

## Introdução à Interpolação Polinomial.

Assuntos: Polinómios.

Aproximação de uma função pelo polinómio de Taylor.

1. Utilize o Matlab no modo imediato e a função *polyval* para determinar:
  - a. o valor de  $P_2(x) = x^2 - 3x + 2$  em  $x=2$ .
  - b. o valor de  $P_2(x) = x^2 - 3x + 2$  para valores entre  $x=0$  e  $x=2$  com incremento 0,5.
2. Escreva um programa em Matlab (Horner.m) para avaliar um polinómio num ponto  $x$  usando o algoritmo de Horner. O programa deverá primeiramente pedir ao utilizador para introduzir o grau do polinómio, os coeficientes e o valor de  $x$ . Como resultado o programa mostrará o valor do polinómio no ponto  $x$ . Compare os resultados obtidos por este programa e os obtidos pela função do Matlab *polyval*.
3. No modo imediato do Matlab e dada as funções  $\sin(x)$ ,  $\cos(x)$ ,  $e^x$ ,  $e^{2x}$ :
  - a. determine o polinómio de Maclaurin de grau 5 que aproxima cada uma das seguintes funções.
  - b. a partir do polinómio obtido em a) e para cada função, construa o gráfico da função e do seu respectivo polinómio. Escolha intervalos adequados a cada função (Por exemplo para o  $\sin(x)$  um intervalo adequado seria  $-\pi \leq x \leq \pi$ ).

Sugestão: Utilize as funções de computação simbólica *taylor*, *sym2poly* e a função *polyval* utilizada em 2.

4. Escreva um programa em Matlab que construa o gráfico duma função e do seu polinómio de Maclaurin de grau 5. O programa deverá pedir ao utilizador para introduzir qual a função e qual o intervalo para  $x$ . Como resultado deverá mostrar qual o polinómio e gerar um gráfico da função e do polinómio.

Sugestão: Utilize as mesmas funções de computação simbólica *taylor*, *sym2poly*, *polyval* que foram utilizadas no exercício 3.

5. Escreva em Matlab a função "*GeraTaylor*" para determinar o polinómio de *Maclaurin* de grau 9 que aproxima uma função. O parâmetro de entrada seria a própria função. O parâmetro de saída seria uma variável simbólica com a expressão do polinómio de Maclaurin de grau 9. Compare os resultados obtidos com o programa do exercício 4.
6. Dada a função  $\sin(x)$ , escreva um programa em MatLab que permita:
  - a. determinar os polinómios de Maclaurin  $P_5(x)$ ,  $P_7(x)$  e  $P_9(x)$ .  
(Sugestão: Utilize a função *GeraTaylor* do exercício 5)
  - b. construir num mesmo gráfico a função  $\sin(x)$  e os polinómios de Maclaurin determinados na linha a) no intervalo  $-\pi \leq x \leq \pi$
  - c. visualizar uma tabela com 4 colunas que contenha os valores para  $x$ ,  $\sin(x)$ ,  $P_9(x)$ ,  $E_9(x) = \sin(x) - P_9(x)$  avaliados em 10 pontos equidistantes do intervalo  $[-1, 1]$ . Mostrar os dados no formato *long*.
  - d. construir o gráfico com o erro de truncatura  $E_9(x)$  para  $-1 \leq x \leq 1$ .

Ainda relativamente ao exercício 6, mostre analiticamente que se  $|x| \leq 1$  então verifica-se que:  $|E_9(x)| < 1/10! \leq 2.75574 \times 10^{-7}$

Compare este resultado com os dados da tabela da linha c) e com o gráfico do erro da linha d).

*Nota: a resolução deste exercício está no ficheiro *TaylorSin.m**

7. Dada a função  $\cos(x)$ , escreva um programa em MatLab que permita:
- determinar os polinômios de Maclaurin  $P_4(x)$ ,  $P_6(x)$  e  $P_8(x)$ .  
(Sugestão: Utilize a função `GeraTaylor` do exercício 5)
  - desenhar num mesmo gráfico a função  $\cos(x)$  e os polinômios de Maclaurin determinados na alinha a) no intervalo  $-\pi \leq x \leq \pi$ .
  - visualizar uma tabela com 4 colunas que contenham os valores para  $x$ ,  $\cos(x)$ ,  $P_8(x)$ ,  $E_8(x) = \cos(x) - P_8(x)$  avaliados em 10 pontos equidistantes do intervalo  $[-1, 1]$ . Mostrar os dados no formato *long*.
  - construir o gráfico com o erro de truncatura  $E_8(x)$  para  $-1 \leq x \leq 1$ .

