AULAS PRÁTICAS DE MATLAB
(FS - 2002/2003)

- **O que é o MATLAB?**

  É um software que ao nível mais básico, pode funcionar apenas como uma calculadora, mas que a um nível mais elevado pode tornar-se uma linguagem de programação poderosa para aplicações de ordem técnica.

  O MATLAB permite cálculo, visualização e programação.

  O MATLAB está estruturado para utilizar um vasto conjunto de ficheiros que são os "M-files".

  O MATLAB possui os seus próprios "M-files", mas outros poderão ser criados pelo utilizador e, em conjunto, torna-se possível criar poderosas ferramentas de cálculo e gráficas.

  Os "M-file"s são de dois tipos: Os script-files e os function-files. Os script-files contém uma sucessão de comandos MATLAB que se destinam a resolver determinado problema.
  Os function-files destinam-se à criação de funções, o que permite alargar o vasto leque das já existentes no MATLAB.

  O MATLAB dispõe ainda do Symbolic Math Toolbox onde é possível realizar operações de cálculo integral e diferencial, álgebra linear, soluções algébicas de equações, etc.

- **Como iniciar e terminar o MATLAB?**

  Click duas vezes com o botão do lado esquerdo do rato sobre o ícone do MATLAB 

  ou, se preferir, abra o menu *start* e escolha *programs –> MATLAB*.

  Para terminar a sua sessão, após verificar se guardou tudo o que lhe interessava, abra o menu *File* e escolha E*xit*.

- **Utilizando o MATLAB**

  Ao longo da utilização do MATLAB vão surgir diversos conjuntos de janelas, umas com gráficos outras com "M-files", outras com o *Help* (dispõe do *Help* "online"), etc.

  É conveniente fazer uma gestão adequada dessas janelas, apagando umas, minimizando outras, caso contrário corre o risco de "se perder".

  A janela de abertura do MATLAB está dividida em duas partes: "**Current Directoy"**, onde poderá ver e escolher os ficheiros que estão em cada directório do seu computador e a **"Command Window"** onde poderá escrever os diversos comandos de MATLAB a executar.

  A 1ª operação deve ser sempre colocar-se no directório que contém os seus ficheiros que será o seu "**Working Directory"**. Caso não exista deve criá-lo.

  Para criar "M-files" deverá ir ao Menu *File* do MATLAB e escolher a opção *New –> "M-file"* . Pode então digitar o seu programa e guardar em seguida no seu directório de trabalho.

  A selecção do directório pode ser feita escrevendo o "Path" na **Command Window** à frente da "prompt" do MATLAB (Ex.: >> cd a:\trabfs ou >> cd c:\matlab\work\ze) ou na janela do Current Directory abrindo o botão ... e escolhendo o directório pretendido.

  Um "M-file" criado pelo utilizador, bem como eventuais alterações de um "M-file" já criado só se tornam efectivas se tiver feito *save* dele. Para o fazer deverá ir ao Menu *File* e escolher uma dos opções: *save* ou *save As* e dar-lhe um nome.

  Para executar um "M-file" basta por exemplo escrever o nome na linha de comandos do MATLAB. Pode também usar o botão de "run".

O nome deve ser sugestivo do seu conteúdo ou do projecto em que está inserido, de modo a poder ser identificado rapidamente.
Não se devem dar aos "M-files" criados pelo utilizador nomes de "M-files" ou palavras reservadas do MATLAB pois incorre-se no grave risco de vir a existir um conflito.

- **Sinais das principais operações básicas**

+	Adição (entre escalares e entre arrays da mesma dimensão)
-	Diferença (escalares e arrays)
*	Multiplicação de escalares e de arrays compatíveis
.*	Multiplicação de arrays elemento a elemento, tendo como resultado um array.
./	Divisão de arrays da mesma dimensão elemento a elemento, tendo como resultado um array.
^	Potenciação de escalares.
.^	Potenciação de arrays elemento a elemento, tendo como resultado um array.
'	Transposto de um "array".

