Algoritmia Aplicada

Ano lectivo 2011-2012

Aula 9 – Algoritmos Matemáticos: Ajuste de curvas; Matrizes

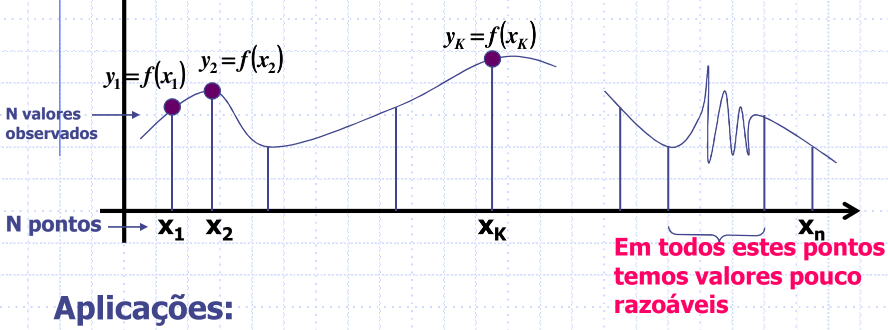
Sumário

- Ajuste de curvas
  - Interpolação polinomial: Fórmula de *Lagrange*
  - Interpolação "*Spline*"
  - Método dos mínimos quadrados
- Matrizes
  - Representação computacional de matrizes esparsas: comparação de diferentes esquemas de representação
  - Adição de matrizes
  - Multiplicação de matrizes

Ajuste de curvas

Problema:

- **Encontrar uma função que satisfaça um conjunto de valores observados num conjunto de pontos e que assuma valores razoáveis noutros pontos**



Aplicações:

- **Análise de dados experimentais**
- **Computação gráfica**

Ajuste de curvas

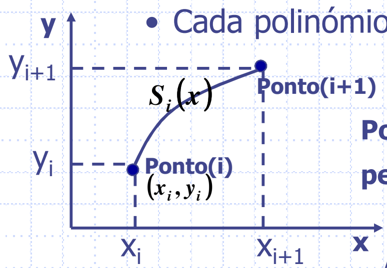
- **Dois tipos de abordagens principais:**
  - **INTERPOLAÇÃO**
    - **Função suave que satisfaz exactamente os valores dados em todos os pontos x_1, x_2, ...x_n**
    - **Iremos ver dois tipos**
      - **Interpolação Polinomial**
      - **Interpolação "Spline"**
  - **MÍNIMOS QUADRADOS**
    - **Os valores dados podem não ser precisos, pelo que não se justifica procurar uma função que os satisfaça exactamente, mas antes uma função que aproxima aqueles valores tanto quanto possível**

Interpolação "SPLINE"

- Ideia: Em vez de calcular um polinómio de grau (N-1) que passe por N pontos do plano, calcular (N-1) polinómios:
  - Cada polinómio é de grau 3:

$$S_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$$
$$i = 1,2,\ldots,N-1$$

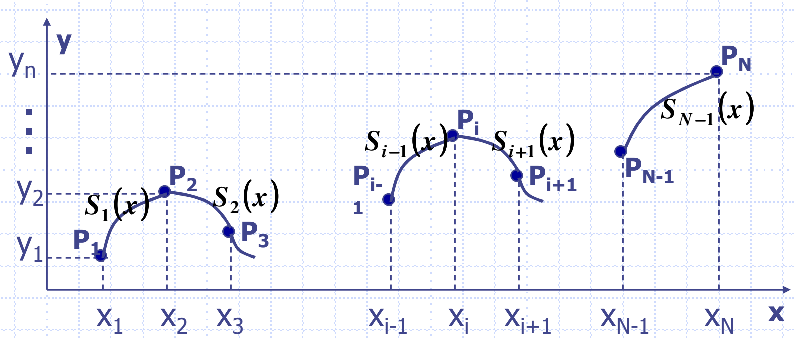
  - Cada polinómio passa por dois pontos consecutivos:



Polinómio $S_i(x)$ passa pelos pontos (x_i, y_i) e (x_{i+1}, y_{i+1})

Interpolação "SPLINE"

- Deste modo, a curva pretendida é constituída por (N-1) partes:



Interpolação "SPLINE"

- **Vantagem:** Apenas usamos polinómios de baixo grau (ao contrário da Interpolação de Lagrange) que são curvas simples e de fácil tratamento analítico

- **Nº de incógnitas a determinar:**
  - Para cada polinómio $S_i(x)$ temos que determinar 4 coeficientes (a_i, b_i, c_i e d_i):

$$S_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$$

  - Como temos N-1 polinómios, teremos:

4(N-1) Coeficientes

É necessário formular 4(N-1) equações

Interpolação "SPLINE"

- **Formulação das equações:**

  a) Cada polinómio deve passar por dois pontos consecutivos:

$$\begin{cases} S_1(x_1) = y_1 \\ S_1(x_2) = y_2 \end{cases} \cdots\cdots\cdots \begin{cases} S_i(x_i) = y_i \\ S_i(x_{i+1}) = y_{i+1} \end{cases} \cdots\cdots\cdots \begin{cases} S_{N-1}(x_{N-1}) = y_{N-1} \\ S_{N-1}(x_N) = y_N \end{cases}$$

ou seja

$$\begin{cases} S_i(x_i) = y_i \\ S_i(x_{i+1}) = y_{i+1} \end{cases} ; i = 1,2,\ldots,N-1$$

