

Métodos Numéricos

Slides MATLAB - aulas práticas

Teresa Monteiro

Departamento de Produção e Sistemas

Escola de Engenharia

Universidade do Minho

tm@dps.uminho.pt

<http://www.norg.uminho.pt/tm/>

Ano lectivo 2014/15

Como usar estes slides?

- são básicos, servem de tópicos de apoio às aulas Práticas
- o aluno poderá consultar outras fontes
- para explorar uma função deve usar o comando `help`:

```
>> help <nome da função>
```

Formato

```
>> format long
>> pi
ans = 3.14159265358979
>> format short % formato por defeito (mostra 4 cd)
>> pi
ans = 3.1416
```

Função úteis

- `ones(n)` -> matriz de 1s quadrada de ordem n
- `zeros(n)` -> matriz nula de ordem n
- `eye(n)` -> matriz identidade de ordem n
- `()'` -> transposta da matriz
- `det ()` -> determinante da matriz
- `rank()` -> característica da matriz
- `inv()` -> inversa da matriz
- `diag()` -> diagonal da matriz
- `triu()`, `tril()` -> matriz triangular superior/inferior da matriz
- `norm(,1)`, `norm(,inf)` -> normas (1 e [inf](#)) da matriz
- `A \ b` -> resolução do sistema linear

Sistemas lineares

```
>> A=[1 -3 4; 2 5 -2; 3 8 10] %introdução da matriz A (3 x 3)
A =
     1     -3      4
     2      5     -2
     3      8     10
>> rank(A) %característica de A
ans =3
>> A' %transposta de A
ans =
     1      2      3
    -3      5      8
     4     -2     10
>> det(A) %determinante de A
ans =
    148
>> inv(A) %inversa de A
ans =
    0.4459    0.4189   -0.0946
   -0.1757   -0.0135    0.0676
    0.0068   -0.1149    0.0743
```

Sistemas lineares

```
>>diag(diag(A)) %matriz diagonal com a diagonal de A
ans =
     1     0     0
     0     5     0
     0     0    10

>> triu(A) %matriz triangular superior de A
ans =
     1    -3     4
     0     5    -2
     0     0    10

>> tril(A) %matriz triangular inferior de A
ans =
     1     0     0
     2     5     0
     3     8    10

>> b=[1; 3; -6] %vector coluna b
b =
     1
     3
    -6
```

```
>> A\b %sistema linear Ax=b: possível e determinado
ans =
    2.2703
   -0.6216
   -0.7838
>> norm(A,1) %norma 1 de A
ans = 16
>> norm(A,inf) %norma infinita de A
ans = 21
```

Característica da matriz

```
>> A=[2 4; 4 8]
A =
     2     4
     4     8
>> rank(A)
ans = 1
>> b=[3 6]
b =
     3     6
>> Amp=[A b']
Amp =
     2     4     3
     4     8     6
>> rank(Amp)
ans = 1
>> A\b'
Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND = 2.467162e-017.
ans = 0.5000
      0.5000 % sistema possível indeterminado
```

Característica da matriz

```
>> A=[2 4; 4 8]
A =
     2     4
     4     8
>> rank(A)
ans = 1
>> c=[ 1 1]
c =
     1     1
>> Amp=[A c']
Amp =
     2     4     1
     4     8     1
>> rank(Amp)
ans = 2
>> A\c'
Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND = 2.467162e-017.
ans = 1.0e+015 *
     1.1259
    -0.5629    % Sistema impossível
```


Interpolação numérica

Função `polyfit`

Forma de utilizar a função `polyfit`:

```
P = polyfit(X,Y,N)
yaux=polyval(P,xaux)
```

P é um vector linha de comprimento $N+1$ que contém os coeficientes de P :

$$P(1) * X^N + P(2) * X^{(N-1)} + ... + P(N) * X + P(N+1)$$

Calcula o polinómio P de grau N que interpola os valores (x,y) sendo x e y vectores de tamanho $N+1$