- **Algumas constantes**

pi	Valor de π.
eps	Precisão relativa em vírgula flutuante (2^{-52}).
realmin	menor número em vírgula flutuante (2^{-1022}).
realmax	maior número em vírgula flutuante (2^{1023}).
Inf	Infinito.
NaN	not a number.

- **Alguns comandos úteis**

zeros(m,n)	Matriz m×n com zeros.
ones(m,n)	Matriz m×n com uns.
fix(x)	Dá a parte inteira de x.
det(A)	Determinante da matriz A
rref(A)	Matriz A condensada.
eig(A)	Valores próprios de A.
A'	Transposta de A.
mean(D)	Média dos elementos do array D.
std(D)	Desvio padrão do array D.

- **Utilidades várias**

→ Criar um vector coluna, n, com números de 0 a 9: n=(0:9)'.

→ Criar uma tabela (matriz) cujas colunas são n, n^2 , 2^n, usando o vector anterior, n: x=[n n.^2 2.^n].

→ Por a zeros a 2ª coluna de uma matriz A : A(:,2).

→ Resolver um sistema Ax=b : A=[1 2 3 ; 4 5 6; 6 7 8] , b=(1:3)' , x=inv(A)*b (se a inversa não existir dá erro).

→ Escrever um polinómio $p(x)=x^3-3x^2+1$: p=[1 -3 0 1].

→ Calcular as raízes de um polinómio: r=roots(p).

→ Ver os coeficientes de um polinómio cujas raízes estão no vector r : p1=poly(r).

1ª Experiência com o MATLAB: Operações básicas

1) À frente da "prompt" do MATLAB : >> , digite o seguinte, terminando cada linha com <*return*>:

```
>> (6+8)/(4+3)
>> 6+8/(4+3)
>> 6+8/4+3
>> 9+(6*2)/(3*4)
>>9+6*2/(3*4)
>> 9+6*2/3*4
```

R: 2, 7.1429, 11, 10, 10, 25.

2) Introduza as seguintes expressões. Em ambos os casos deve dar 12. Se não der verifique que erros cometeu.

a) $5+\frac{9^{3/2}+1}{2(5-3)}$ b) $\frac{17}{\frac{2}{3}+\frac{3}{4}}$

- **Nomes de variáveis**

  O MATLAB distingue as letras maiúsculas das minúsculas. Assim XPTO é diferente de xpto e de XpTo.

  Os nomes das variáveis devem ter até 31 caracteres.

  O primeiro caracter do nome da variável deve ser uma letra.

3) Digite o seguinte conjunto de comandos:

```
>> x=7.231
>> y1=x+1/x
>> y2=x-1/x
>> dif = y1^2 - y2^2
```

Usando as setas do teclado recupere o primeiro comando e introduza novo valor de x (à sua escolha) e <*return*>. Recupere do mesmo modo os restantes comando sem alterações.
Comente e justifique analiticamente os resultados.

R: 4 para qualquer valor de x.

- **O comando clear**

  Se pretendermos reutilizar uma variável anteriormente definida com características diferentes das que pretendemos (por exemplo x era um vector e agora pretendemos que seja um escalar) ou se pretendermos limpar todas as variáveis, usamos o comando *clear*.
  No primeiro caso *clear* <*nomevar*>, no segundo caso *clear all*.

- **Linhas de Comentário**

  Os comentários são úteis para documentar o programa.
  Numa linha, tudo o que escrevermos à frente do símbolo de percentagem % é tomado como comentário e portanto não é executado pelo MATLAB.

- **Supressão de output ;**

  Se pretendermos suprimir "outputs" intermédios, usamos **;** à frente da expressão que o calcula.
  Ex: y=x1+x2;

4) Usando as setas do teclado que permitem subir e descer no ecran linha a linha, pode aceder aos comandos que já utilizou. Execute novamente o conjunto de comandos do exercício 3), suprimindo o "output" dos resultados intermédios de x, y1 e y2.

