-t_n)^{n-2} & \ldots & t_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \ldots \\ a_n \end{bmatrix} = \begin{bmatrix} r_0 \\ r_1 \\ \ldots \\ r_n \end{bmatrix}$$

Tal como verificámos na pergunta anterior, para qualquer tabela dada existe um único polinómio interpolador. A única diferença é que, neste caso, são considerados os polinómios de Bernstein, que também formam uma base no conjunto dos polinómios de grau menor ou igual a n.

Sendo a matriz, quando se usa a base de Bernstein, necessária para a construção do polinómio interpolador, é necessário desenvolver um programa no MATLAB que a calcule. Assim sendo, desenvolveu-se o seguinte código, cujo único dado de entrada é o valor de n e que devolve a respetiva matriz, para os nós considerados.

```
function B = bernstein(n)
% calcula a matriz utilizando a base de Bernstein
% primeiro cria um vetor coluna com os nos de
% interpolacao t=i/n
t = zeros(n,1);
for j = 1:n+1
    t(j) = (j-1)/n;
end
B = zeros(n+1,n+1);
l = length(t);
for k = 0:n
    for i = 1:l
        % expressao dos polinomios de Bernstein
        B(i,k+1) = nchoosek(n,k)*t(i)^k*(1-t(i))^(n-k);
    end
end
end
```

Pergunta 3

O comando *cond(M,inf)* do MATLAB devolve o número de condição da matriz M na norma ∞:

$$cond_\infty(M) = \|M\|_\infty \left\|M^{-1}\right\|_\infty$$

onde a norma ∞ (norma matricial por linha) é definida por:

$$\|M\|_\infty = \max_{i=1,\ldots,n} \sum_{j=1}^{n} \left|m_{ij}\right|$$

Com a ajuda deste comando, criámos um *Script* com o nome *numcondicao* que calcula os números de condição, $cond_\infty(M)$, das matrizes associadas aos problemas de interpolação considerados, com $n = 2,\ldots,20$. De forma a ser mais fácil comparar os dois casos, segue-se a tabela que resume os resultados obtidos:

n	**Base canónica**	**Base de Bernstein**
2	24	3
3	216	5.6667
4	1.7067×10^3	15.2222
5	1.2500×10^4	33.5333
6	9.8784×10^4	89.2400
7	8.1271×10^5	210.2305
8	6.2915×10^6	559.7650
9	4.8184×10^7	1.3674×10^3
10	4.0042×10^8	3.6480×10^3
11	3.1739×10^9	9.1115×10^3
12	2.4228×10^{10}	2.4358×10^4
13	2.0000×10^{11}	6.1747×10^4
14	1.6055×10^{12}	1.6535×10^5
15	1.2485×10^{13}	4.2362×10^5
16	1.0057×10^{14}	1.1361×10^6
17	8.1337×10^{14}	2.9335×10^6
18	6.4317×10^{15}	7.8771×10^6
19	5.0131×10^{16}	2.0463×10^7
20	3.3588×10^{17}	5.5004×10^7

TABELA 1: Números de condição das matrizes: base canónica e base de Bernstein

Para mostrarmos ser exponencial o crescimento dos números de condição da matriz de Vandermonde e da matriz associada à base de Bernstein, criámos o *Script graficocond*, que desenha os gráficos da sucessão $\ln(cond_\infty(M))$ em função de n, para as duas bases. Seguem-se os gráficos:

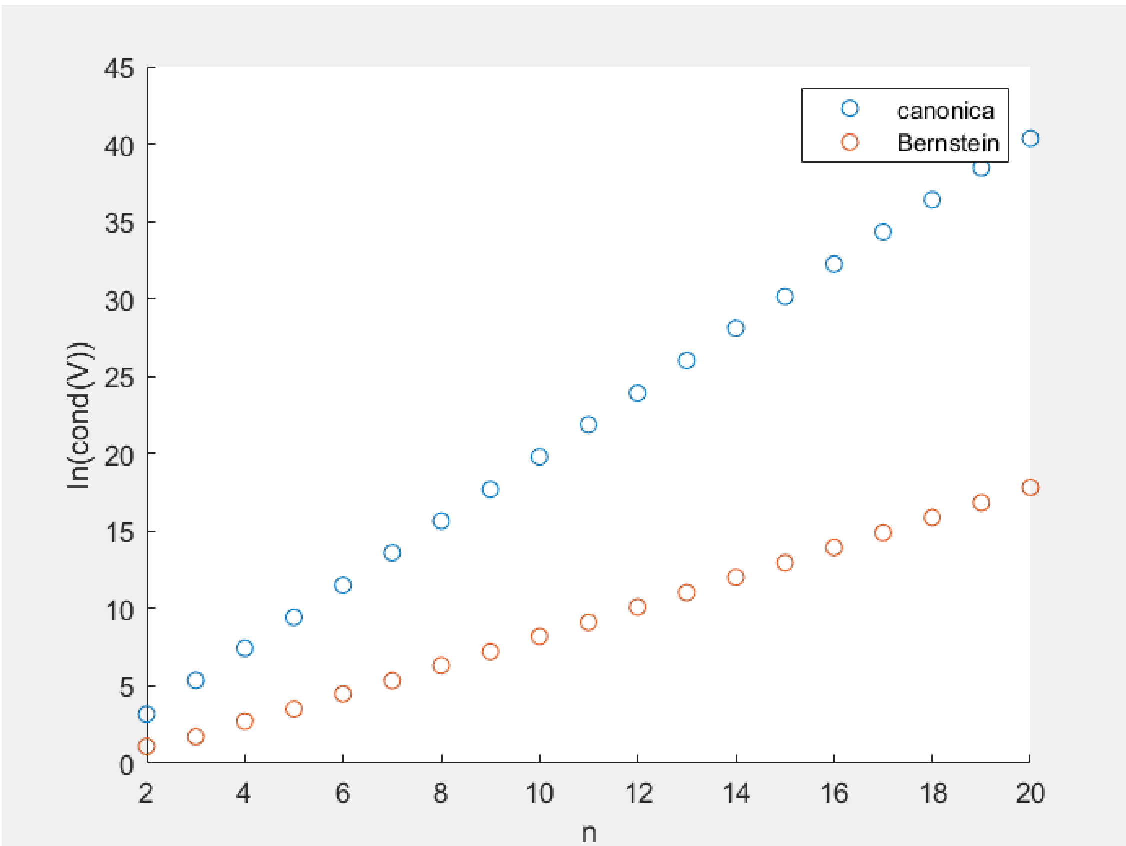

FIGURA 1: Gráficos de $\ln(cond_\infty(M))$ em função de n, para as duas bases

Tal como esperado, ambos os gráficos se aproximam a uma reta, de declive C. Então temos:

$$\ln(cond_\infty(M)) \approx Cn$$

Colocando a expressão em função do número de condição, obtemos:

$$cond_\infty(M) \approx \exp(Cn)$$

Para obtermos experimentalmente o valor de C, para cada uma das bases, efetuámos uma regressão linear, obtendo os resultados que se encontram na figura seguinte:

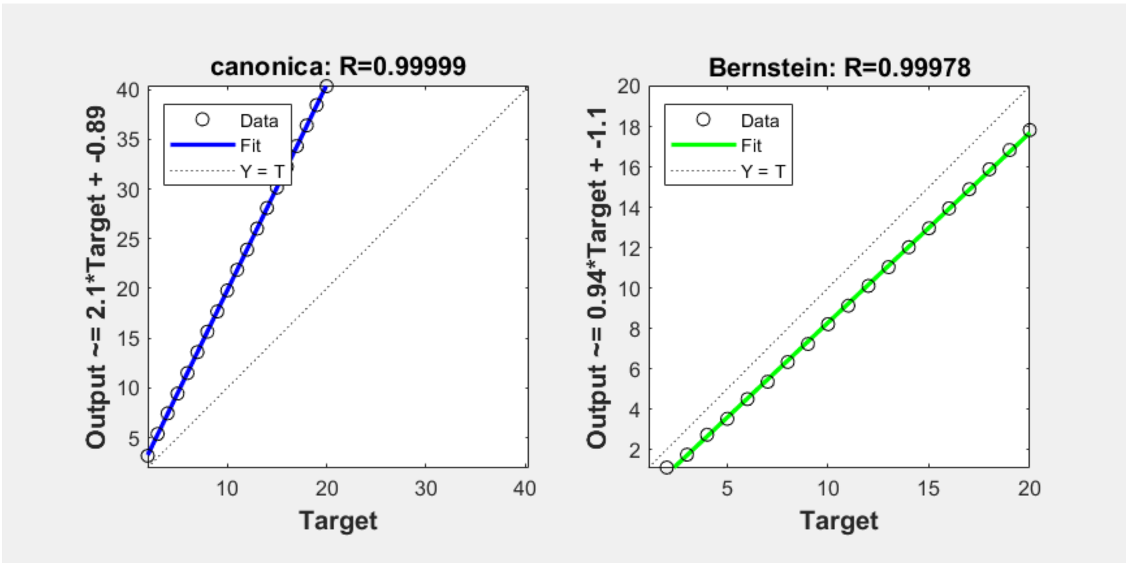

FIGURA 2: Gráficos das regressões lineares, para as duas bases

Como se pode observar, os valores obtidos foram:

- Base canónica: $cond_\infty(V) \approx \exp(2.1n)$
- Base de Bernstein: $cond_\infty(B) \approx \exp(0.94n)$

Com base na tabela e nos gráficos, podemos concluir que, neste suporte que contém os nós $t_i = i/n$, os números de condição da matriz de Vandermonde e da matriz associada à base de Bernstein crescem de forma exponencial, conforme aumentam as suas dimensões. No entanto, é visível que o número de condição da matriz de Vandermonde cresce muito mais abruptamente, sendo logo bastante elevado, mesmo para valores moderados de n. Ou seja, o sistema associado ao suporte de interpolação considerado é extremamente mal condicionado: um pequeno erro relativo nos valores da tabela pode levar a grandes erros relativos nos valores do polinómio interpolador. Embora a matriz associada à base de Bernstein ainda seja mal condicionada, é muito menos quando comparada com a de Vandermonde, o que leva a uma menor perda de precisão no cálculo computacional do polinómio interpolador. Esta é uma das principais vantagens da utilização desta matriz, especialmente para valores moderados de n, uma vez que o número de condição ainda é relativamente baixo. Além disso, é também uma vantagem, em termos numéricos, que os polinómios de Bernstein sejam sempre não negativos.