Muito importante

Para que haja interpolação usando um polinómio de grau N o número de pontos tem de ser igual a $N+1$

Exemplo: polyfit

x	-2	0	1	5	7
y	88	10	13	1005	4021

Considere a tabela com 5 pontos e calcule o polinómio interpolador de grau 4 para estimar o valor em $x = 2$

Exemplo: polyfit

```
>> x=[-2 0 1 5 7]
x =
    -2     0     1     5     7
>> y=[88 10 13 1005 4021]
y =
     88     10     13    1005    4021
>> p=polyfit(x,y,4)
p =
     2.0000    -3.0000     5.0000    -1.0000    10.0000
>> y=polyval(p,2)
y =
    36.0000
% p(2)=36
```

O polinómio é $p(x) = 2x^4 - 3x^3 + 5x^2 - x + 10$

Interpolação numérica segmentada

Função `spline`

Formas de utilizar a função `spline`:

```
PP = SPLINE (X, Y)
```

```
YY = SPLINE (X, Y, XX)
```

Fornece uma forma polinomial segmentada do tipo spline cúbica interpoladora para os dados (vectors X e Y)

Se Y contém dois valores a mais do que X , o primeiro e o último são usados para a curvatura inicial e final da spline cúbica, *ie*, há hipótese de forçar as curvaturas nos extremos

Se XX for um vector, YY é o vector correspondente obtido pela spline

Interpolação numérica segmentada

Exemplo

x	0	2	3	5	8	12
y	0	15	13	25	28	42

Considere a tabela para construir a spline cúbica que interpola todos pontos. Estime o valor em $x = 4$

Interpolação numérica segmentada

Exemplo

```
>> x=[0 2 3 5 8 12];
>> y=[0 15 13 25 28 42];
>> pp=spline(x,y);
>> pp.coefs
ans =
    2.0488   -13.4105    26.1260         0
    2.0488    -1.1179    -2.9309    15.0000
   -1.2591     5.0284     0.9797    13.0000
    0.2883    -2.5263     5.9839    25.0000
    0.2883     0.0688    -1.3887    28.0000

%construção dos 5 segmentos
S1(x) = 2.0488*(x-0)^3 + -13.4105*(x-0)^2 + 26.1260*(x-0) + 0
S2(x) = 2.0488*(x-2)^3 + -1.1179*(x-2)^2 + -2.9309*(x-2) + 15.0000
S3(x) = -1.2591*(x-3)^3 + 5.0284*(x-3)^2 + 0.9797*(x-3) + 13.0000
S4(x) = 0.2883*(x-5)^3 + -2.5263*(x-5)^2 + 5.9839*(x-5) + 25.0000
S5(x) = 0.2883*(x-8)^3 + 0.0688*(x-8)^2 + -1.3887*(x-8) + 28.0000

% estimacão do valor em x=4
>> xx=spline(x,y,4)
xx=
    17.7490
```

Interpolação numérica segmentada

Exemplo

```
>> x=[0 2 3 5 8 12];
>> y=[0 15 13 25 28 42];
% forçar a curvatura nula nos extremos "[0 y 0]"
>> pp=spline(x,[0 y 0]);
>> pp.coefs
ans =
    -3.2593    10.2685         0         0
     5.3240    -9.2870     1.9630    15.0000
    -1.6828     6.6850    -0.6390    13.0000
     0.5919    -3.4116     5.9079    25.0000
    -0.3488     1.9154     1.4192    28.0000
%construção dos 5 segmentos
S1(x) = -3.2593*(x-0)^3 + 10.2685*(x-0)^2
S2(x) = 5.3240*(x-2)^3 -9.2870*(x-2)^2 + 1.9630*(x-2) + 15.0000
S3(x) = -1.6828*(x-3)^3 + 6.6850*(x-3)^2 -0.6390*(x-3) + 13.0000
S4(x) = 0.5919*(x-5)^3 -3.4116*(x-5)^2 + 5.9079*(x-5) + 25.0000
S5(x) = -0.3488*(x-8)^3 + 1.9154*(x-8)^2 + 1.4192*(x-8) + 28.0000
% estimacão do valor em x=4
>> xx=spline(x,[0 y 0],4)
xx =
    17.3633
>> xx=spline(x,[0 y 0], [0 1 2 3 4 5 6 7 8 12])
xx =
0     7.0093    15.0000    13.0000    17.3633    25.0000    28.0881    27.9043    28.0000    42.0000
% reparar que há interpolação nos nós da spline (0,0), (2,15), (3,13), (5,25), (8,28), (12,42)
```