2(N-1) equações

Interpolação "SPLINE"

- **Substituindo $S_i(x)$ pela expressão**
  $S_i(x)=a_i x^3+b_i x^2+c_i x+d_i$,

vem $\begin{cases} S_i(x_i)=y_i \\ S_i(x_{i+1})=y_{i+1} \end{cases}$ $;i=1,2,\ldots,N-1$

$$\begin{cases} a_i x_i^3+b_i x_i^2+c_i x_i+d_i=y_i \\ a_i x_{i+1}^3+b_i x_{i+1}^2+c_i x_{i+1}+d_i=y_{i+1} \end{cases} \quad ;i=1,2,\ldots,N-1$$

2(N-1) equações

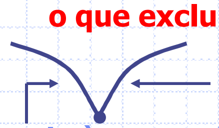
Incógnitas

Interpolação "SPLINE"

- **FORMULAÇÃO das EQUAÇÕES:**
  b) • A curva a obter (composição das várias porções polinomiais) deve ser suave nos pontos em que se dão as uniões

  • A suavidade significa, nomeadamente, que em qualquer ponto da união, as derivadas de 1ª e de 2ª ordem, das duas curvas que aí se unem, são iguais, o que exclui situações como as seguintes (i) e (ii):

(i)

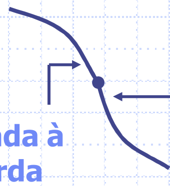


Derivada à direita é positiva (Função crescente)

Derivada à esquerda é negativa (Função decrescente)

Interpolação "SPLINE"

(ii)



2ª Derivada à direita é positiva

Derivada à direita é negativa

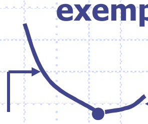
Derivada à esquerda é negativa

... até podem ser iguais mas não o serão as 2ªˢ derivadas ....

2ª Derivada à esquerda é negativa

Interpolação "SPLINE"

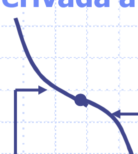
- **As condições de "suavidade" verificam-se, por exemplo, em situações como as seguintes:**



Derivada à direita é negativa
2ª Derivada à direita é positiva

Derivada à esquerda é negativa
2ª Derivada à esquerda é positiva

Há condições para que as 1ª e 2ª derivadas sejam iguais



Derivada à direita é negativa
2ª Derivada à direita é negativa

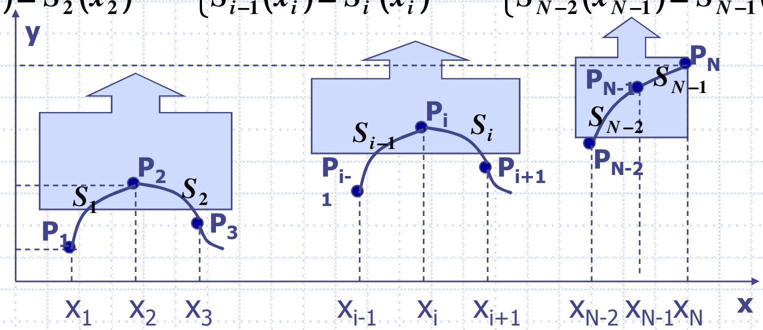
Derivada à esquerda é negativa
2ª Derivada à esquerda é negativa

Idem

Interpolação "SPLINE"

- **A formulação das condições de igualdade das derivadas (de 1ª e 2ª ordem) é a seguinte:**

$$\begin{cases} S_1'(x_2) = S_2'(x_2) \\ S_1''(x_2) = S_2''(x_2) \end{cases} \cdots\cdots \begin{cases} S_{i-1}'(x_i) = S_i'(x_i) \\ S_{i-1}''(x_i) = S_i''(x_i) \end{cases} \cdots\cdots \begin{cases} S_{N-2}'(x_{N-1}) = S_{N-1}'(x_{N-1}) \\ S_{N-2}''(x_{N-1}) = S_{N-1}''(x_{N-1}) \end{cases}$$



Interpolação "SPLINE"

- **Ou seja**

$$\begin{cases} S_{i-1}'(x_i) = S_i'(x_i) \\ S_{i-1}''(x_i) = S_i''(x_i) \end{cases} ; i = 2,3,\ldots,N-1$$

2(N-2) equações

- **Com (recorde-se que** $S_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$ **):**

$$S_{i-1}'(x_i) = 3a_{i-1}x_i^2 + 2b_{i-1}x_i + c_{i-1}$$

$$S_i'(x_i) = 3a_i x_i^2 + 2b_i x_i + c_i$$

$$S_{i-1}''(x_i) = 6a_{i-1}x_i + 2b_{i-1}$$

$$S_i''(x_i) = 6a_i x_i + 2b_i$$

Incógnitas

Interpolação "SPLINE"

- **Até ao momento já dispomos de:**

2(N-1) + 2(N-2) equações

i.e.

4N-6 equações

Para determinar

4(N-1) coeficientes

i.e.