No entanto, na base de Bernstein, o grau dos polinómios é dependente do número de nós de interpolação considerados e, para graus elevados, é difícil ter precisão, fazendo com que a curva obtida não acompanhe de forma razoável a curva pretendida. Além disso, fazendo qualquer modificação nos nós de interpolação, é necessário recalcular toda a curva novamente.

Quanto à matriz de Vandermonde, apesar da sua simplicidade teórica, o seu mau condicionamento, leva a que esta não deva ser utilizada para calcular o polinómio interpolador de um suporte contendo os nós considerados ($t_i = i/n$). Além disso, computacionalmente, a resolução de um sistema linear $n+1 \times n+1$, tem um custo muito elevado. Devem ser procuradas fórmulas alternativas para calcular o polinómio interpolador, tais como a fórmula interpoladora de Lagrange ou utilizando a base de Bernstein.

Pergunta 4

O polinómio interpolador de um dado suporte de dados é único: independentemente do algoritmo que usarmos para o construir, o polinómio final será sempre o mesmo. No entanto, coloca-se a questão prática de o calcular, no sentido de obter o método computacionalmente mais eficiente e estável possível.

Nesta pergunta, tendo em conta que é uma das fórmulas mais simples para a construção do polinómio interpolador, pretende-se usar a fórmula interpoladora de Lagrange. Começamos por definir $n+1$ polinómios linearmente independentes, que formam uma base dos polinómios de grau não superior a n, designada por base de Lagrange. Esses polinómios vão ser designados como polinómios de Lagrange, representados por $L_i(t)$, com $i = 0, 1, ..., n$. Os polinómios são todos de grau n, são construídos para um dado conjunto de nós distintos (neste caso, $t_i = i/n$) e estabelecem uma correspondência entre cada nó de interpolação t_i e o polinómio L_i, que se estabelece do seguinde modo:

$$L_i(t_j) = \begin{cases} 1, & \text{se } i = j \\ 0, & \text{caso contrario} \end{cases}$$

com $j = 0, 1, ..., n$.

Portanto, obtém-se que um polinómio de Lagrange deve ter a forma:

$$\boxed{L_i(t) = \frac{\prod_{j=0, j\neq i}^{n} (t - t_j)}{\prod_{j=0, j\neq i}^{n} (t_i - t_j)}, \qquad i = 0, ..., n}$$

Para os nós de interpolação $t_i = i/n$, com $i = 0, ..., n$ temos:

$$L_i(t) = \frac{(t-0)(t-1/n)...(t-\frac{i-1}{n})(t-\frac{i+1}{n})...(t-1)}{(i/n-0)(i/n-1/n)...(i/n-\frac{i-1}{n})(i/n-\frac{i+1}{n})...(i/n-1)}$$

Como estes polinómios constituem um conjunto de $n+1$ funções linearmente independentes, então formam uma base dos polinómios de grau não superior a n, a base de Lagrange associada aos nós $t_0, ..., t_n$. Por conseguinte, dada uma tabela de valores da função r nos pontos t_i, o polinómio interpolador de r nesses nós pode ser representado, de forma única, como:

$$\boxed{P_n(t) = \sum_{j=0}^{n} r(t_j) L_j(t)}$$

Substituindo t por t_i temos:

$$P_n(t_i) = \sum_{j=0}^{n} r(t_j) L_j(t_i) = r_i \qquad i = 0, 1, ..., n$$

Sabendo isto, para a base de Lagrange, temos que a matriz do sistema linear é a matriz identidade, uma matriz bem condicionada e um sistema que é rapidamente resolvido.

$$\begin{bmatrix} 1 & 0 & ... & 0 \\ 0 & 1 & ... & 0 \\ ... & ... & ... & ... \\ 0 & ... & 0 & 1 \end{bmatrix} \begin{bmatrix} r_0 \\ r_1 \\ ... \\ r_n \end{bmatrix} = \begin{bmatrix} r_0 \\ r_1 \\ ... \\ r_n \end{bmatrix}$$

Pergunta 5

O objetivo é aplicar os dois métodos (base canónica e base de Bernstein) à resolução de um problema concreto.

A função para a qual se pretende determinar expressões analíticas é

$$r : [0,1] \subset \mathbb{R} \longmapsto \mathbb{R}^2$$

tal que $r(t) = \big(x(t), y(t)\big)$ represente a curva da figura apresentada no enunciado do Trabalho Computacional, para qualquer valor do parâmetro $t \in [0,1]$. Para aproximar $x(t)$, componente da curva $r(t)$, utiliza-se interpolação polinomial de grau adequado à tabela dada. O mesmo se faz para aproximar $y(t)$.