- **Mais do que uma instrução por linha ,**

  Podemos escrever mais do que uma instrução por linha separadas por vírgula **,**. Esta não é necessária quando a instrução já tem ponto e vírgula **;** no fim, ou seja:

```
>> y1=x+1/x , y2=x-1/x
```

ou, com supressão do output,

```
>> y1=x+1/x ; y2=x-1/x;
```

5) Considere o conjunto de instruções:

```
>> clear all
>>x1=3;y1=3;
>>x2=8;y2=7;
>> % m - declive da recta que une os dois pontos: y=mx+b
```

```
>> m=(y2-y1)/(x2-x1)
>> % ordenada na origem – b
>>b=y1-m*x1
```

R: m = 0.6000, b = 2.2000.

Acrescente um comando que lhe permita calcular a intersecção da recta com o eixo dos xx.

R: x0 = -3.6667.

6) Seja y=sin(x) e x1=π/6 , x2=π/3.

a) Determine as imagens y1 e y2 destas abcissas.

b) Use as setas do teclado para correr de novo o procedimento de 5) (a partir do cálculo de m, claro).

R: a) y1 = 0.5000, y2 = 0.8660
b) m = 0.6991, b = 0.1340, x0 = -0.1917

- **Funções matemáticas elementares**

Abrindo o menu *Help* do MATLAB, escolhendo *MATLAB Help* e na janela da direita escolhendo *MATLAB documentation*, *MATLAB Functions Listed by Category* e *Elementary Math Functions*, pode ter uma ideia geral das funções disponíveis no MATLAB. Observe e retenha os seus nomes para futura utilização.

O argumento de cada função deve ser sempre colocado entre parêntesis.

2ª Experiência com o MATLAB : Utilização de funções

1) Atribua a x o valor 1.253 . Obtenha os valores de u,v,w,y,z dados por:

a) $u = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$

b) $y = sin^2(\pi x)$

c) $z = \frac{e^{sin\,x}}{\sqrt{x^2+1}}$

R: u = 0.1820, v = 0.6523, w = 3.0107, y = 0.5094, z = 1.6128

- **Criação de "M-files"**

Para entrar no Editor do MATLAB abrir o menu *File*, *New*, "M-file". Abre-se uma nova janela em branco.

2) Escreva o seguinte procedimento:

```
clear all
x1=3, y1=4, x2=8, y2=7
% m - declive da recta que une os dois pontos: y=mx+b
m=(y2-y1)/(x2-x1)
% ordenada na origem – b
b=y1-m*x1
x0=-b/m
```

Abra o menu *File-> Save As*, dê-lhe o nome **recta1** e guarde no directório de trabalho.

Execute agora o "M-file" que acabou de criar, escrevendo na "**Command window**" **recta1** seguido de <*return*>.

Ao escrever um nome na "**Comand window**", o MATLAB procede do seguinte modo:

1º Vê se é uma variável da qual se pretende imprimir o valor. Se não...
2º Vê se é um comando do próprio MATLAB. Se não...

3º Vê se existe no "**Current Directory"** um "M-file" com esse nome e executa-o (**daí a importância de se colocar no directório certo, o que contém os "M-files", quando pretende criar e executar esses "M-files"**). Se não ...

4º Responde
"??? Undefined function or variable name".

3) Crie um "M-file" para determinar os zeros da seguinte parábola:

$$f(x) = ax^2 + bx + c$$

usando a fórmula resolvente.

Dê-lhe o nome **zerosparab**.

```
clear all
a=1; b=-3; c=2;
rdelta=sqrt(b^2-4*a*c)
x1=(-b-rdelta)/(2*a)
x2=(-b+rdelta)/(2*a)
```

Terá que dar valores aos parâmetros a,b,c, mas como ainda não sabe fazer "input" pelo teclado, terá que os colocar no próprio programa.

