Equações Diferenciais em Matlab

- O matlab apenas calcula **soluções numéricas de sistemas de equações diferenciais de primeira ordem**. Para resolver equações diferenciais de ordem superior deve convertê-las, por intermédio de uma mudança de variáveis, em equações diferenciais de 1ª ordem

- Antes de definir o sistema de equações no matlab tem de, previamente, manipular manualmente tal sistema de modo a que ele fique na forma **dV/dt=f(t,V)**, onde V é um vector com as funções do sistema. Por outras palavras, o sistema tem de ficar num formato do tipo dy1/dt =...; dy2/dt =...., etc)

Tratamento de Dados de Entrada

- O comando **isempty** permite verificar se uma dada variável está "vazia". Por exemplo, se pedir ao utilizador um número e ele apenas premir a tecla <Enter>, a variável associada ficará vazia e o comando acima referido devolve o valor lógico Verdadeiro. Um exemplo:

```
A=input('Introduza uma matriz:');
if isempty(A)
    disp('Não introduziu qualquer matriz')
else
    disp('A inversa da matriz que acabou de introduzir é:')
    disp(inv(A))
end
```

- Prima F1 e faça uma busca por **is**. Seleccione a primeira linha dos resultados da busca e observe o conjunto de funções que o matlab disponibiliza para verificar o tipo de dados que o utilizador introduziu

Funções na Janela de Comando

- Pode definir funções na própria janela de comando (ou ficheiro **.m**) usando o comando **inline**. Três exemplos:

```
f=inline('0.1*exp(x)+sin(x)-5-x')     % Define a função f
fplot(f,[0 5]);                       % Gráfico 2D de f(x)

f=inline('x^2+3*y')                   % Define a função de duas variáveis independentes
z=f(3,4)                              % z toma o valor da função nos pontos (3,4)

f=inline('sin(x1)+x2','x1','x2')      % Define uma função f de variáveis
                                      % independentes x1 e x2
```

Interpolações em Matlab

- O matlab possui comandos que permitem efectuar interpolações de uma dada função, sendo conhecido apenas um conjunto finito de pontos (abcissas) e respectivas imagens (ordenadas)

- Existem diferentes métodos para efectuar uma interpolação. O matlab, por defeito, efectua uma interpolação linear mas o utilizador pode escolher um outro método se assim o entender

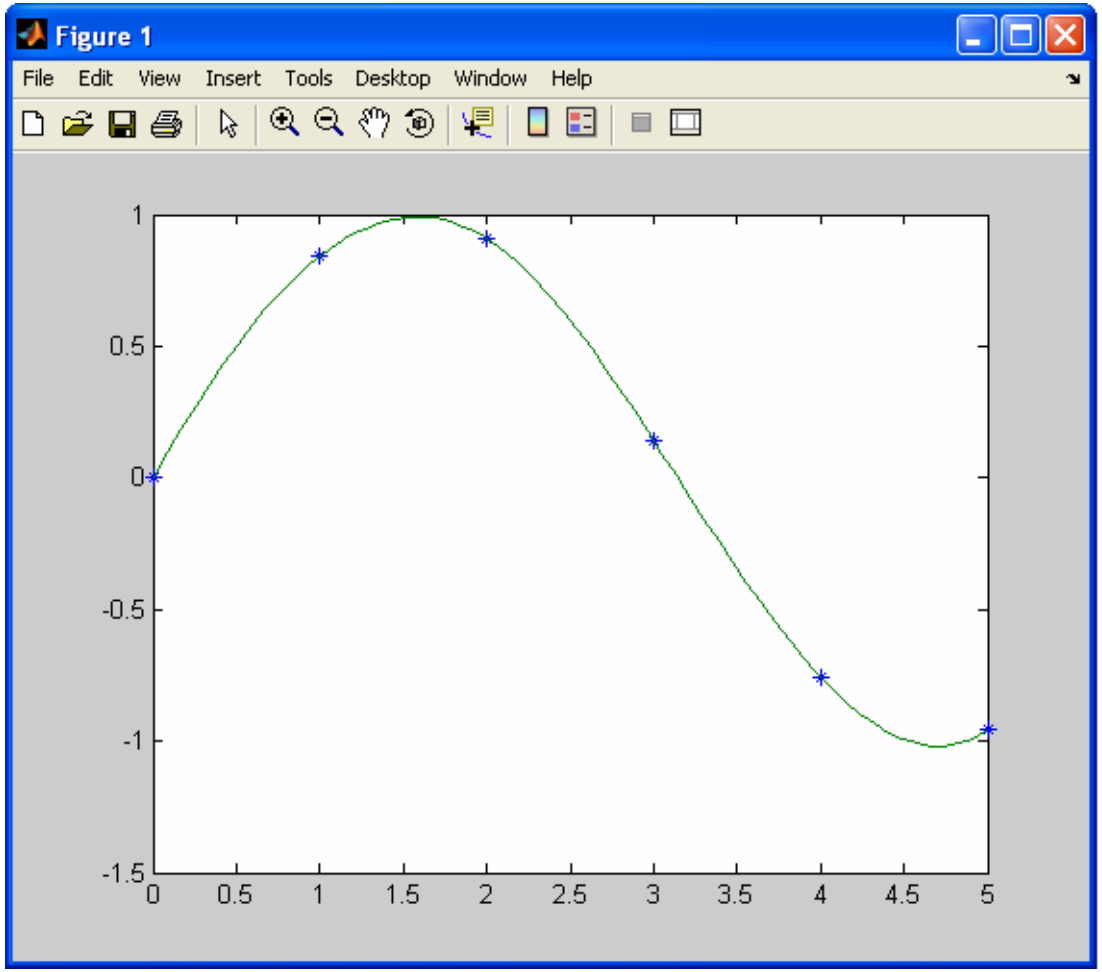
- Os comandos **interp1**, **interp2**, **interp3** e **interpn** permitem obter interpolações de uma dada função a uma dimensão, duas dimensões, três dimensões e *n* dimensões, respectivamente