Pergunta 5 a)

Nesta alínea *a*), aplica-se o método, no caso da base canónica, para resolver o problema em questão. Executa-se, portanto, o programa da pergunta 1, a fim de se calcular a matriz deste sistema linear. Uma vez que os nós de interpolação são $t_0 = 0$, $t_1 = 1/3$, $t_2 = 2/3$, $t_3 = 1$, facilmente se conclui que o valor de n adequado é $n = 3$. Assim sendo, fazemos *vandermonde(3)* e a matriz do sistema linear obtida é a seguinte:

```
V =
    1.0000         0         0         0
    1.0000    0.3333    0.1111    0.0370
    1.0000    0.6667    0.4444    0.2963
    1.0000    1.0000    1.0000    1.0000
```

NOTA: Todas as operações efetuadas estão resumidas no *Script pergunta5*.

- Para aproximar a função $x(t)$:

  Da tabela retiramos os seguintes valores:

  - $x(0) = 0$
  - $x(1) = 2$
  - $x(2) = 0$
  - $x(3) = 2$

  Portanto, queremos resolver o seguinte sistema linear:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1/3 & 1/9 & 1/27 \\ 1 & 2/3 & 4/9 & 8/27 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 0 \\ 2 \end{bmatrix}$$

  Usando o comando $V \backslash x$ do MATLAB, obtivemos o seguinte resultado:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 20 \\ -54 \\ 36 \end{bmatrix}$$

  Conclui-se que o polinómio interpolador para $x(t)$ é:

$$P_3(t) = 20t - 54t^2 + 36t^3$$

- Para aproximar a função $y(t)$:

  Da tabela retiramos os seguintes valores:

  - $y(0) = 0$
  - $y(1) = 2$
  - $y(2) = 3$
  - $y(3) = 4$

  Portanto, queremos resolver o seguinte sistema linear:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1/3 & 1/9 & 1/27 \\ 1 & 2/3 & 4/9 & 8/27 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

  Usando o comando $V \backslash y$ do MATLAB, obtivemos o seguinte resultado:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 8.5 \\ -9 \\ 4.5 \end{bmatrix}$$

  Conclui-se que o polinómio interpolador para $y(t)$ é:

$$P_3(t) = 8.5t - 9t^2 + 4.5t^3$$

Pergunta 5 b)

Na alínea *b*), aplica-se o método, no caso da base de Bernstein, para resolver o problema em questão. Executa-se, portanto, o programa da pergunta 2, a fim de se calcular a matriz deste sistema linear. Uma vez que os nós de interpolação são $t_0 = 0, t_1 = 1/3, t_2 = 2/3, t_3 = 1$, facilmente se conclui que o valor de n adequado é $n = 3$. Assim sendo, fazemos *bernstein(3)* e a matriz do sistema linear obtida é a seguinte:

```
B =
    1.0000         0         0         0
    0.2963    0.4444    0.2222    0.0370
    0.0370    0.2222    0.4444    0.2963
         0         0         0    1.0000
```

NOTA: Todas as operações efetuadas estão resumidas no *Script pergunta5*.

- Para aproximar a função $x(t)$:

  Da tabela retiramos os seguintes valores:

  - $x(0) = 0$
  - $x(1) = 2$
  - $x(2) = 0$
  - $x(3) = 2$

  Portanto, queremos resolver o seguinte sistema linear:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 8/27 & 4/9 & 2/9 & 1/27 \\ 1/27 & 2/9 & 4/9 & 8/27 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 0 \\ 2 \end{bmatrix}$$

Usando o comando $B\backslash x$ do MATLAB, obtivemos o seguinte resultado:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 20/3 \\ -14/3 \\ 2 \end{bmatrix}$$

Este resultado encontra-se na base canónica $\{1, t, t^2, t^3\}$. A fim de obter o polinómio interpolador para $x(t)$, é preciso efetudar uma mudança para a base de Bernstein

$$\{(1-t)^3, 3t(1-t)^2, 3t^2(1-t), t^3\} =$$
$$= \{-t^3+3t^2-3t+1, 3t^3-6t^2+3t, -3t^3+3t^2, t^3\}$$

Portanto, a matriz mudança de base é:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

Fazemos:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 20/3 \\ -14/3 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 20 \\ -54 \\ 36 \end{bmatrix}$$

Conclui-se que o polinómio interpolador para $x(t)$ é:

$$P_3(t) = 20t - 54t^2 + 36t^3$$

- Para aproximar a função $y(t)$:

  Da tabela retiramos os seguintes valores:

  - $y(0) = 0$
  - $y(1) = 2$
  - $y(2) = 3$
  - $y(3) = 4$

  Portanto, queremos resolver o seguinte sistema linear:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 8/27 & 4/9 & 2/9 & 1/27 \\ 1/27 & 2/9 & 4/9 & 8/27 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