Teste em particular os seguintes casos:

```
a) a=1, b=-3, c=2
b) a=1, b=2, c=1
c) a=1, b=0, c=1
```

R: a) x1= 1.00, x2= 2.00 b) x1=x2=-1.00
c) x1 = 0 - 1.0000i, x2 = 0 + 1.0000i.

Para alterar um "M-file" já criado basta abrir o Menu *File , Open, <nome do "M-file">* ou duplo click sobre o nome do ficheiro no "**Current Directory**".

- **Impressão formatada : 1ª Abordagem - fprint**

Certamente vai querer o seu output mais perceptível. Tem então interesse aprender a formatá-lo.

No seu programa **zerosparab** faça as seguintes alterações:

1º - suprima todos os "outputs" intermédios.

2º - Acrescente as seguintes linhas no fim do programa:

```
fprintf(' Menor raiz %5.2f \n',x1)
fprintf(' Maior raiz %5.2f \n',x2)
```

O que acontece quando as raizes são complexas?

Altere agora as linhas que acabou de escrever, para:

```
fprintf(' Raizes do polinómio %5.2f x^2 + %5.2f x+%5.2f \n',a,b,c)
fprintf(' x1= %5.2f + %5.2f i \n',real(x1), imag(x1))
fprintf(' x2= %5.2f + %5.2f i \n',real(x2), imag(x2))
```

real(x1) dá a parte real de x1.
Imag(x1) dá a parte imaginária de x1.

Neste momento o seu programa **zerosparab** deverá ter o seguinte aspecto:

```
clear all
a=1; b=-3; c=2;
rdelta=sqrt(b^2-4*a*c);
x1=(-b-rdelta)/(2*a);
x2=(-b+rdelta)/(2*a);
fprintf(' Raizes do polinómio %5.2f x^2 + %5.2f x+%5.2f \n',a,b,c)
fprintf(' x1= %5.2f + %5.2f i \n',real(x1), imag(x1))
fprintf(' x2= %5.2f + %5.2f i \n',real(x2), imag(x2))
```

R: Raízes do polinómio 1.00 x^2 + 0.00 x+ 1.00
x1= 0.00 + -1.00 i
x2= 0.00 + 1.00 i

fprintf é o comando para impressão formatada. Agora já podemos escrever os nossos resultados na forma que mais nos convém.

No comando fprintf todos os textos e indicações de formato ficam entre plicas. A simbologia %5.2f indica que o número será escrito no formato f (floating point) de ponto decimal, ocupando um total de 5 dígitos, com 2 decimais. A simbologia \n obriga à mudança de linha. Antes de fechar o último parêntesis deve colocar os nomes das variáveis a imprimir.

Adiante veremos mais sobre formatações.

- **"Arrays" – Vectores e matrizes**

Existem vários modos de criar um "array".

1º Colocar os elementos do "array" entre parêntesis rectos, separados por vírgulas ou espaços:

```
x=[2  3  4  5]
ou
y=[2,3,4,5]
```

2º Se os elementos do array forem valores igualmente espaçados, usa-se a notação seguinte:

```
z1=[-3:2:20]
z2=2.5+1.5*[0:0.04:1]
```

Em z1 tem-se:

-3 representa o primeiro valor do vector.
2 representa o espaçamento entre os elementos do vector.

20 representa o limite máximo para o último elemento do vector, de acordo com o espaçamento.

Em z2 é criado um vector com elementos de 0 a 1, espaçados de 0.04 (passo 0.04), que são multiplicados por 1.5 e a cada um é somado 2.5.

3º Uma alternativa a este procedimento é definir o vector usando a função MATLAB linspace:

```
w=linspace(2.5,4.0,10)
```

R: Columns 1 through 8
2.5000 2.6667 2.8333 3.0000 3.1667 3.3333 3.5000 3.6667
Columns 9 through 10
3.8333 4.0000

Neste caso cria um vector com 10 valores igualmente espaçados desde 2.5 até 4. O espaçamento é calculado pelo MATLAB.