Função: trapz, quad, quadl

Calculam aproximações ao integral usando:

- trapz: a regra do trapézio
- quad: quadratura de Simpson adaptativa
- quadl: quadratura de Lobatto adaptativa

Exemplo: trapz

x	0.0	0.3	0.6	0.8	1.0	1.2	1.8	2.4	3.0	3.6	4.2
$f(x)$	4.0	3.9	3.7	3.5	3.3	2.9	2.5	2.0	1.25	0.75	0.0

Calcule uma aproximação a $\int_0^{4.2} f(x)dx$

```
>>x=[0.0 0.3 0.6 0.8 1.0 1.2 1.8 2.4 3.0 3.6 4.2];  
>> y= [4.0 3.9 3.7 3.5 3.3 2.9 2.5 2.0 1.25 0.75 0.0];  
>> z = trapz(x,y)  
z =  
    9.1150
```

Exemplo: quad, quadl

Calcule $\int_1^2 \frac{e^x}{x} dx$ usando métodos numéricos.

m-file e comandos

```
function [F]=teste(x)
F=exp(x)./x; % ./ operador divisão "ponto a ponto" - operador para vectores

>> quad('teste',1,2)
ans =
    3.05911655903093
>> quad('teste',1,2,1.e-20)% estipula a tolerância do erro absoluto=1.e-20
(valor por defeito=1.e-6)
ans =
    3.05911653964595
>> [z,nf]=quad('teste',1,2,1.0e-20)% mostra o n° de cálculos da função em nf
z =
    3.05911653964595
nf = 5093
>> quadl('teste',1,2)
ans =
    3.05911654523322
NOTA: quad e quadl usam operadores para vectores .*, ./ e .^ na definição
da função a integrar
```

Solução de uma equação não linear tipo polinomial

Função `roots`

Forma de utilizar a função `roots`:

```
roots(C)
```

Calcula as raízes do polinómio cujos coeficientes são os elementos do vector C .

Se C tiver $N + 1$ componentes o polinómio é

$$C(1) * X^N + C(2) * X^{N-1} \dots + C(N) * X + C(N + 1)$$

Solução de uma equação não linear tipo **polinomial**

Exemplo

Calcular os zeros do polinómio $4x^5 + 3x^3 - 2x^2 - 8x + 3$.

comandos

```
>> C=[4 0 3 -2 -8 3];  
  
>> roots(C)  
ans =  
    -0.1468 + 1.3818i  
    -0.1468 - 1.3818i  
    -1.0695  
     1.0000  
     0.3631  
% 2 raízes imaginárias e 3 raízes reais
```

Solução de uma equação não linear

Função `fzero`

Formas de utilizar a função `fzero`:

```
X = FZERO(FUN,X0)
```

```
...
```

```
[X,FVAL,EXITFLAG,OUTPUT] = FZERO(FUN,X0,OPTIONS)
```

Tenta encontrar um zero da função escalar não linear `FUN` perto de `X0`, sendo `X0` um valor escalar

Para visualizar toda informação fazer na janela de comandos:

```
» help fzero
```

Solução de uma equação não linear

Exemplo

Calcular o zero da função $7(2 - 0.9^x) - 10 = 0$ perto de 6.