  Usando o comando $B\backslash y$ do MATLAB, obtivemos o seguinte resultado:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 17/6 \\ 8/3 \\ 4 \end{bmatrix}$$

  Este resultado encontra-se na base canónica, $\{1, t, t^2, t^3\}$. A fim de obter o polinómio interpolador para $y(t)$, é preciso efetuar uma mudança para a base de Bernstein

$$\{(1-t)^3, 3t(1-t)^2, 3t^2(1-t), t^3\}$$
$$\{-t^3+3t^2-3t+1, 3t^3-6t^2+3t, -3t^3+3t^2, t^3\}$$

  Portanto, a matriz mudança de base é:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

Fazemos:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 17/6 \\ 8/3 \\ 4 \end{bmatrix} = \begin{bmatrix} 0 \\ 17/2 \\ -9 \\ 9/2 \end{bmatrix}$$

Conclui-se que o polinómio interpolador para $y(t)$ é:

$$P_3(t) = 8.5t - 9t^2 + 4.5t^3$$

Pergunta 5 c)

A curva $r(t) = \big(x(t), y(t)\big)$ obtida é tal que:

- o polinómio interpolador que aproxima $x(t)$ é: $P_3(t) = 20t - 54t^2 + 36t^3$
- o polinómio interpolador que aproxima $y(t)$ é: $P_3(t) = 8.5t - 9t^2 + 4.5t^3$

De acordo com estes resultados, mediante uma rotina gráfica do MATLAB realizada pelo *Script* com o nome *grafico* e com ajuda do comando *plot*, obtivemos um gráfico contendo os pontos da curva $r(t)$, um gráfico análogo ao da Figura 1 do enunciado do Trabalho Computacional. O gráfico obtido é o que se segue:

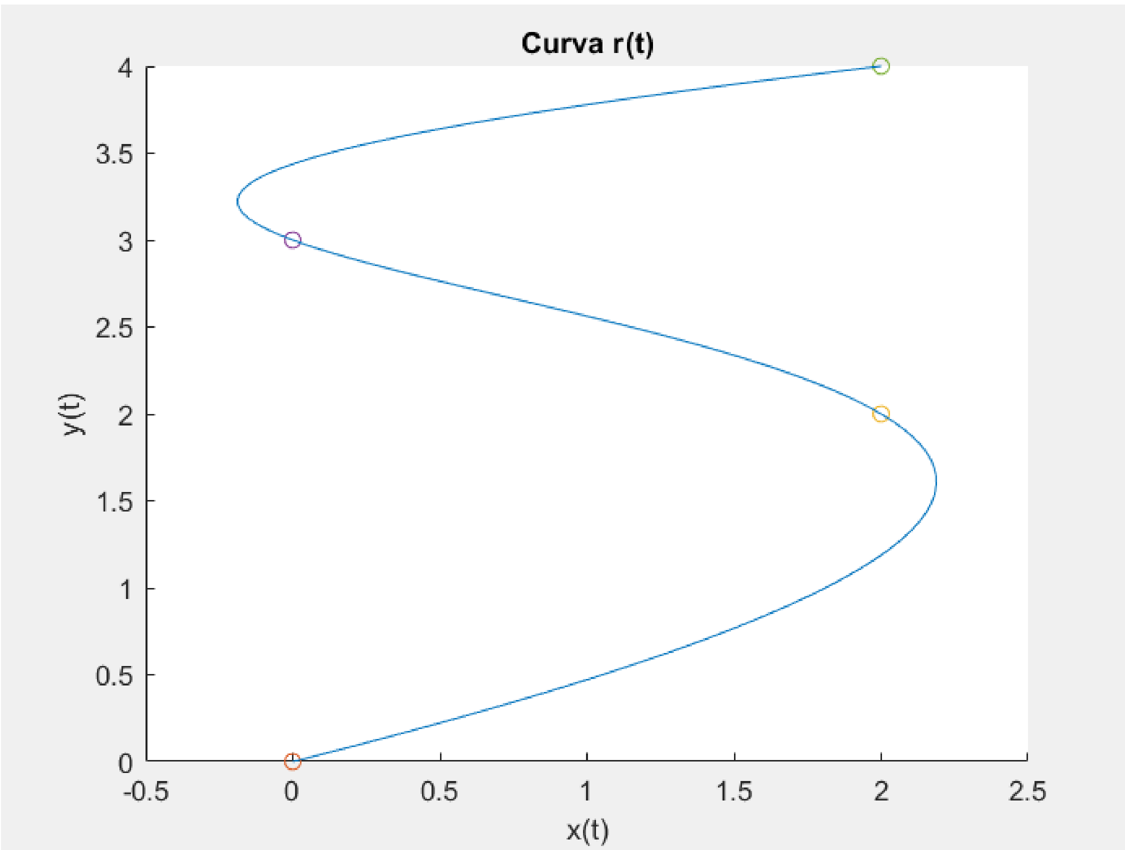

FIGURA 3: Gráfico da curva $r(t)$, que passa pelos pontos P_0 a P_3

Parte II

Resolução numérica de equações diferenciais

Pretende-se aplicar um método numérico para aproximar a solução $y(x)$ de um PVI (problema de valor inicial) num certo intervalo $[0, a]$ usando um certo passo h.