- Os elementos dos vectores que constituem os parâmetros de entrada das funções interpoladoras devem ser dispostos de forma monótona (isto é, sempre de forma crescente ou decrescente), excepto para **interp1**. No entanto, tais pontos podem estar espaçados de forma irregular

Interpolações em Matlab

- Um exemplo de interpolação a 1 dimensão:

```
x=0:5; y=sin(x);
xi=linspace(0,5,60);
% interpolação com splines cúbicos
yi=interp1(x,y,xi,'spline');
% gráfico para aferir a qualidade da
    interpolação
plot(x,y,'*',xi,yi)
```



Interpolações em Matlab

- Um exemplo de interpolação linear a 2 dimensões:

```
% Desenha um gráfico 3D da função z=f(x,y)=x^2+y^2
xe = -2:0.1:2; ye = -3:0.1:3; % repare que o espaçamento dos pontos é de 0.1 !!
[Xe,Ye]=meshgrid(xe,ye);
Ze=Xe.^2+Ye.^2;
subplot(3,1,1), mesh(Xe,Ye,Ze) % representa o gráfico da função f 'exacta'

%====== VAMOS COMEÇAR A INTERPOLAÇÃO AQUI =====
x= -2:1:2; y= -3:1:3;          % Admite-se que só se conhece a função nestes pontos
[X,Y]=meshgrid(x,y);
Z=X.^2+Y.^2;                   % Valores da função a usar na interpolação
subplot(3,1,2), mesh(X,Y,Z)    % representa o gráfico da função f 'conhecida'
Zi=interp2(x,y,Z,Xe,Ye);       % obtém-se uma interpolação linear da função f
subplot(3,1,3), mesh(Xe,Ye,Zi) % representa o gráfico da função f interpolada
```

Interpolações em Matlab

- Uma forma ligeiramente diferente de interpolar a 2 dimensões (talvez menos intuitiva pois x é um vector linha e y um vector coluna):

```
x = -2:2;
y = [-2:2]';
z = (sin(y).*exp(-y.^2))*(sin(x).*exp(-x^2));
xi = linspace(-2,2,100);
yi = linspace(-2,2,100)';
zi = interp2(x,y,z,xi,yi);
mesh(xi,yi,zi)
```

Tratamento de Erros – Try e Catch

- Quando surge um erro num programa executado em matlab, ele sinaliza tal erro e aborta a execução do programa (ficheiro **.m**)

- Se pretender "apanhar" o referido erro e ter uma acção correctiva do mesmo (tratamento adequado dos dados), sem abortar a execução do ficheiro em causa, tem ao seu dispor os comandos **try** e **catch**

- A sintaxe no uso destes comandos é a seguinte

```
try
    % fragmento de código 1 que o matlab vai tentar executar sempre
catch
    % fragmento de código que o matlab vai executar se houver um erro no
    fragmento de código anterior (situado entre try e catch)
end   % fim dos comandos try-catch
```

Tratamento de Erros – Try e Catch

- Um exemplo muito simples no uso de try e catch:

```
clc
A=zeros(4,3)
B=eye(4,4)
try
    C=A*B;
    disp('O produto de A por B é igual a:')
    disp(C)
catch
    C=NaN;
    disp('As matrizes A e B não podem ser multiplicadas')
end
```