m-file e comando

```
function [F] = teste(x)
F = [7*(2-0.9^(x))-10];
end
>> [x,y,exitflag,output] = fzero('teste', 6)
x = 5.3114 % zero
y = -1.7764e-015 % valor da função em x: y=f(x)
exitflag =
    1 % encontra o zero x
output = ...
    iterations: 6 % n° iterações
    funcCount: 18 % n° de cálculos da função
    ...
```

Solução de uma equação não linear

Função `fsolve`

Formas de utilizar a função `fsolve`:

```
X = FSOLVE(FUN,X0)
```

```
...
```

```
[X,FVAL,EXITFLAG,OUTPUT] = FSOLVE(FUN,X0,OPTIONS)
```

Resolve um sistema de equações não lineares de várias variáveis

Atenção

A função `fsolve` é vocacionada para um sistema de equações não lineares, no entanto pode ser utilizada também para uma só equação

Optimset

OPTIMSET cria/altera a estrutura de opções de otimização OPTIONS

```
>>OPTIONS = OPTIMSET('PARAM1',VALUE1,'PARAM2',VALUE2,...)  
>> optimset fsolve % ver os parâmetros por defeito de fsolve
```


Solução de uma equação não linear

Exemplo

Calcula a raiz de $f(x) = 10 - 20 * (e^{(-0.2*x)} - e^{(-0.75*x)}) - 5$ próxima de 1 considerando um erro relativo em x e em f menor do que 10^{-2}

m-file e comandos

```
function [F,d] = teste(x)
F = [10-20*(exp(-0.2*x)-exp(-0.75*x))-5];
if nargin>1
    d=[4*exp(-0.2*x)-15*exp(-0.75*x)];
end
>>x0=[1]
>>options=optimset('Jacobian','on','maxIter',4,'TolX',1.0e-2,'
>>[xsol,fsol,exitflag,output]=fsolve('teste',x0,options)
xsol = 0.6023
fsol = 1.1350e-004
exitflag = 1
output = ... iterations: 4 ...
```

Sistemas de equações não lineares

Função: fsolve

exemplo

$$\begin{cases} (x_1^4 + 0.06823x_1) - (x_2^4 + 0.05848x_2) - 0.01509 &= 0 \\ (x_1^4 + 0.05848x_1) - (2x_2^4 + 0.11696x_2) &= 0 \end{cases}$$

Resolva o sistema utilizando para aproximação inicial $(0.30, 0.30)$, fornecendo a informação acerca das derivadas

Sistemas de equações não lineares

Exemplo: fsolve

```
function [F,d] = teste(x)
F(1) = [(x(1)^4+0.06823*x(1))-(x(2)^4+0.05848*x(2))-0.01509];
F(2) = [(x(1)^4+0.05848*x(1))-(2*x(2)^4+0.11696*x(2))];
% fornecendo as primeiras derivadas
if nargin>1
    d = [4*x(1)^3+0.06823 -4*x(2)^3-0.05848; 4*x(1)^3+0.05848 -8*x(2)^3-0.11696];
end
>> x0=[0.30 0.30]
>> options=optimset('Jacobian','on')
%para que a rotina use as primeiras derivadas fornecidas na m-file
>> [xsol,fsol,exitflag,output]=fsolve('teste',x0,options)
xsol =
    0.2928    0.1879
fsol =
    1.0e-005 *
   -0.1429   -0.2858
exitflag =
     1
output =
    iterations: 3
    ...
```

Aproximação dos mínimos quadrados - modelo polinomial

Função: `polyval`, `polyfit`

Forma de utilizar a função `polyfit`:

```
P = polyfit(X,Y,N)
yaux=polyval(P,iaux)
```

P é um vector linha de comprimento $N + 1$ que contém os coeficientes de P :

$$P(1) * X^N + P(2) * X^{(N-1)} + ... + P(N) * X + P(N + 1)$$

Calcula o polinómio P de grau N que ajusta, no sentido dos **mínimos quadrados**, os dados Y

Muito importante

No ajuste dos mínimos quadrados o número de pontos da amostra (M) tem de ser superior ao grau do polinómio (N) - **não é obrigatório que $M=N+1$** (esta condição é para o caso da interpolação)