Definição 0.2 (Equação diferencial de ordem n)**.** Uma equação diferencial é uma equação que relaciona uma ou mais funções com as suas derivadas. Dizemos que uma equação diferencial ordinária é de ordem n quando é uma relação do seguinte tipo

$$F(x, y, y', y'', ..., y^{(n)})$$

Definição 0.3 (Problema de valor inicial)**.** Um problema de valor inicial (ou Problema de Cauchy) corresponde a uma equação diferencial que está acompanhada do valor da função objetivo num dado ponto (valor inicial).

$$\begin{cases} y'(t) = f(t, y(t)) \\ y(t_0) = y_0 \end{cases}$$

É importante termos conhecimento à priori sobre a existência e unicidade de solução da equação que estamos a tentar resolver. Para isso temos o seguinte teorema.

Teorema 1 (Existência e unicidade de solução)**.** Seja f definida e contínua em $D = \{(x, y) : a \le x \le b, y \in \mathbb{R}\}$ com a e b finitos. Seja $\frac{\partial f}{\partial y}$ contínua e limitada em D. Então $\forall x_0 \in [a, b]$ e $y_0 \in \mathbb{R}$, o problema tem solução única $y = y(x)$ continuamente diferenciável para $x \in [a, b]$.

Nem sempre se podem aplicar métodos analíticos para a resolução de equações diferenciais, pois apenas num certo tipo de problemas é que estes nos vão ajudar a encontrar a solução pretendida. Por isso recorre-se com frequência ao uso de métodos numéricos para obter a solução de uma equação diferencial sujeita a uma dada condição.

Pergunta 1

Nesta questão é nos pedido para aproximar a solução $y(x)$ do PVI

$$y'(x) = \sin(\exp(y(x))), \qquad y(0) = 0$$

através do **Método dos Trapézios**, num certo intervalo $[0, a]$ e com certo passo h.

O método dos trapézios é um método numérico explícito utilizado na resolução de equações diferenciais ordinárias com condição inicial.

Este método é também conhecido como **Método de Heun** e faz parte da família dos métodos de **Runge-Kutta** de segunda ordem de convergência, que se definem pela seguinte equação:

$$y_{i+1} = y_i + h\left[(1 - \frac{1}{2\alpha})f(t_i, y_i) + \frac{1}{2\alpha}(f(t_i + \alpha h, y + \alpha h f(t_i, y_i))\right]$$

Substituindo o parâmetro α por $\alpha = 1$ obtém-se o método de Heun que resolve equações do tipo $y' = f(t, y)$ e é dado pela seguinte fórmula :

Método dos trapézios

$$y_{n+1} = y_n + \frac{h}{2}(f(t_n, y_n) + f(t_{n+1}, \tilde{y}_{n+1}))$$

onde $\tilde{y}_{n+1} = y_n + hf(t_n, y_n)$
$h = t_n - t_{n-1}$ é o *step*

Inicialmente, de modo a confirmar a existência e unicidade da solução do problema dado aplicou-se o **Teorema 1**: Sejam

$$f(x, y) = \sin(\exp(y)), \quad \frac{\partial f}{\partial y} = \exp(y)\cos(\exp(y)),$$

$$D = \left\{(x, y) \in \mathbb{R}^2 : 0 < x < a, \quad y \in \mathbb{R}\right\}$$

Para quaisquer valores de (x, y) em D a função $f(x, y)$ é obviamente contínua e $\frac{\partial f}{\partial y}$ é contínua e limitada no intervalo considerado. Assim é correto afirmar que existe solução única do problema de valor inicial enunciado.

De modo a responder à questão foi implementado em MATLAB o programa *trapezios* que aproxima a solução $y(x)$ do problema de valor inicial do enunciado através do método dos trapézios.

O programa recebe como dados de entrada o espaçamento $h = x_{k+1} - x_k$ e o valor a do intervalo $[0, a]$, onde se pretende aproximar a solução. Este retorna a lista de valores de y da função aproximada e a lista com os valores de x correspondentes através das variáveis *ylist* e *xlist*, respetivamente. Ambas têm uma dimensão igual a $\left(\frac{a}{h} + 1\right)$ pois definiu-se como primeiro termo $(x_0, y_0) = (0, 0)$.

```
function [xlist, ylist] = trapezios(h, a)
% Aproxima a solucao y atraves do metodo dos trapezios
% INPUT: passo h para as iteracoes do metodo e extremo a do
    intervalo de valores de x
% OUTPUT: Valores (x,y) da aproximacao da funcao solucao da
    eq. diferenciavel
x0=0;
y0=0;
xlist=[x0];
ylist=[y0];
for x0=h:h:a
    y1=y0+h*sin(exp(y0));
    y=y0+1/2*h*(sin(exp(y0))+sin(exp(y1)));
    y0=y;
    xlist = [xlist x0];
    ylist=[ylist y];
end
```

Exemplificando a primeira iteração para $h = 0.1$:

1. $y_1 = 0 + \frac{0.1}{2}\Big(\sin(\exp(0)) + \sin(\exp(0 + 0.1 * (\sin(\exp(0)))))\Big) \approx 0.086353656431469$

Pergunta 2

Nesta questão iremos apresentar um valor $z \in \mathbb{R}^+$, tal que $y(z) = 1$, sendo $y(x)$ a solução aproximada na alínea anterior.