4º Podem-se criar "arrays" por concatenação de outros "arrays":

```
x1=[x z1]
```

5º Há casos em que se pode definir um array à custa de funções que não o linspace:

```
A=rand(1,9)
```

Obtemos um vector com 9 elementos de uma distribuição normal com média zero e variância 1.

Se quisermos operar sobre um determinado elemento do vector, por exemplo o 5º, fazemos o seguinte:

```
y1=2*A(5)
```

que dá o dobro do 5º elemento do vector A.

6º Para introduzir uma matriz do tipo 2 por 3 basta fazer:

```
A=[1  2  3 ; 4  5  6]
```

Neste caso ; indica que vamos definir outra linha.

7º Para introduzir um vector coluna 3 basta fazer:

```
A=[1  2  3 4  5  6]'
```

- **Utilização de funções matemáticas em arrays**

Podemos aplicar funções matemáticas a arrays.
Nas operações de divisão, multiplicação e exponenciação entre arrays usam-se os símbolos habituais precedidos de . (ponto). Estas operações têm como resultado arrays em que cada elemento é o resultado da operação sobre os elementos homónimos de cada um dos arrays.

Ex.:

```
x=[1 2 3] e y=[4 5 6] e z=[7 8]
x.*y=[1*4 2*5 3*6] , x./y=[1/4 2/5 3/6] ,
x.^2=[1^2 2^2 3^2], x^y=[1^4  2^5  3^6]
x.*z dá erro porque os vectores não são da mesma dimensão.
```

Exercícios:

1) Entre no editor do MATLAB e introduza o seguinte conjunto de comandos:

```
clear all
x=[0:pi/6:pi]
y=sin(x)
z1=1-2*y.^2  %Calcula o quadrado de cada componente
z2=cos(2*x)
z=z1./z2 %Divide os vectores componente a componente
```


R:

```
x =  0    0.5236    1.0472    1.5708    2.0944    2.6180    3.1416
y =  0    0.5000    0.8660    1.0000    0.8660    0.5000    0.0000
z1 = 1.0000    0.5000   -0.5000   -1.0000   -0.5000    0.5000
1.0000
z2 =  1.0000    0.5000   -0.5000   -1.0000   -0.5000    0.5000
1.0000
z =  1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
1.0000
```

Porque é que os elementos de z são todos iguais a 1?

O comando *format long* destina-se a obter mais casas decimais no output.

- **Gráficos**

Algumas Funções gráficas do Matlab

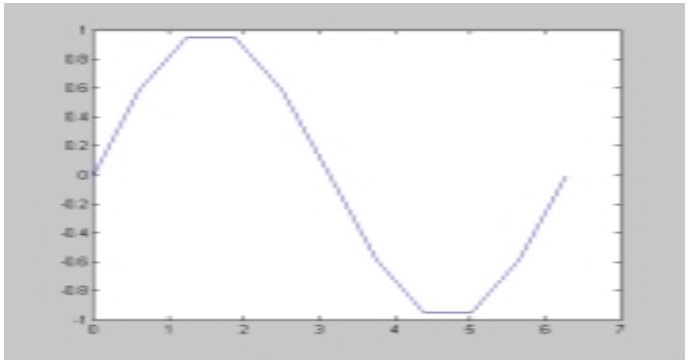
axis	Fixa os limites dos eixos
bar	Faz gráfico de barras
contour	Faz contour plots
grid	Faz grid no plot
gtext	Coloca o texto onde se quiser com o rato
histogram	Faz histograma
hold	Entra em espera até o plot estar terminado
plot	Faz um plot x-y
polar	Faz um plot polar
loglog	Faz plot log versus log (ambos os eixos logaritmizados)
semilogx	Faz um plot x-y semilog (eixo-x logarítmico)
semilogy	Faz um plot x-y semilog (eixo-y logarítmico)
text	Coloca texto no plot, numa posição especificada no comando
title	Coloca título no gráfico
xlabel	labels eixo-x
ylabel	labels eixo-y
Fplot	plot de uma função

Dados 2 vectores com o mesmo nº de elementos, o comando plot(x,y) permite desenhar o gráfico de x *vs* y.