Aproximação dos mínimos quadrados

Exemplo polyfit

x	-50	-20	10	70	100	120
y	0.125	0.128	0.134	0.144	0.150	0.155

Use a técnica dos mínimos quadrados para calcular um modelo linear e um modelo quadrático que estime y como função de x

Aproximação dos mínimos quadrados - modelo polinomial

Exemplo polyfit

```
>> x=[-50 -20 10 70 100 120]
>> y=[0.125 0.128 0.134 0.144 0.150 0.155]
>> [P,s]=polyfit(x,y,1)
P =      0.000177      0.132537
s =      ...

normr: 0.002285
```

O modelo polinomial é: $p_1(x) = 0.000177x + 0.132537$.

O somatório do quadrado do erro é

$$S = \text{normr}^2 = 0.002285^2 = 0.000005.$$

```
>> [P,s]=polyfit(x,y,2)
P =      0.000000      0.000155      0.131725
s =      ...

normr: 0.001124
```

O modelo polinomial é:

$p_2(x) = 0.000000x^2 + 0.000155x + 0.131725$. O somatório do quadrado do erro é $S = \text{normr}^2 = 0.001124^2 = 0.000001$.

Aproximação dos mínimos quadrados - modelo não polinomial

Função: `lsqcurvefit`

LSQCURVEFIT resolve um problema de **mínimos quadrados não linear** Formas de utilizar a função `lsqcurvefit`:

```
X=LSQCURVEFIT(FUN,X0,XDATA,YDATA)
```

```
...
```

```
[X,RESNORM,RESIDUAL,EXITFLAG,OUTPUT,...]=LSQCURVEFIT(FUN,X0,XDATA,YDATA,...)
```

- X é a solução, RESNORM=S (**soma do quadrado dos resíduos**)
- RESIDUAL é o vector resíduo
- EXITFLAG apresenta a condição de saída
- FUN - m-file com o modelo
- X0 aproximação inicial
- XDATA,YDATA são os vectores dos dados (x,y)

Aproximação dos mínimos quadrados - modelo não polinomial

Exemplo: lscurvefit

x	1	1.5	3	4	6	10
y	-2.2	-2.1	-1.6	0.5	0.5	-0.5

Aproxime os dados da tabela no sentido dos mínimos quadrados por um modelo do tipo

$$\sin(c_1 x) + \frac{c_2}{x} + e^{c_3 x}.$$

Considere para aproximação inicial dos parâmetros o vector
 $[1 \ 1 \ 1]$

Aproximação dos mínimos quadrados - modelo não polinomial

m-file e comandos

```
function f=teste(c,x) %o vector c contem os 3 parâmetros
f=sin(c(1).*x)-c(2)./x+exp(c(3).*x);

>> x=[1; 1.5; 3; 4; 6; 10];
>> y=[-2.2; -2.1; -1.6; 0.5; 0.5; -0.5];
>> [C,RESNORM,RESIDUAL,EXITFLAG,OUTPUT]=lsqcurvefit('teste',[1 1 1],x,y)
>> % também se pode colocar options com o comando optimset
>> [C,RESNORM,RESIDUAL,EXITFLAG,OUTPUT]=lsqcurvefit('teste',[1 1 1],x,y,[],[],options)
```

Mínimos quadrados - modelo não polinomial

```
C= 16.0526    2.2002   -1.0438
```

```
RESNORM = 0.0070
```

```
RESIDUAL =
```

```
0.0141
```

```
-0.0272
```

```
0.0509
```

```
-0.0531
```

```
0.0142
```

```
-0.0202
```

```
EXITFLAG =
```

```
1
```

```
OUTPUT =
```

```
...
```

```
iterations: 27
```

```
funcCount: 112
```

```
...
```

O modelo encontrado é

$$\sin(16.0526 x) + \frac{2.2002}{x} + e^{-1.0438 x}$$