Primeiramente apresenta-se o gráfico da função aproximada $y(x)$ com os parâmetros $a = 7$ e $h = 0.1$:

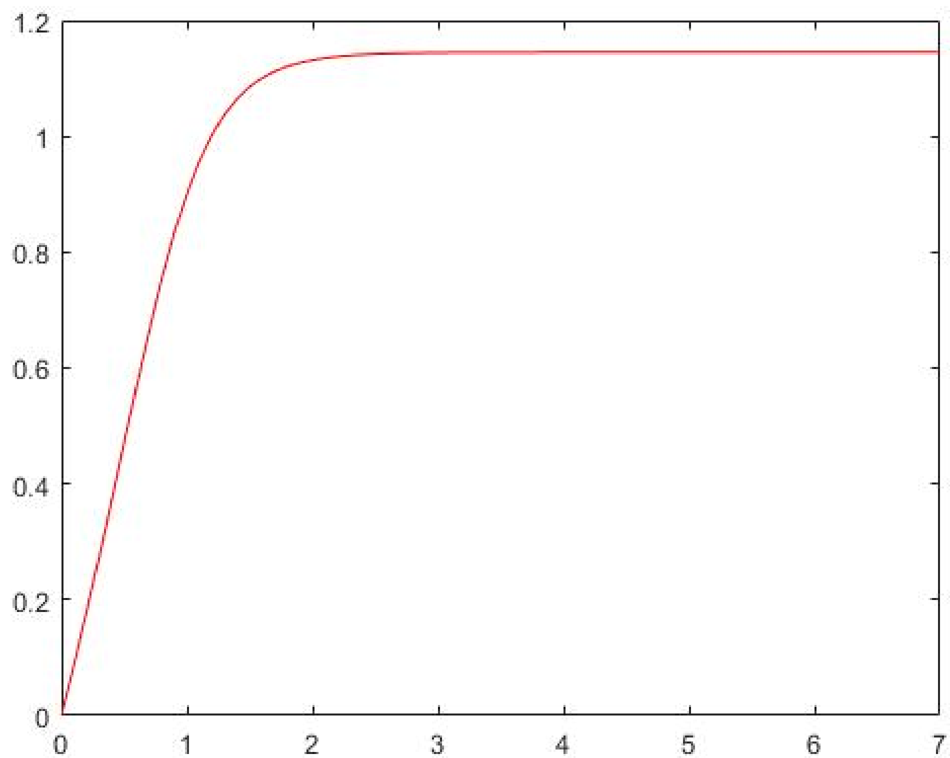

FIGURA 4: Aproximação de $y(x)$ no intervalo $0 < x < 7$.

Através da observação do gráfico definiu-se que $a = 2$ é um valor aceitável para definir o intervalo da aproximação de modo a escolher o passo h mais adequado para o método dos trapézios e definir assim o valor de z com um erro absoluto inferior a 10^{-3}.

Verifica-se ainda que a função $y(x)$ é monótona crescente no intervalo $[0,a]$. Facilmente se observa que $y'(x) = \sin(e^y) = 0 \iff y = \log(k\pi)$, $k \in \mathbb{N}$ e que $y'(x_0) = \sin(e^{y(x_0)}) = \sin(e^0) > 0$. Ao analisar os valores obtidos pela execução da função *trapezios* concluímos que os valores de $y(x)$ pertencem ao intervalo $]0,\log(\pi)[$, pois os valores de $y(x)$ tendem para $\log(\pi) \approx 1.14473$, por valores menores que $\log(\pi)$. Assim podemos concluir que a derivada de $y(x)$ nunca se anula e é sempre positiva, portanto $y(x)$ é monótona crescente.

Tendo em conta a observação anterior, foi implementada a função *interpol* que recebe como argumentos o passo $h = x_{k+1} - x_k$, o valor a do intervalo $[0,a]$ e o valor Y tal que $y(z) = Y$. Inicialmente recorre à função *trapezios* definida na alínea anterior e calcula os valores (x_i, y_i) da solução aproximados ($[xlist, ylist] = trapezios(h,a)$). A função *interpol* cumpre duas tarefas:

1. Encontra os índices das listas *xlist* e *ylist* cujos os valores formam o menor intervalo que contém os valores (z,Y), tendo em conta que a função solução $y(x)$ é monótona crescente.

