Aplicações Informáticas para Engenharia

Matlab – Módulo Avançado

Sérgio Manuel Ângelo da Cruz

2007

DEPARTAMENTO DE ENGENHARIA
ELECTROTÉCNICA E DE COMPUTADORES
UNIVERSIDADE DE COIMBRA Faculdade de Ciências e Tecnologia

· UC ·
DEEC

Estruturas

- Em matlab, tal como em muitas outras linguagens, torna-se importante dispormos de estruturas capazes de armazenar dados de diferentes tipos
- Por exemplo, se pretendermos criar uma ficha individual de um aluno, é importante ter a possibilidade de dispormos de um campo onde podemos inserir o seu nome, morada, idade, etc. Claramente necessitamos de poder inserir strings nos dois primeiros campos e um valor numérico no terceiro campo
- Pegando no exemplo anterior, poder-se-ia fazer o seguinte em matlab para criar a tal ficha do aluno:

```
>> Fichaaluno.nome='António da Silva'
Fichaaluno =
    nome: 'António da Silva'
```

Estruturas

```
>> Fichaaluno.morada='Coimbra'
Fichaaluno =
        nome: 'António da Silva'
    morada: 'Coimbra'

>> Fichaaluno.idade=22
Fichaaluno =
        nome: 'António da Silva'
    morada: 'Coimbra'
        idade: 22
```

- No exemplo anterior, o matlab vai criar automaticamente uma estrutura com o nome Fichaaluno, com três campos ('fields'):nome, morada, idade

- A qualquer momento podemos aceder ao conteúdo de cada campo escrevendo o nome da estrutura, seguida de um ponto final e por fim o nome do campo que pretendemos consultar. Um exemplo:

Estruturas

```
>> Fichaaluno.idade
ans =
    22
```

- Escrevendo apenas o nome da estrutura, o matlab irá mostrar todos os campos que a compõem:

```
>> Fichaaluno
Fichaaluno =
       nome: 'António da Silva'
    morada: 'Coimbra'
       idade: 22
```

- Usando o comando **fieldnames**, podemos saber quais os campos que compõem uma determinada estrutura:

```
>> fieldnames(Fichaaluno)
ans =
    'nome'
    'morada'
    'idade'
```

Estruturas

- Para acrescentarmos mais fichas de outros alunos, por forma a criar um vector de estruturas, basta colocarmos um índice à frente do nome da estrutura. Continuando com o exemplo anterior pode-se fazer o seguinte:

```
>> Fichaaluno(2).nome='Bárbara Guimarães';
>> Fichaaluno(2).morada='Lisboa';
>> Fichaaluno(2).idade=35;
>> Fichaaluno(2)
ans =
        nome: 'Bárbara Guimarães'
      morada: 'Lisboa'
       idade: 35
```

Estruturas

```
>> Fichaaluno
Fichaaluno =
1x2 struct array with fields:
    nome
    morada
    idade
```

- Pode-se criar uma estrutura de uma única vez com o comando **struct**. O mesmo exemplo anterior:

```
>> Fichaaluno=struct('nome','António Ferreira','morada','Coimbra','idade',23)
Fichaaluno =
       nome: 'António Ferreira'
    morada: 'Coimbra‘
        idade: 23
```

Toolbox de Matemática Simbólica - Introdução

- Para além de cálculo numérico, o matlab também permite efectuar cálculo simbólico. Para esse efeito, existe uma *toolbox* de matemática simbólica ("*Symbolic Math Toolbox*")

- Com esta toolbox podem-se efectuar cálculos de diferentes naturezas, designadamente: cálculo propriamente dito (diferenciação, integração, determinação de limites, somatórios, série de Taylor, etc.); algebra linear (inversa de uma matriz, determinantes, valores próprios, etc.); simplificação de expressões algébricas; obtenção de soluções analíticas de equações algébricas e diferenciais; Transformadas de Laplace, Z e de Fourier, etc., etc.