Gravação/Leitura de Dados

- Para além de efectuar cálculos em matlab, torna-se importante armazenar os resultados obtidos em ficheiro e lê-los a partir de ficheiros previamente gravados

- Foi visto anteriormente que o matlab dispõe dos comandos **save** e **load** para gravar/ler num/de ficheiro (por defeito grava ficheiros com extensão **.mat**, próprios do matlab) as variáveis do ambiente de trabalho. Usando o parâmetro **-ascii** a gravação será efectuada num ficheiro de texto (Ex: **>> save -ascii c:\teste.txt a b** %grava as variáveis a e b num ficheiro de texto)

- Ao gravar as variáveis em ficheiros **.mat**, tem a vantagem de poder ler apenas algumas variáveis do ficheiro. O mesmo exemplo anterior: **>> save teste a b; clear all; load teste a;** %neste caso são gravadas as variáveis *a* e *b* mas apenas *a* é lida do ficheiro

- Para além dos comandos **load** e **save**, o matlab possui outros comandos por forma a exportar dados para serem usados por outras ferramentas de análise e vice-versa. O matlab suporta os tipos de dados mais frequentes (ficheiros de texto, binários, Excel (xls), etc)

Gravação/Leitura de Dados

- O comando **fprintf** permite formatar dados e gravá-los em ficheiro (ou afixá-los no ecrã). Sintaxe: **fprintf (ficheiro_ID, formatosaida, variavelsaida)**

- Este comando faz o seguinte:
  - grava a(s) *variavelsaida* no ficheiro apontado por *ficheiro_ID*, de acordo com o *formatosaida* especificado
  - omitindo *ficheiro_ID*, o comando afixa o resultado no ecrã!!
  - *formatosaida* é uma string com informação acerca da formatação a dar à(s) variável(is) a gravar. Envolve o uso do caracter **%** e permite especificar um conjunto de aspectos que serão apresentados de seguida. É permitido o uso de caracteres especiais (de controlo), tais como **\n**, **\r**, **\t**, **\b**, **\f** (nova linha, <enter>, tab, espaço para trás e nova página, respectivamente)

- Obtenha mais informações sobre este comando digitando **help fprintf**

Gravação/Leitura de Dados

- Alguns exemplos no uso de **fprintf**:

```
>> x=3; fprintf('A raiz quadrada de %g é %8.6f \n',x,sqrt(x));
A raiz quadrada de 3 é 1.732051
```

- No exemplo anterior, **%g** indica que a primeira variável deve ser afixada usando uma notação o mais compacta possível e **%8.6f** indica que a segunda variável deve ser afixada num formato de vírgula fixa com um número máximo de oito caracteres (incluindo a vírgula!!), e 6 casas decimais

- Repare que como temos de afixar dois números (x e sqrt(x)) é conveniente usaram-se dois códigos de conversão na string de formatação. Experimente omitir por exemplo o primeiro código de conversão (%g) e observe os resultados obtidos

- Apresenta-se de seguida uma tabela com alguns códigos de conversão vulgarmente usados:

Gravação/Leitura de Dados

Código	Descrição da conversão
%s	Converte string de caracteres
%d	Número em notação decimal (com sinal)
%f	Número em notação de vírgula fixa
%e	Notação científica (%E também é admissível)
%g	Notação mais compacta entre os resultados obtidos com %e e %f (omitindo zeros sem significado)

Gravação/Leitura de Dados

- Para além do tipo de conversão a efectuar (%d, %f, etc), pode-se ainda especificar o número máximo de caracteres das variáveis (w) bem como o número de caracteres à direita da vírgula (p):

  %w.pf

  %w.pe

- Alguns exemplos de formatação de números com o comando **fprintf** (experimente os comandos pois por vezes são afixados espaços em branco que não são visíveis na tabela):

Número	%8.4f	%12.3e	%10g	%8d
2	2.0000	2.000e+000	2	2
sqrt(2)	1.4142	1.414e+000	1.41421	1.414214e+000
sqrt(2e-11)	0.0000	4.472e-006	4.47214e-006	4.472136e-006
sqrt(2e11)	447213.5955	4.472e+005	447214	4.472136e+005

Gravação/Leitura de Dados

- Exemplos no uso de **fprintf**:

```
>> x=1:4; y=sqrt(x); fprintf('%9.4f\n',y)
   1.0000
   1.4142
   1.7321
   2.0000
```

- No exemplo anterior, %9.4f é usado para formatar todos os elementos do vector y. No entanto, por vezes esta funcionalidade pode originar resultados um tanto ou quanto invulgares. Um exemplo:

```
>> clc; x=1:4; y=sqrt(x); fprintf('y = %9.4f\t',y)
y =    1.0000  y =    1.4142        y =    1.7321        y =    2.0000        >>
```