2. Realiza **interpolação linear** com os extremos desse intervalo e devolve o valor de z tal que $p_1(z) = Y$.

   Calculado o declive m obteve-se o polinómio $p_1(x) = mx + (y_{indMin} - mx_{indMin})$ que permitiu calcular

$$z = xlist(indMin) + \Big(Y - ylist(indMin)\Big)/m$$

O valor de z devolvido é a solução da equação a resolver $y(z) = 1$ para $Y = 1$.

```
function [z] = interpol(h, a, Y)
%Realiza interpolacao linear no intervalo que contem Y e
    devolve o valor z tal que y(z)=Y
[xlist, ylist]=trapezios(h, a);
min=abs(Y-ylist(1));
indMin=1;
for i=1:(a/h+1)   %considerando que a funcao e monotona
    crescente
    if abs(Y-ylist(i))<min
        min=abs(Y-ylist(i));
```

```
        indMin=i; %Obtemos o valor de ylist mais proximo de
             y
    end
end

if Y-ylist(indMin)<0  %Obtemos o segundo valor de ylist
    mais proximo de y
    ind2=indMin-1;
else
    ind2=indMin+1;
end

m=(ylist(ind2)-ylist(indMin))/(xlist(ind2)-xlist(indMin));
    %Declive
z=xlist(indMin)+(Y-ylist(indMin))/m;
end
```

De modo a finalizar a questão é necessário avaliar o **erro absoluto** e a relação deste com o erro cometido pelo **Método dos Trapézios** e pela **Interpolação Linear**.

Como já foi visto, o Método dos Trapézios faz parte da família dos métodos de Runge-Kutta de segunda ordem de convergência e portanto tem um erro da ordem $0(h^3)$. Sendo este um método de ordem 2, na majoração do erro existe uma constante **C** (independente de h) tal que

$$\max_{k=1,\ldots,n} |e(x_k)| = Ch^2$$

A constante C é independente de h e portanto para a determinar atribuímos diferentes valor a h e resolvemos o seguinte sistema

$$\begin{cases} \max\limits_{k=1,\ldots,n} |y(x_k) - y_k| = Ch^2 \\ \max\limits_{k=1,\ldots,n} |y‘(x_k) - y_k| = Ch‘^2 \end{cases}$$

obtendo-se

$$C = \frac{\max|y(x_k) - y'(x_k)|}{h^2 - (h‘^2)}$$

Usando $h = 0.005$ e depois $h/2$ obtemos $C = 0.2158$.

Esta constante foi calculada através da função *const*, que utiliza os valores da solução obtidos com a função *trapezios* e obtém a constante C através da fórmula deduzida anteriormente.

```
function [C] = const(h, a)
%Calcula constante da majoracao do erro do metodo dos trapezios Ch^2
[x1, y1]=(trapezios(h, a));
[x2, y2]=(trapezios(h/2, a));
y2=y2(1:2:length(y2));
y=y1-y2;
C=max(abs(y))/(h^2-(h/2)^2);
end
```

No caso da interpolação linear temos a seguinte majoração do erro

$$e_{abs}(x) \leq \frac{\max\limits_{x\in[0,a]} y''(x)}{2} |(x-x_0)(x-x_1)|$$

Como $y''(x) = \cos(e^{y(x)})e^{y(x)}\sin(e^{y(x)})$, temos $y''(x) \leq e^{y(x)}$. Vimos que $y(x) < log(\pi)$ e portanto $y''(x) < e^{log(\pi)} = \pi$.

Quanto à expressão $|(x-x_0)(x-x_1)| = |x^2-(x_0+x_1)x-x_0x_1|$ sabemos que tem um extremo no ponto $\frac{x_0+x_1}{2}$ de valor $\frac{(x_1-x_0)(x_0-x_1)}{4} = \frac{h^2}{2}$ e portanto temos a majoração

$$e_{abs}(x) < \frac{\pi}{2}\frac{h^2}{4}$$

Podemos assim determinar o valor de h que permite obter uma aproximação de z com erro absoluto inferior a 10^{-3}.

$$e_{abs}(x) < \frac{\pi}{2}\frac{h^2}{4} + 0.2158h^2 = h^2(\frac{\pi}{8} + 0.2158)$$

$$e_{abs} < 10^{-3} \implies h^2(\frac{\pi}{8} + 0.2158) \leq 10^{-3} \iff h \leq \sqrt{\frac{10^{-3}}{\frac{\pi}{8} + 0.2158}} \approx 0.040540265634931$$

Por fim, vamos aproximar o valor $z \in \mathbb{R}^+$ tal que $y(z) = 1$, com um erro absoluto inferior a 10^{-3}. Ao correr a função *interpol* com o valor de h determinado anteriormente $h = 0.04053872$, com $a = 2$ e $Y = 1$ obtemos 1.190237232507747.

```
%Escolha do valor a do intervalo de x atraves da visualizacao do grafico da
%aproximacao de y
[X,Y]=trapezios(0.0001, 7);
grafy=plot(X,Y,'r');
%a=2

%Calculo de h e z
hbest=sqrt(10^(-3)/(pi/8+const(0.005,2)))
zbest=interpol(hbest, 2, 1)
```