4N-4 coeficientes

Interpolação "SPLINE"

- **As duas equações em falta, serão formuladas para os pontos externos da curva (pontos P_1 e P_N):**

$$\begin{cases} S_1''(x_1) = 0 \\ S_{N-1}''(x_N) = 0 \end{cases}$$

2 equações

- **Ou:**

$$\begin{cases} 6a_1 x_1 + 2b_1 = 0 \\ 6a_{N-1} x_N + 2b_{N-1} = 0 \end{cases}$$

Incógnitas

2 equações

Interpolação "SPLINE"

- **Resolução das equações:**
  - **Acabamos de formular as 4(N-1) ≡ 4N-4 equações lineares necessárias para a determinação dos 4(N-1) ≡ 4N-4 incógnitas**

    Recorde-se que são os coeficientes dos N-1 polinómios

  - **Para a resolução deste sistema de equações lineares, podemos usar um qualquer método eficiente, sendo de notar que a matriz dos coeficientes é bastante esparsa (por exemplo as 2 equações acima só têm 2 coeficientes não nulos – os restantes (4N-6) coeficientes valem 0)**

  - **Uma adequada mudança de variável $x \rightarrow t$, permitiria reduzir significativamente a dimensão do sistema de equações a resolver**

Método dos Mínimos Quadrados

a) Se FUNÇÃO – POLINÓMIO / VALORES EXACTOS → **INTERPOLAÇÃO... já visto!**

b) No caso FUNÇÕES GENÉRICAS / DADOS POUCO PRECISOS (medidas têm erros)

- **EXPRESSÃO GERAL PARA *f***

$$f(x) = c_1 f_1(x) + c_2 f_2(x) + \ldots + c_M f_M(x)$$

Combinação linear de M funções conhecidas

Os parâmetros (c_1, c_2, ..., c_M) a determinar por forma a que *f*(x) se ajuste o melhor possível às medidas efectuadas

Método dos Mínimos Quadrados

Critério dos Mínimos Quadrados

$$E = \sum_{j=1}^{N} \left(f\left(x_j\right) - y_j \right)^2$$

Quadrado para que erros de sinal oposto não se anulem

Erro

M Parâmetros a determinar

N Medidas

Objectivo do método

Encontrar os valores dos parâmetros que minimizem o erro - E

Método dos Mínimos Quadrados

Ilustração

- **N =3** ⟵ 3 PONTOS
  M=2 ⟵ FUNÇÃO COM 2 TERMOS
- (x_1, y_1) (x_2, y_2) (x_3, y_3)
  $$f(x) = c_1 f_1(x) + c_2 f_2(x)$$
- **Que C_1 e C_2 minimizam o erro?**

$$E = \left[c_1 f_1(x_1) + c_2 f_2(x_1) - y_1\right]^2 + \left[c_1 f_1(x_2) + c_2 f_2(x_2) - y_2\right]^2 + \left[c_1 f_1(x_3) + c_2 f_2(x_3) - y_3\right]^2 +$$

Método dos Mínimos Quadrados

- **Derivar E em ordem a C_1 e C_2, e igualar a zero**

$$\frac{dE}{dC_1} = 2[c_1 f_1(x_1)+c_2 f_2(x_1)-y_1]f_1(x_1)+$$
$$2[c_1 f_1(x_2)+c_2 f_2(x_2)-y_2]f_1(x_2)+$$
$$2[c_1 f_1(x_3)+c_2 f_2(x_3)-y_3]f_1(x_3)=0$$

- **Escrevendo em ordem a C_1 e C_2 vem:**

$$c_1[f_1(x_1)f_1(x_1)+f_1(x_2)f_1(x_2)+f_1(x_3)f_1(x_3)]+$$
$$c_2[f_2(x_1)f_1(x_1)+f_2(x_2)f_1(x_2)+f_2(x_3)f_1(x_3)]=$$
$$=\underbrace{y_1 f_1(x_1)+y_2 f_1(x_2)+y_3 f_1(x_3)}_{\text{Constantes conhecidas}}$$

Método dos Mínimos Quadrados

- **Procede-se de igual modo com dE/dC_2**
- **Usando Notação Vectorial:**

$x=(x_1, x_2, x_3)$ $y=(y_1, y_2, y_3)$

$x \cdot y = x_1 y_1 + x_2 y_2 + x_3 y_3$ ← **PRODUTO INTERNO**

- **Definindo:**

$$f_1 = (f_1(x_1), f_1(x_2), f_1(x_3))$$
$$f_2 = (f_2(x_1), f_2(x_2), f_2(x_3))$$

Método dos Mínimos Quadrados

- **Podemos escrever:**

$$\frac{\delta E}{\delta C_1} = 0 \rightarrow C_1 f_1 \cdot f_1 + c_2 f_1 \cdot f_2 = y f_1$$

$$\frac{\delta E}{\delta C_2} = 0 \rightarrow C_1 f_2 \cdot f_1 + c_2 f_2 \cdot f_2 = y f_2$$

Sistemas de 2 Equações a 2 Incógnitas (C_1 e C_2)

Notação Matricial

$$\begin{bmatrix} f_1 \cdot f_1 & f_1 \cdot f_2 \\ f_2 \cdot f_1 & f_2 \cdot f_2 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} y f_1 \\ y f_2 \end{bmatrix}$$

Resolução por um qualquer método eficiente

Notação Matricial Compacta

A • C = b

Método dos Mínimos Quadrados

- **Exemplo numérico:**
  - **N=6 ⟶ 6 pontos:**

(1.0, 2.05) ⟵ **(x_1,y_1)**
(2.0, 1.53)
(4.0, 1.26)
(5.0, 1.21)
(8.0, 1.13)
(10.0, 1.1) ⟵ **(x_6,y_6)**