Ainda relativamente ao exercício 7, mostre analiticamente que se  $|x| \leq 1$  então

$$\text{verifica-se que: } |E_8(x)| < 1/9! \leq 2.75574 \times 10^{-6}$$

Compare este resultado com os dados da tabela da alinha c) e com o gráfico do erro da alinha d).

## Notas sobre as funções do Matlab:

### 1. Polinômios

(Estas funções encontram-se no **Interpolation and Polynomials toolbox**)

- **função POLYVAL:** avalia um polinômio num ponto

Seja um polinômio  $P_n(x) = c(1)x^n + c(2)x^{n-1} + \dots + c(n)x + c(n+1)$

Se  $C$  é um vector cujos elementos são os coeficientes  $c(1)$ ,  $c(2)$ , ...,  $c(n+1)$  (nesta ordem!!!) então  $y = \text{polyval}(C, x)$  é o valor do polinômio avaliado em  $x$ .

**Exemplo 1:** Determinar o valor de  $P_2(x) = x^2 - 3x + 2$  em  $x=2$ .

```
» C = [1 -3 2] % os coeficientes são armazenados na lista(vector) C
```

```
C =
    1    -3     2
```

```
» x = 2
```

```
x =
     2
```

```
» y = polyval(C,x)
```

```
y =
     0 % o valor do polinômio em x=2 é igual a 0
```

**Exemplo 2:** Determinar o valor de  $P_2(x) = x^2 - 3x + 2$  para valores entre  $x=0$  e  $x=2$  com incremento 0,5.

```
» for x=0:0.5:3, % são avaliados valores desde x=0 até 2 com incremento de 0.5
```

```
    disp([x,polyval(C,x)]),
end
```

```

0      2
0.5000  0.7500
1      0
1.5000 -0.2500
2      0
2.5000  0.7500
3      2
```

□ **função ROOTS:** Determina as raízes de um polinómio

**Exemplo 13.2:** Determinar as raízes de  $p(x) = x^4 - 10x^3 + 35x^2 - 50x + 24$

```
» C = [1 -10 35 -50 24]
```

```
C =
```

```
1 -10 35 -50 24
```

```
» roots(C)
```

```
ans =
```

```
4.0000
```

```
3.0000
```

```
2.0000
```

```
1.0000
```

□ **poly(A)** : função inversa de roots. Dadas as raízes, determina o polinómio.

Exemplo:

```
» z=[1;2] % é introduzido um vector com as raízes do polinómio
```

```
z =
```

```
1
```

```
2
```

```
» p=poly(z)
```

```
p =
```

```
1 -3 2
```

```
% é obtido o polinómio de grau 2:  $x^2-3x+2$ 
```

Se realizamos agora a função inversa **roots** obtém-se:

```
» roots(p)
```

```
ans =
```

```
% são obtidas as raízes do polinómio
```

```
2
```

```
1
```

□ **conv(A)** : determina o produto de 2 polinómios

Exemplo: Determine o produto de dois polinómios de 1º grau p e q , cujas raízes são 2 e 3 respectivamente.

```
» p=poly(2)
```

```
p =
```

```
1 -2
```

```
% 2 é raiz do polinómio  $x-2$ 
```

```
» q=poly(3)
```

```
q =
```

```
1 -3
```

```
% 3 é raiz do polinómio  $x-3$ 
```

```
» conv(p,q)
```

```
ans =
```

```
1 -5
```

```
6 % o polinómio resultado é  $x^2-5x+6$ 
```

## 2. Computação Simbólica em MatLab

(Estas funções encontram-se no **Symbolic Math Toolbox**)

O toolbox para computação simbólica utiliza objectos simbólicos definidos através da função **sym**

□ **SYM** → constrói números simbólicos, variáveis e objectos

» **S = sym(A)**

constrói um objecto S da classe "sym" a partir da expressão A

- se A é string, o resultado é uma variável.
- se A é um número escalar ou uma matriz, o resultado é a representação simbólica deste(s) número(s)

Por exemplo:

```
» x = sym('x')           % cria a variável simbólica com o nome 'x' e armazena
                        % o resultado em x
» x = sym('x', 'real') % assume que x é real
```

□ **SYMS** → define vários objectos simbólicos

Por exemplo, os seguintes comandos:

```
» sym('a'); sym('t'); sym('x'); sym('y')
```

podem ser combinados num único comando

```
» syms a t x y % define 4 variáveis simbólicas
```

```
» syms % lista os objectos simbólicos no workspace
```