2) Introduza na linha de comandos do MATLAB os seguintes comandos:

```
clear all
x=linspace(0,2*pi,11);
y=sin(x);
plot(x,y)
```

R:



Obtém um gráfico muito tosco, claro, porque os pontos são poucos e muito espaçados.

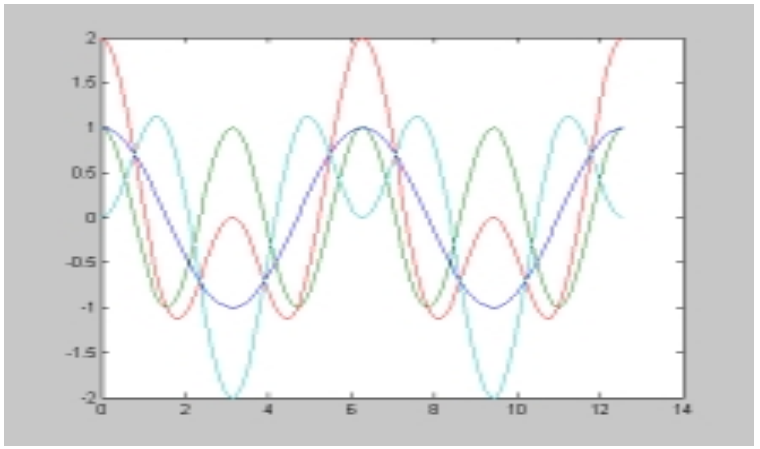
Tente agora o seguinte:

3) Crie um "M-file" com o nome **graph2**, com os seguintes comandos:

```
clear all
t=linspace(0,4*pi,201);
y1=cos(t); y2=cos(2*t); y3=y1+y2; y4=y1-y2;
plot(t,y1,t,y2,t,y3,t,y4)
```

Execute este programa.

R:



O gráfico é bonito, mas não muito esclarecedor. Que curva corresponde a cada função?

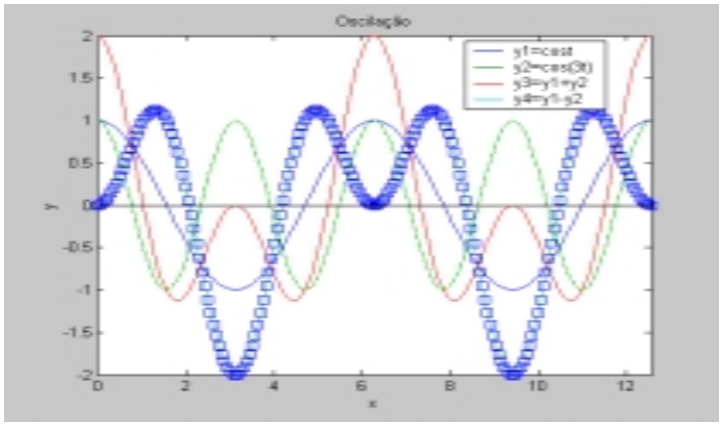
Convém acrescentar legenda e títulos.

4) Altere agora o **graph2** (basta fazer duplo click sobre o nome do programa), acrescentando-lhe algumas linhas:

```
clear all
t=linspace(0,4*pi,201);
y1=cos(t); y2=cos(2*t); y3=y1+y2; y4=y1-y2;
axis([0 4*pi –2 2])  % Define os limites dos eixos
hold on %Não desenha logo o gráfico. Espera até definirmos...
%tudo o que pretendermos.Quando encontra hold off desenha
plot(t,y1,'b –',t,y2,'g:',t,y3,'r',t,y4,'s') %Desenha linhas, ...
 %cores e padrões b-blue, g-green, r-red, s-desenha quadrados
%k-preto.
legend('y1=cost','y2=cos(3t)','y3=y1+y2','y4=y1-y2')
xlabel('x')
ylabel('y')
title('Oscilação')
plot([0 4*pi],[0,0],'k') % desenha o eixo a preto
hold off
```

Observe bem o gráfico que obteve.