  - **Ajustar por função do tipo:**

$$f(x) = c_1 + \frac{c_2}{x} \Rightarrow M = 2$$

  - **1º Passo: Identificar as funções f_1 e f_2**

$$f(x) = c_1 f_1(x) + c_2 f_2(x)$$ **Fórmula geral**

$$f(x) = c_1 + c_2 \frac{1}{x}$$ **Nosso caso**

$$f_1(x) = 1 \qquad f_2(x) = \frac{1}{x}$$ **Então**

Método dos Mínimos Quadrados

- **2º Passo: Identificar as funções f_1 e f_2**

$$f_1 = (1.0 \quad 1.0 \quad 1.0 \quad 1.0 \quad 1.0 \quad 1.0)$$

$$f_1(x_1) \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad f_1(x_6)$$

$$f_2 = (1.0 \quad 0.5 \quad 0.25 \quad 0.2 \quad 0.125 \quad 0.1)$$

$$f_2(x_1) \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad f_2(x_6)$$

- **3º Passo: Calcular os coeficientes da Matriz A e do vector b**

$$f_1 \cdot f_1 \qquad f_1 \cdot f_2 \qquad f_1 \cdot y$$

$$\underbrace{\begin{bmatrix} 6.000 & 2.175 \\ 2.175 & 1.378 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} c_1 \\ c_2 \end{bmatrix}}_{c} = \underbrace{\begin{bmatrix} 8.280 \\ 3.623 \end{bmatrix}}_{b}$$

$$f_2 \cdot f_1 \qquad f_2 \cdot f_2 \qquad f_2 \cdot y$$

Método dos Mínimos Quadrados

- **Generalização para M parâmetros (C_1, C_2, ..., C_M)**
  - $f(x) = c_1 f_1(x) + c_2 f_2(x) + ... + c_M f_M(x)$
  - **Considerando N observações**

$$((x_1, y_1); (x_2, y_2), \ldots, (x_N, y_N)):$$

$$x = (x_1, x_2, \ldots, x_N)$$
$$y = (y_1, y_2, \ldots, y_N)$$

Método dos Mínimos Quadrados

- **Teremos então:**

$$f_1 = (f_1(x_1), f_1(x_2), \ldots, f_1(x_N))$$

$$f_2 = (f_2(x_1), f_2(x_2), \ldots, f_2(x_N))$$

$$\ldots$$

$$f_M = (f_M(x_1), f_M(x_2), \ldots, f_M(x_N))$$

- **E o seguinte sistema de M equações a M incógnitas**

$$\begin{bmatrix} f_1 \cdot f_1 & f_1 \cdot f_2 & \ldots & f_1 \cdot f_M \\ f_2 \cdot f_1 & f_2 \cdot f_2 & \ldots & f_2 \cdot f_M \\ \ldots & \ldots & \ldots & \ldots \\ f_M \cdot f_1 & f_M \cdot f_2 & \ldots & f_M \cdot f_M \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \ldots \\ c_M \end{bmatrix} = \begin{bmatrix} f_1 \cdot y \\ f_2 \cdot y \\ \ldots \\ f_M \cdot y \end{bmatrix}$$

- **Em notação matricial compacta**

$$a_{ij} = f_i \cdot f_j \qquad A \bullet C = b \qquad b_{ij} = f_i \cdot y$$

Método dos Mínimos Quadrados

- **ALGORITMO para construção da MATRIZ A e do VECTOR b**

Os vectores $\underbrace{f_1, f_2, \ldots, f_M, y}_{\text{M+1 vectores}}$, cada um deles com N elementos

são memorizados num array bidimensional

F [1..M+1, 1..N]

Método dos Mínimos Quadrados

		1	2	....	K	.....	N
$f_1 \longrightarrow$	1	$f_1(x_1)$	$f_1(x_2)$	...	$f_1(x_k)$	...	$f_1(x_N)$
$f_2 \longrightarrow$	2	$f_2(x_1)$	$f_2(x_2)$	...	$f_2(x_k)$	...	$f_2(x_N)$
$f_i \longrightarrow$	I	$f_i(x_1)$	$f_i(x_2)$	...	$f_i(x_k)$	...	$f_i(x_N)$
$f_M \longrightarrow$	M	$f_M(x_1)$	$f_M(x_2)$	...	$f_M(x_k)$	...	$f_M(x_N)$
$y \longrightarrow$	M+1	Y_1	Y_2		Y_K		Y_N

$$F[I,K]=f_i(x_k) \qquad I \neq M+1$$
$$F[M+1,K]=y_k \qquad I = M+1$$

Método dos Mínimos Quadrados

- **O algoritmo, assumindo que <u>um array bidimensional A</u>, memoriza a <u>matriz A e o vector b</u> (A:A[1..M, 1..M]), estando este representado na coluna M+1:**

Linhas de A

Colunas de A

$$a_{ij} = f_i . f_j$$
$$b_{ij} = f_i \cdot y$$

```
DO FOR I=1 TO M
    DO FOR J=1 TO M+1
        SOM ← 0
        DO FOR K=1 TO N
                SOM ← SOM + F[I,K] * F[J,K]
        A[I,J] ← SOM
```