As variáveis simbólicas podem ser utilizadas como argumento das funções:

Por exemplo:

```
» r = x^2 + y^2
» theta = atan(y/x)
» atan(y/x)
```

```
» i=10
```

```
» e = exp(i*pi*t)
```

e =

```
exp(10*pi*t) % a variável i não é simbólica, por isso fica substituída na
                expressão
```

Por vezes é aconselhável utilizar as funções "simple" or "simplify" para transformar expressões em formatos mais simples:

Por exemplo:

```
» f = cos(x)^2 + sin(x)^2
```

f =

```
cos(x)^2+sin(x)^2
```

```
» f = simple(f)
```

f =

```
1
```

Podem ser calculadas as derivadas e integrais de uma função utilizando as funções "diff" e "int":

```
» diff(x^3)
```

```
ans =  
3*x^2
```

```
» int(x^3)
```

```
ans =  
1/4*x^4
```

Também podem ser criadas constantes simbólicas com a função **sym**. O argumento pode ser uma representação simbólica de um valor numérico. Por exemplo, comandos como **pi = sym('pi')** e **delta = sym('1/10')** criam números simbólicos os quais evitam os erros de aproximações inerentes aos valores de pi e 1/10. O pi criado com a função sym substitui temporariamente a função do Matlab construída com o mesmo nome

```
» pi = sym('pi')
```

```
pi =  
pi
```

```
» delta = sym('1/10')
```

```
delta =  
1/10
```

```
» s = sym('sqrt(2)')
```

```
s =  
sqrt(2)
```

#### □ Lista de funções para computação simbólica:

##### ▪ Simplificação

- ◆ **simplify** → simplifica uma expressão
- ◆ **simple** → re-escreve uma expressão numa forma mais simples
- ◆ **expand** → expande uma expressão
- ◆ **collect** → re-escreve a expressão como um polinómio
- ◆ **subs** → substitui a variável x por um valor

##### ▪ Calculus

- ◆ **diff** → diferencial
- ◆ **int** → integral
- ◆ **limit** → limite
- ◆ **taylor** → a série de Taylor (polinómio de Maclaurin de grau 5)

##### ▪ Conversões

- ◆ **poly2sym** → constrói um polinómio simbólico a partir dum vector com os seus coeficientes
- ◆ **sym2poly** → constrói um vector com os coeficientes dum polinómio a partir dum polinómio simbólico

##### ▪ Solução de equações

- ◆ **solve** → solução simbólica de equações
- ◆ **dsolve** → solução simbólica de equações diferenciais
- ◆ **finverse** → a função inversa
- ◆ **compose** → a função composta

**Exemplos / Exercícios resolvidos:**

2.1. Determine a série de Taylor para a função  $\sin(x)$

» **syms x**

» **taylor(sin(x))**

ans =

$x - 1/6*x^3 + 1/120*x^5$

2.2. Determine a derivada da função  $1 + \ln(x)$ . Avalie-a no ponto  $x=0.25$

» **syms x**

» **f=1+log(x)**

f =

$1 + \log(x)$

» **f1=diff(f)**

f1 =

$1/x$

» **subs(f1,0.5)**      % esta função substitui na função f1 a variável x pelo valor 0.5

ans =

2

2.3. Determine o polinómio de Maclaurin de grau 5 que aproxima a função  $e^x$ . Construa o gráfico da função e do polinómio de Maclaurin no intervalo  $-10 \leq x \leq 10$  com incremento 0.5.

```

syms x; % a variável x é definida como simbólica
t=taylor(exp(x)) % determina o polinómio de Taylor simbólico
pretty(t);
C=sym2poly(t) % determina o vector com os coeficientes do polinómio
x = -10:0.05:10; % um vector x com pontos equidistantes
fun = 'exp(x)'; % a função pode ser armazenada numa variável como string
y = eval(fun); % a função eval avalia uma função definida como string
p=polyval(C,x); % avalia o polinómio de Taylor
figure(1); clf; % mostra a janela de gráficos
hold on;
title('Comparação da função exp(x) e o polinómio de Taylor');
% desenhar as linhas dos eixos
ymin=min(y); % são calculados os valores min e max para as ordenadas
if min(p)<ymin
    ymin=min(p);
end
ymax=max(y);
if max(p)<ymax
    ymax=max(p);
end
axis([-10 10 ymin ymax]); % define os eixos
plot([-10 10],[0 0], 'b',[0 0],[ymin ymax], 'b'); % desenha os eixos
plot(x,y, 'g',x,p, 'b'); % desenha os dois gráficos
grid;
hold off;

```