Gravação/Leitura de Dados

- Um outro exemplo, agora com matrizes:

```
>> clc; A=[1 2 3;4 5 6;7 8 9], fprintf('%8.2f %8.3f %8.4f\n',A)
A =
     1     2     3
     4     5     6
     7     8     9

    1.00   4.000   7.0000
    2.00   5.000   8.0000
    3.00   6.000   9.0000
```

- No exemplo anterior, repare que a instrução vai percorrer a matriz por colunas e não por linhas, daí que o resultado de **fprintf** seja diferente da matriz original! Tenha cuidado ao usar fprintf e confira sempre os resultados

Gravação/Leitura de Dados

- Mais um exemplo, agora para afixar uma tabela no ecrã:

```
clc
etiquetas=char('pequena','média','grande','enorme'); % vector coluna de strings
largura=[5;5;10;15];
altura=[5;8;15;25];
profundidade=[15;15;20;35];
volume=largura.*altura.*profundidade/10000; % volume em metros cúbicos
fprintf('\n      Tamanhos das caixas de equipamentos\n\n');
fprintf('Tamanho\t\tlargura\t  altura\tprofundidade\tvolume\n');
fprintf('\t\t\t   (cm)\t\t(cm)\t\t(cm)\t\t(m^3)\n');
for i=1:length(largura)
fprintf('%-8s  %8d %8d    %8d       %9.5f\n',...
etiquetas(i,:),largura(i),altura(i),profundidade(i),volume(i))
end
```

Gravação/Leitura de Dados

- Vamos agora gravar num ficheiro os resultados criados pelo comando **fprintf**. Para esse efeito, para além deste comando tem também de usar **fopen** e **fclose** para abrir/criar e fechar o ficheiro. Um exemplo:

```
f=input('Introduza uma temperatura em graus Fahrenheit[F]:');
c=5/9*(f-32);
fprintf('%5.2f graus Fahrenheit correspondem a %5.2f graus centigrados.\n',f,c)
ficheiro_ID=fopen('c:\dados.txt', 'w'); % cria ou abre o ficheiro dados.txt
fprintf(ficheiro_ID, '%5.2f graus Fahrenheit correspondem a %5.2f graus
        centigrados.\n',f,c);
fclose(ficheiro_ID); % fecha o ficheiro
```

Gravação/Leitura de Dados

- O comando **dlmwrite** permite gravar ficheiros de texto em que os dados numéricos estão separados por um determinado delimitador (um *tab* por exemplo). A sintaxe é a seguinte: **dlmwrite('ficheiro',Matriz,'delimitador')**. Um exemplo:

```
M = magic(3);
dlmwrite('c:\meuficheiro.txt', [M*5 M/5], '\t')
```

- O comando **dlmread** é idêntico mas para o processo de leitura. Sintaxe: **dlmread('ficheiro','delimitador')**. Omitindo o delimitador, a função detecta-o automaticamente. Consulte o help para ver as diferenças entre especificar ou não o delimitador. Um exemplo:

```
A=dlmread('c:\meuficheiro.txt','\t') % Lê os números contidos no ficheiro,
    considerando que os mesmos estão separados por tabs
```

- **dlmwrite** e **dlmread** apenas trabalham com dados numéricos

Gravação/Leitura de Dados

- Os comandos **xlswrite** e **xlsread** permitem gravar e ler ficheiros no formato usado pelo Microsoft Excel (extensão **.xls**). Um exemplo:

```
A=rand(4,3);
xlswrite('c:\teste.xls',A)  % Grava a matriz A em ficheiro
B=xlsread('c:\teste.xls'); % Lê o ficheiro e armazena o resultado em B
```

- Os comandos **fread** e **fwrite** permitem ler e gravar ficheiros binários. Dois exemplos:

```
identificador=fopen('c:\teste.bin','w');
A=rand(4,3);
fwrite(identificador,A,'float64');  % Grava os elementos da matriz  A (percorre os
    elementos da matriz por colunas)
fclose(identificador);
```

Gravação/Leitura de Dados

```
ident=fopen('c:\teste.bin','r')
B=fread(ident,'float64');          % Lê os dados e armazena o resultado em B
fclose(ident);
B=reshape(B,4,3);  % ...porque os dados estão armazenados num vector coluna
% em alternativa poderia ter-se lido e formatado B ao mesmo tempo:
B=fread(ident,[4,3],'float64');
```

- O comando **importdata** permite ler qualquer tipo de ficheiro (incluindo som, imagens, etc.). Faça **help importdata** para mais informações