Método dos Mínimos Quadrados

- **Caso muito frequente: *f(x)* é um polinómio**
  - **Temos**

$$f_j(x) = x^{j-1}$$

  - **O que significa**

$$f(x) = c_1 + c_2 x + c_3 x_2 + \ldots + c_M x^{m-1}$$

Polinómio grau M-1 em x

  - **Se M = N** → **Polinómio que se ajusta aos pontos** → **INTERPOLAÇÂO**
  - **Se grau inferior** → **Função que se aproxima dos pontos dados (desprezando erros)**

Método dos Mínimos Quadrados

- **Exemplo**

5 pontos: (-1,2) (1,1) (2,1) (3,0) (5,3)

Polinómio de 2º grau: $f(x) = c_1 + c_2 x + c_3 x^2$

$f_1 = (1 \quad 1 \quad 1 \quad 1 \quad 1) = (1 \quad 1 \quad 1 \quad 1 \quad 1)$

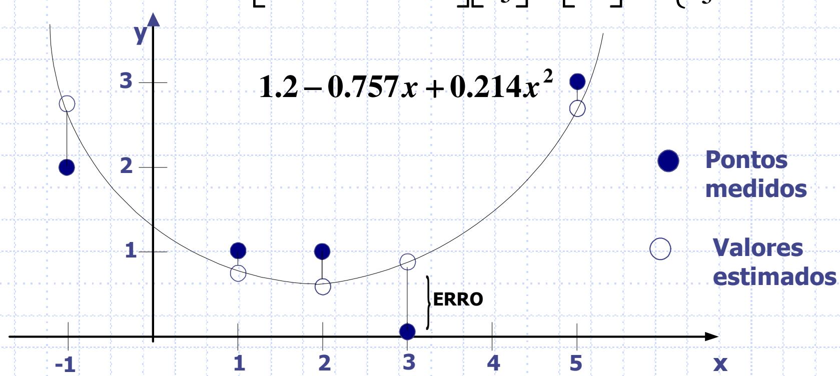
$f_1 = (x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5) = (-1 \quad 1 \quad 2 \quad 3 \quad 5)$

$f_1 = (x_1^2 \quad x_2^2 \quad x_3^2 \quad x_4^2 \quad x_5^2) = (1 \quad 1 \quad 4 \quad 9 \quad 25)$

$y = \qquad = (2 \quad 1 \quad 1 \quad 0 \quad 3)$

Método dos Mínimos Quadrados

$$\begin{bmatrix} 5 & 10 & 40 \\ 10 & 40 & 160 \\ 40 & 160 & 724 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 16 \\ 83 \end{bmatrix} \Rightarrow \begin{cases} c_1 = 1.2 \\ c_2 = -0.757 \\ c_3 = 0.214 \end{cases}$$



Matrizes

- Representação computacional de matrizes esparsas: comparação de diferentes esquemas de representação
- Adição de matrizes
- Multiplicação de matrizes

Matrizes

Representação computacional

Caso geral

Seja A uma matriz com L linhas e C colunas. Computacionalmente podemos representá-la por um array bidimensional A, com o mesmo tamanho, havendo a seguinte correspondência:

aij → **A[i,j]** **; i = 1,...,L ; j=1,...,C**

(Elemento da linha i e coluna j da matriz A)

Sendo C= 1 (A é um vector coluna) temos

a_i → **A[i]** **; i = 1,...,L**

Matrizes

Caso de matrizes esparsas

Esparsidade de uma matriz

- Define-se através do coeficiente seguinte:

$$\text{Coeficiente de esparsidade (CE)} = \frac{\text{Nº de elementos nulos}}{\text{Nº total de elementos}} \times 100\%$$

- As matrizes associadas a sistemas físicos (como redes eléctricas, redes de gás, estruturas mecânicas, entre outros) são caracterizadas por terem, normalmente, coeficientes de esparsidade muito elevados

Matrizes

- Por exemplo, em redes de distribuição de gás de baixa pressão, é vulgar lidar com matrizes com cerca de quatro elementos não nulos (em média) por linha/coluna.

  Já em redes de transportes de energia eléctrica típicos, aquele número passa para cerca de 6 elementos não nulos (em média) por linha/coluna.

Se A é uma Matriz (100x100)

$$CE = \frac{(100x100) - \overbrace{6x100}^{\text{Nº de elementos não nulos}}}{100x100} x\ 100\% = 94\%$$

Matrizes

- **O valor de CE aumentará com a dimensão da matriz, como se vê com o cálculo seguinte, relativo a uma matriz (1000x1000) com 6 elementos não nulos por linha/coluna:**

$$CE = \frac{(1000x1000) - 6x1000}{1000x1000} x\ 100\% = 99.4\%$$

(994.000 zeros!!!)