- Para efectuar cálculo simbólico, o matlab define um novo tipo de dado, designado por *objecto simbólico*

- Para indicar ao matlab que uma dada variável (ou expressão) deve ser encarada como objecto simbólico usa-se o comando **sym**

Toolbox de Matemática Simbólica - Introdução

- Observe as diferenças entre os dois exemplos a seguir apresentados:

```
>> a=sqrt(2)
a =
   1.4142

>> b=sqrt(sym(2))
 b =
2^(1/2)
```

- Neste último caso, foi indicado ao matlab que ele devia considerar o número 2 como uma constante simbólica pelo que ele devolve o resultado sem efectuar qualquer cálculo numérico (***b*** será encarado em todos os cálculos posteriores como uma constante simbólica)

- Havendo interesse, pode-se a qualquer momento converter ***b*** para o valor numérico correspondente:

Toolbox de Matemática Simbólica - Introdução

```
>> double(b)
ans =
   1.4142
```

- Outro exemplo ilustrativo das diferenças entre cálculo numérico e simbólico:

```
>> sym(2)/sym(3)+sym(1)/sym(5)
 ans =
 13/15
>> 2/3+1/5
ans =
   0.8667
```

- Considere agora, a título de exemplo, que pretende simplicar uma dada expressão designatória. O comando **simplify** pode ajudar nessa tarefa. Se essa expressão for (x^2-1)/(x-1), podemos simplificá-la da seguinte forma:

Toolbox de Matemática Simbólica - Introdução

```
>> a=sym('x^2-1'), b=sym('x-1')
 a =
 x^2-1
  b =
 x-1
 >> simplify(a/b)
 ans =
 x+1
```

- Frequentemente, temos necessidade de dizer ao matlab que diversas variáveis devem ser consideradas como simbólicas. Nesse caso, torna-se mais fácil e prático usar o comando **syms** em vez de **sym**

- Como exemplo, vamos assumir que se pretende estudar a função quadrática ***f***, dada pela expressão a*x^2+b*x+c. Neste caso, devemos em primeiro lugar indicar ao matlab que ***a***, ***b***, ***c*** e ***x*** devem ser tratadas como variáveis simbólicas e só depois definir a função ***f***:

Toolbox de Matemática Simbólica - Introdução

```
>> syms a b c x
>> f=sym('a*x^2+b*x+c')  % em alternativa pode-se fazer f=a*x^2+b*x+c
 f =
 a*x^2+b*x+c
```

- Pode-se agora manipular *f* da forma desejada, tendo a garantia que o matlab vai tratar todas as variáveis como simbólicas. (Nota a respeito de syms: podemos, para além de enumerar as variáveis simbólicas, indicar qual o tipo dessas variáveis: reais (***real***), positivas (***positive***), etc. Isto é efectuado colocando um parâmetro a seguir à última variável definida. Um exemplo: **syms x real**)

- Uma vez definida *f*, pode-se agora criar uma nova função em que a sua expressão designatória é igual à da função *f* após substituirmos *a*, *b* e *c* por 4, 5 e 8, respectivamemente. Essa operação poderá ser efectuada usando o comando **subs**:

```
>> g=subs(f,[a,b,c],[4,5,8])
 g =
 4*x^2+5*x+8
```

Toolbox de Matemática Simbólica

- O comando **findsym** permite ver quais as variáveis simbólicas presentes numa dada expressão. Por exemplo, em *f* existem 4 variáveis simbólicas mas em *g* existe apenas uma (x):

```
>> findsym(g)
ans =
x
```

- O comando **solve** é bastante útil pois permite resolver equações (ou sistemas de equações) algébricas. Como exemplo, pretende-se achar as soluções da equação x^2-10*x+21=0. Pode-se então fazer:

```
>> solve('x^2-10*x+21')
 ans =
  7
  3
```

- Mais exemplos demonstrativos do uso da função **solve**:

Toolbox de Matemática Simbólica

```
>> equacaosegundograu='a*x^2+b*x+c=0‘;
>> solve(equacaosegundograu)
 ans =
  1/2/a*(-b+(b^2-4*a*c)^(1/2))
  1/2/a*(-b-(b^2-4*a*c)^(1/2))

>> B=solve(equacaosegundograu,'a=1','b=-3','x=10') % calcula o valor de c que
    %fará com que a condição seja verdadeira
B =
    a: [1x1 sym]
    b: [1x1 sym]
    c: [1x1 sym]
    x: [1x1 sym]
>> B.c                                   % vê qual o valor de c armazenado na estrutura B
 ans =
 -70
```

Toolbox de Matemática Simbólica

- Mais um exemplo do uso do comando **solve**:

```
>> S = solve('x^2 + x*y + y = 3','x^2 - 4*x + 3 = 0')
S =
    x: [2x1 sym]
    y: [2x1 sym]
>> S.x
 ans =
  1
 3
  >> S.y
 ans =
    1
 -3/2
```

- Neste último exemplo, o resultado do sistema de equações é armazenado numa estrutura, a qual contém as duas soluções para x e y