- **Estes números mostram que haverá enorme vantagem (em termos economia de memória e de eficiência de cálculos) em usar esquemas de representação computacional que explorem a esparsidade das matrizes, sendo a vantagem tanto maior quanto maior for a dimensão das matrizes a operar**

Matrizes

- **De seguida vamos apresentar um conjunto de esquemas – baseados na utilização de listas encadeadas – para o armazenamento computacional de matrizes esparsas**

- **A exposição será ilustrada com o exemplo seguinte:**

$$\begin{bmatrix} 9 & 0 & 0 & 8 & 0 \\ 7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & 6 & 0 & -8 & 0 \end{bmatrix}$$

- **É uma matriz com 4x5=20 elementos, dos quais 14 são nulos, donde: CE=70%**

- **Nos esquemas que vamos apresentar de seguida, apenas serão memorizados os 6 elementos não nulos e as suas posições na matriz A**

Matrizes

Esquema 1

- **Utilização de uma lista simplesmente encadeada, apontada por um apontador A**

- **Cada nodo da lista terá a seguinte estrutura:**

LIN	COL	ELEM	Prox

LIN: linha da matriz a que pertence o elemento

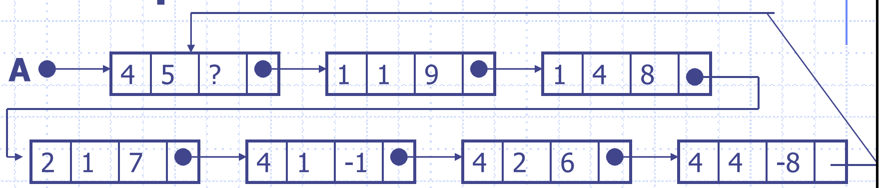
COL: coluna da matriz a que pertence o elemento

ELEM: Elemento da matriz

Prox: Apontador para o nodo seguinte (elemento seguinte não nulo)

Matrizes

➢ Exemplo



- Este esquema tem como grande desvantagem o **não permitir o acesso directo** quer às linhas quer às colunas da matriz, o que é necessário, por exemplo, em operações de multiplicação de matrizes

Matrizes

➢ Esquema 2

- **Utilização de um array de apontadores A[1], A[2], …., A[L], cada um deles apontando para uma lista encadeada que contém os vários elementos não nulos da respectiva linha da matriz**

- **Cada nodo das listas assim construídas terá a seguinte estrutura:**

COL	ELEM	Prox

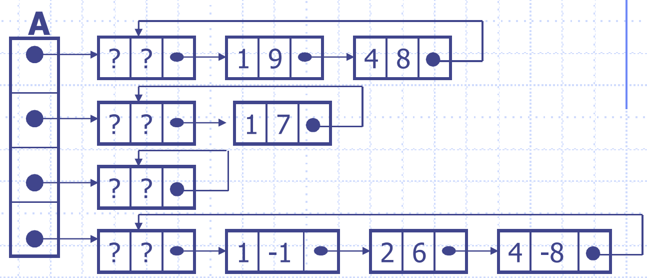
COL: coluna da matriz a que pertence o elemento

ELEM: Elemento da matriz

Prox: Apontador para o nodo seguinte na mesma linha da matriz

Matrizes

Exemplo:



- Este esquema tem a vantagem de **permitir um acesso directo às linhas da matriz**, o que é necessário, por exemplo, quando se pretende multiplicar a matriz A por uma matriz, estando A à esquerda do produto
  Já quando a matriz A estiver à direita de um produto (necessidade de aceder às colunas de A) não será muito prático usar este esquema de representação

Matrizes

- Por outro lado o **tamanho do array** limita o número de linhas que a matriz possa ter
  Para além disso, haverá um grande desperdício no array se a matriz a memorizar tiver **muitas linhas só com zeros**

- Note-se que um esquema, em tudo semelhante a este e com as mesmas vantagens e inconvenientes, poderia ser usado para memorizar uma matriz, **coluna a coluna**, em vez de linha a linha

Matrizes

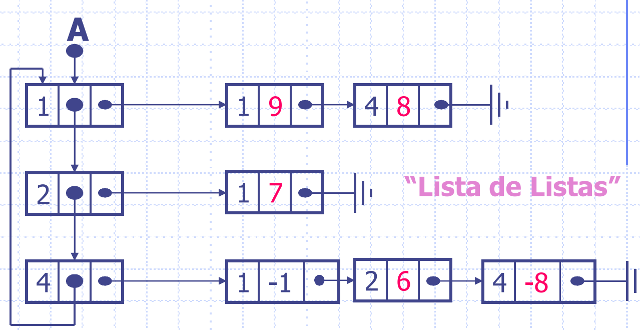
Esquema 3

- **É semelhante ao ESQUEMA 2, excepto no que se refere à utilização do array, que agora é substituído por uma lista simplesmente encadeada, apontada por um apontador A**

- **Deste modo deixa de haver problemas com o nº de linhas da matriz e, por outro lado, não desperdiçamos espaço com a ocupação de informação relativa a linhas que só têm zeros**

Matrizes

Exemplo:



- **Estrutura dos nodos da lista principal (apontada por A):**

LIN	PROX	PRIMCOL

LIN: linha da matriz

PROX: Apontador para o nodo seguinte (linha seguinte não nula)

PRIMCOL: Apontador para o primeiro nodo (primeiro elemento/primeira coluna não nula) da actual linha

Matrizes

- **Tal como no ESQUEMA 2, não há acesso (quase) directo às colunas da matriz. Para isso era necessário memorizar a matriz coluna a coluna, perdendo-se assim o acesso (quase) directo às linhas da matriz!**

Matrizes

Esquema 4

- **São utilizados dois apontadores em cada nodo: um deles aponta para o nodo seguinte da mesma linha da matriz; o outro aponta para o nodo seguinte da mesma coluna da matriz**

- **Com este esquema conseguimos acesso (quase) directo às linhas e colunas da matriz**

Matrizes

- **Estrutura dos nodos:**

LIN	COL	ELEM
PROXLIN	PROXCOL	

LIN: Linha da matriz a que pertence o elemento
COL: Coluna da matriz a que pertence o elemento
ELEM: Elemento da matriz
PROXLIN: Apontador para o nodo seguinte (elemento seguinte/linha seguinte) da actual coluna
PROXCOL: Apontador para o nodo seguinte (elemento seguinte/coluna seguinte) da actual linha

Matrizes

- **Conforme se vê no exemplo que se apresenta de seguida, os campos LIN e COL do primeiro nodo da lista (nodo-cabeça), guardam a informação sobre o nº de linhas e o nº de colunas da matriz**

Matrizes

➢ **Exemplo:**

Matrizes

➢ Esquema 5

- **Estrutura de armazenamento muito usada quando A é uma matriz de coeficientes que deva ser operada para a resolução de um sistema de equações lineares (a ver mais à frente):**

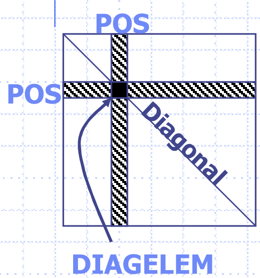
Ax = b (Note-se que A é quadrada)

- **Basicamente, temos uma lista simplesmente encadeada, apontada por um apontador A cujos nodos contêm os elementos da diagonal principal da matriz A e outra informação relevante.**
- **Cada um daqueles nodos conterá ainda um apontador para o início de uma lista em que estão representados os restantes elementos da respectiva linha (ou coluna)**

Matrizes

- **Dois tipos de nodos:**

POS	DIAGELEM	NAO_NULOS	PROX	PRIM



POS: Posição da diagonal principal da matriz

DIAGELEM: Elemento na posição POS da diagonal principal

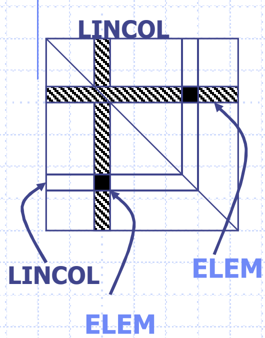
NAO_NULOS: Nº de elementos não nulos na linha (ou coluna)

PROX: Apontador para o nodo seguinte

PRIM: Apontador para o primeiro nodo (primeiro elemento não nulo) na linha (ou coluna) POS

Matrizes

LINCOL	ELEM	PROX



LINCOL: Linha (ou coluna) da matriz

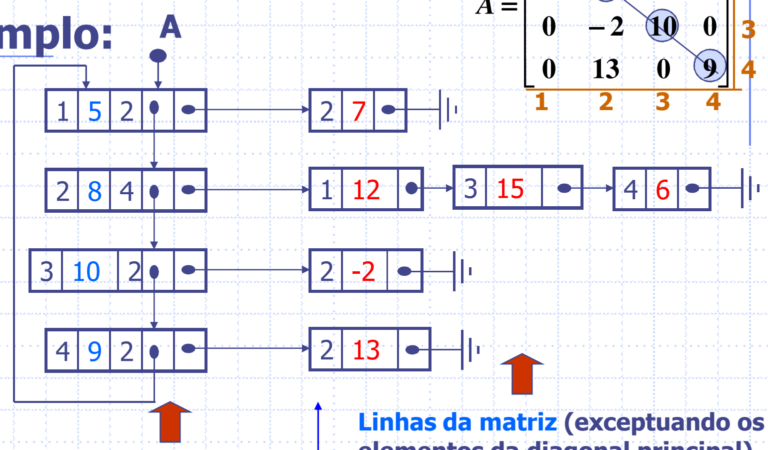
ELEM: Elemento da matriz na linha (ou coluna) LINCOL

PROX: Apontador para o nodo seguinte

Matrizes

Exemplo:

$$
\begin{array}{c c}
A = \begin{bmatrix} 5 & 7 & 0 & 0 \\ 12 & 8 & 15 & 6 \\ 0 & -2 & 10 & 0 \\ 0 & 13 & 0 & 9 \end{bmatrix} & \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} \\
\begin{array}{cccc} \quad 1 & 2 & 3 & 4 \end{array} &
\end{array}
$$



Elementos da **diagonal principal** e outra informação relevante

Linhas da matriz (exceptuando os elementos da diagonal principal)
nº da coluna a que pertence o respectivo elemento

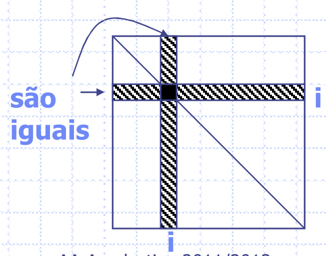
Matrizes

Caso de Matrizes Simétricas (Esparsas)

- **Matriz simétrica: é uma matriz quadrada para a qual se tem:**

$$a_{ij} = a_{ji} ; \forall_{i,j} \text{ com } i \neq j$$

- **Daqui resulta que se tem $A=A^T$, ou seja, uma linha qualquer, i, é igual à coluna i:**



Exemplo:

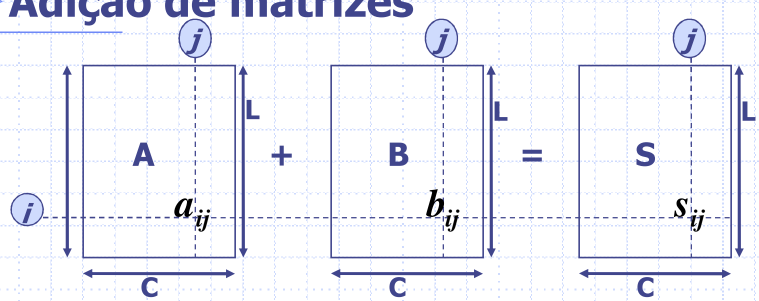
$$A = \begin{bmatrix} 5 & 7 & 0 & 0 \\ 7 & 8 & 0 & 6 \\ 0 & 0 & 10 & 0 \\ 0 & 6 & 0 & 9 \end{bmatrix}$$

Matrizes

- No caso de **matrizes simétricas**, há toda a vantagem em usar estruturas de armazenamento linha a linha (ou coluna a coluna) visto que se tem também acesso imediato às colunas (ou às linhas)
- Este é o caso dos **ESQUEMAS 2, 3 e 5** vistos atrás
- Note-se que em sistemas físicos é relativamente vulgar lidar com matrizes simétricas, e daí a importância desta questão

Matrizes

➢ Adição de matrizes



- É necessário que **A e B** tenham o mesmo tamanho:

$$S_{(L*C)} = A_{(L*C)} + B_{(L*C)}$$

- $$s_{ij} = a_{ij} + b_{ij} \quad ;i = 1,...,L \quad ;j = 1,...,C$$

Matrizes

- **Computacionalmente** $\rightarrow$
  matrizes memorizadas em arrays bidimensionais

$$S[I,J] = A[I,J] + B[I,J]$$

- **O algoritmo da soma é trivial:**

```
ALGORITMO SOMAMAT(A,B,S,L,C)

    DO FOR I=1 TO L
        DO FOR J=1 TO C
                S[I,J] ← A[I,J] + B[I,J]
```


Matrizes

- **Analisemos o algoritmo da soma, admitindo que se tem L=C=N (matrizes quadradas):**

$$T(n) \cong 2n^2$$ **(Ver cálculo rigoroso)**

com

$$n = N$$ **(Nº de linhas/colunas da matriz)**

ou seja, o algoritmo da soma é quadrático, ou ainda, o algoritmo é de ordem n²

Matrizes

- **Uma outra perspectiva, menos comum, considere que:**

com $$T(n) \cong 2n$$
$$n = N^2$$ **(Nº de elementos da matriz)**

- **Cálculo exacto de T(n):**

1ª instrução executada N+1 vezes

2ª instrução executada N(N+1) vezes

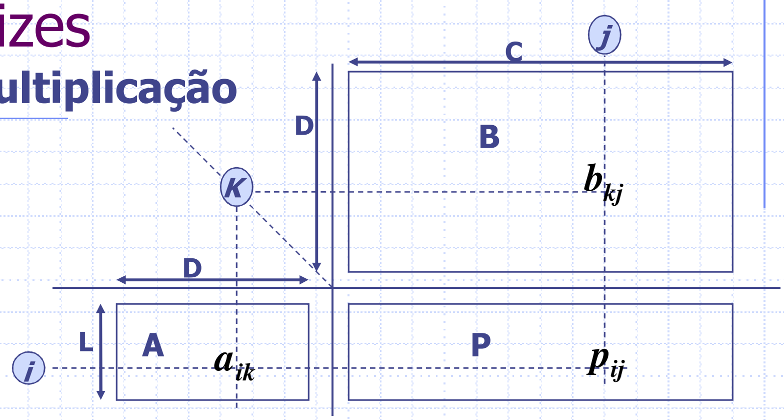
3ª instrução executada N^2 vezes

Total **$2N^2+2N+1$**

Vem ainda $2N^2 + 2N + 1 \cong 2N^2$ **quando N é elevado**

Matrizes

Multiplicação



- **É necessário que o nº de colunas de A seja igual ao nº de linhas de B:**

$$P_{(L*C)} = A_{(L*D)} . B_{(D*C)}$$

- $$p_{ij} = \sum_{k=1}^{D} a_{ik} + b_{kj} \quad ;i = 1,...,L \;\; ;j = 1,...,C$$

Matrizes

- Matrizes memorizadas em arrays bidimensionais

```
ALGORITMO MULTMAT (A,B,P,L,D,C)

    DO FOR I=1 TO L
        DO FOR J=1 TO C
                T ← 0
                DO FOR K=1 TO D
                        T ← T +A[I,K] * B[K,J]
                P[I,J] ← T
```


Matrizes

- **Analisemos o algoritmo da multiplicação, admitindo que se tem L=D=C=N (matrizes quadradas):**

$$T(n) \cong 2n^3$$

(Sugestão: calcular rigorosamente T(n))

com

$$n = N$$

(Nº de linhas/colunas da matriz)

ou seja, o algoritmo da multiplicação é **cúbico**, ou ainda, o algoritmo é de **ordem n^3**

- **Uma outra perspectiva, menos comum, considera que:**

$$T(n) \cong 2n^{3/2}$$

com

$$n = N^2$$

(Nº de elementos da matriz)

Próxima aula

- Resolução de sistemas de equações