Toolbox de Matemática Simbólica

- Podemos ver alguns dos resultados anteriores com um aspecto mais amigável usando o comando **pretty**. Um exemplo:

```
>> h=solve(equacaosegundograu);
>> pretty(h)

             [            2        1/2]
             [    -b + (b  - 4 a c)   ]
             [1/2 --------------------]
             [              a         ]
             [                        ]
             [            2        1/2]
             [     -b - (b  - 4 a c)  ]
             [1/2 --------------------]
             [              a         ]
```

Matemática Simbólica - Derivação

- A toolbox de matemática simbólica disponibiliza um conjunto de funções de cálculo, entre as quais se destacam a derivação, a qual pode ser efectuada usando o comando **diff**:

```
>> f=sym('3*x^3+4*x^2-5');
>> diff(f)
 ans =
 9*x^2+8*x

% ===== Código alternativo =========
>> syms x
>> f=3*x^3+4*x^2-5
 f =
 3*x^3+4*x^2-5
  >> diff(f)
 ans =
 9*x^2+8*x
```

Matemática Simbólica – Derivação / Integração

- Para calcular derivadas de ordem superior à primeira pode-se usar o mesmo comando mas com mais um parâmetro de entrada:

```
>> diff(f,3)          %  calcula a terceira derivada da função f
 ans =
 18
```

- Para calcular integrais indefinidos pode-se usar o comando **int**:

```
>> int(f)             % calcula a primitiva de f
 ans =
 3/4*x^4+4/3*x^3-5*x
```

- Outro exemplo:

```
>> syms a b c x
>> h=a*x^2+b*x+c;
>> int(h,b)          % calcula a primitiva de h considerando que b é a variável independente
 ans =
 a*x^2*b+1/2*b^2*x+c*b
```

Matemática Simbólica – Integração / Limites

- Ainda outro exemplo, agora com integrais definidos:

```
>> f=sym(sin(x))
f =
sin(x)
>> int(f,0,pi)
ans =
2
```

- Outra funcionalidade disponibilizada por esta *toolbox* é o cálculo de limites, usando o comando **limit**. A sintaxe deste comando é **limit(f,x,x0)** (calcula o limite da função *f* quando *x* tende para *x0*). Alguns exemplos:

```
>> syms x
>> limit(sin(x)/x,x,0)
ans =
1
```

Matemática Simbólica – Limites / Somatórios

```
>> limit(x/abs(x),x,0,'left')          % calcula o limite à esquerda
ans =
-1
>> limit(x/abs(x),x,0,'right')         % calcula o limite à direita
ans =
1
```

- Pode-se também calcular somatórios, neste caso com o comando **symsum**. Como exemplo, vamos supor que se pretende obter a soma de todos os elementos de uma dada série: soma=1+1/2^2+1/3^2+....1/n^2+... Este cálculo pode ser efectuado da seguinte forma:

```
>> syms k
>> soma=symsum(1/k^2,1,inf)
soma =
1/6*pi^2
```

Matemática Simbólica – Série de Taylor

- Pode-se ainda decompor uma dada função em série de Taylor (ou Maclaurin), usando o comando **taylor**. Alguns exemplos:

```
>> syms x
>> g=exp(x*sin(x))
 g =
 exp(x*sin(x))
 >> t=taylor(g,12,2);  % decompõe g em série de Taylor, obtendo os 12 primeiros
    termos não nulos, em torno do ponto x=2.

>> syms x
>> f=1/(5+4*cos(x));
>> M=taylor(f,8)    % obtém os termos da série de Maclaurin até (mas não
    incluindo) o de ordem 8
 M =
 1/9+2/81*x^2+5/1458*x^4+49/131220*x^6
```

Matemática Simbólica – Equações Diferenciais

- Para resolver equações diferenciais de qualquer ordem, usa-se o comando **dsolve**. Alguns exemplos:

```
>> dsolve('Dy=1+y^2')      % calcula a solução da equação y'=1+y^2
ans =
tan(t+C1)                  % C1 é uma constante

>> y=dsolve('Dy=1+y^2','y(0)=1') % resolve a equação com uma condição inicial
y =
tan(t+1/4*pi)
```

- Nos exemplos anteriores, a letra '***D***' indica ao matlab que se trata da primeira derivada (por defeito a variável independente é *t*). Colocando um número inteiro após ***D*** dá indicação ao matlab de uma derivada de ordem superior. Como exemplo, vamos admitir que se pretende resolver a equação diferencial $d^{(3)}y/dx^3=y$, com as condições iniciais y(0)=1; y'(0)=-1; y''(0)=pi. A solução desta equação pode ser obtida da seguinte forma:

Matemática Simbólica – Equações Diferenciais / Algebra

```
>> y=dsolve('D3y=y','y(0)=1','Dy(0)=-1','D2y(0)=pi','x')
 y =
 1/3*pi*exp(x)-1/3*(1+pi)*3^(1/2)*exp(-1/2*x)*sin(1/2*3^(1/2)*x)+(-1/3*pi+1)*exp(-
    1/2*x)*cos(1/2*3^(1/2)*x)
```

- Para calcular os valores e/ou vectores próprios de uma matriz usa-se o comando **eig**. Um exemplo:

```
>> A=[1 3 5;2 4 6;7 8 9];
>> eig(A)
ans =
   16.4340
   -2.4340
   -0.0000
```

- Outra funcionalidade proporcionada pela *toobox* de matemática simbólica é a manipulação de expressões simbólicas (polinomiais ou de outro tipo), ao permitir factorizar ou expandir as referidas expressões

Matemática Simbólica – Factorização e...

- Neste contexto, dois comandos muito úteis são **expand** e **factor**. Alguns exemplos da sua utilidade:

```
>> syms x
>> f=(x-3)^3+(x+1)^2-3
 f =
 (x-3)^3+(x+1)^2-3
 >> expand(f)
 ans =
 x^3-8*x^2+29*x-29
>> factor(x^3+x^2-21*x-45)
 ans =
 (x-5)*(x+3)^2
```

- Consulte o help e veja ainda a utilidade de **simple**, **collect** e **combine**

Transformada de Laplace

- Para calcular a Transformada de Laplace ou Transforma de Laplace Inversa usamos os comandos **laplace** e **ilaplace**, respectivamente. Alguns exemplos:

```
>> syms t
>> f=5*exp(-5*t)
 f =
 5*exp(-5*t)
 >> laplace(f)     % calcula a Transformada de Laplace de f
 ans =
 5/(s+5)
>> g=t^2+5*t-6;
>> laplace(g)      % calcula a Transformada de Laplace de g
 ans =
 2/s^3+5/s^2-6/s
```

Transformada de Laplace, Z e Fourier

```
>> syms s
>> ilaplace(1/sqrt(s))          % calcula a Transformada de Laplace Inversa
 ans =
 1/(pi*t)^(1/2)
>> h=(2*s+2)/(s^2+7*s+12)  % define h
 h =
 (2*s+2)/(s^2+7*s+12)
 >> ilaplace(h)                 % calcula a Transformada de Laplace Inversa de h(s)
 ans =
 -4*exp(-3*t)+6*exp(-4*t)
```

- De forma análoga à Transformada de Laplace, a Transformada Z e a Transformada de Fourier podem ser calculadas por intermédio dos comandos **ztrans** e **fourier**, respectivamente

Transformada Z

- A Transformada Z (unilateral) de uma função ***f*** é definida como:

$$f(n),\ n \text{ é variável independente} \quad \Rightarrow \quad Z[f] = \sum_{n=0}^{\infty} f(n)\, z^{-n}$$

- Um exemplo concreto:

```
>> syms n
>> f=n^4
 f =
 n^4
>> sol=ztrans(f)
 sol =
 z*(z^3+11*z^2+11*z+1)/(z-1)^5
>> pretty(sol)
                    3         2
               z (z  + 11 z  + 11 z + 1)
                -------------------------
                                 5
                          (z - 1)
```

Transformada de Fourier

- A Transformada de Fourier de uma função ***f*** é definida como:

$$f(x),\ x \text{ é variável independente} \quad \Rightarrow \quad F(\omega) = \int_{-\infty}^{\infty} f(x)\, e^{-i\omega x}\, dx$$

- Dois exemplos concretos:

```
>> syms x
>> f=exp(-x^2)
f =
exp(-x^2)
 >> fourier(f)
ans =
pi^(1/2)*exp(-1/4*w^2)

>> h=x*exp(-abs(x))
h =
x*exp(-abs(x))
>> fourier(h)
ans =
-4*i/(1+w^2)^2*w
```

Transformadas Inversas

- De forma análoga à Transformada de Laplace, os comandos **iztrans** e **ifourier** permitem calcular as Transformadas Inversas de Z e Fourier, respectivamente

- Muitas outras funções da *toolbox* de Matemática Simbólica não foram aqui abordadas pelo que se recomenda uma consulta ao manual da mesma