R:

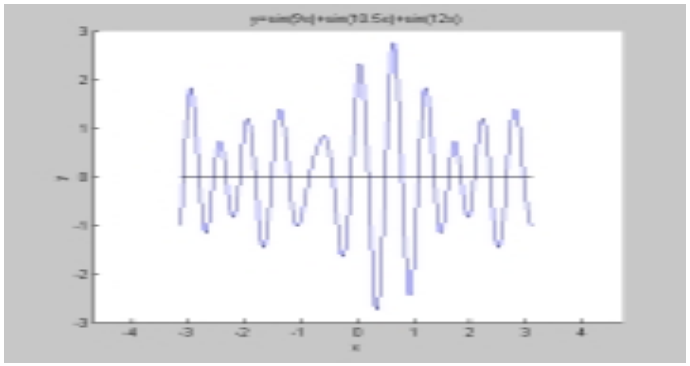


5) Faça o gráfico de $y = sin\,9x + sin\,10.5x + sin\,12x$ para $-\pi \le x \le \pi$, usando 601 pontos.

Para isso crie um "M-file" com o nome **sinsinsin**.

```
clear all
t=linspace(-pi,pi,601);
y1=cos(9*t); y2=cos(10.5*t); y3=sin(12*t); y4=y1+y2+y3;
axis([-1.5*pi 1.5*pi -3 3])  % Define os limites dos eixos
hold on % Evita que apareçam janelas intermédias com o graf
plot(t,y4,'b -')
xlabel('x')
ylabel('y')
title('y=sin(9x)+sin(10.5x)+sin(12x)')
plot([-pi pi],[0,0],'k')
hold off
```


R:



6) Exemplo com escala logarítmica

Considere o seguinte problema: Desenhar o gráfico da função $y = e^x$ com x a variar entre 0 e 100, com 200 pontos.

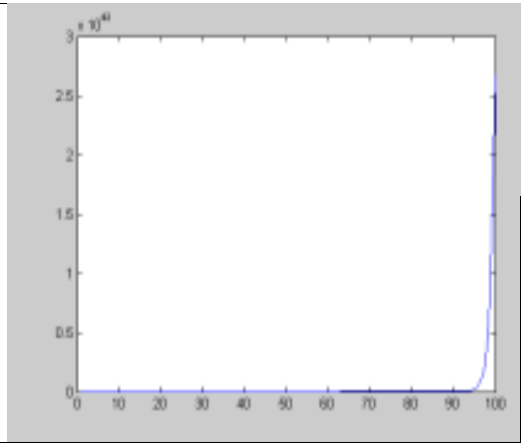
Tem várias opções:

Uma é:

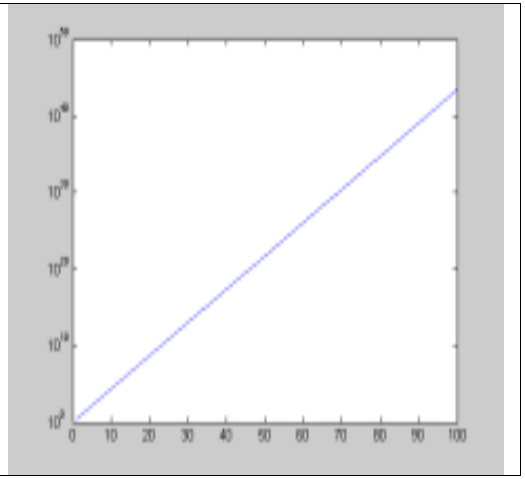
```
x=linspace(1,100,200);
y=fexp(x);
plot(x,y)
```

com a function:

```
function y=fexp(x)
y=exp(x);
```



Outra é:

x=linspace(1,100,200);
y=fexp(x);
semilogy(x,y)
	

com a function:	
function y=fexp(x) y=exp(x);	

- **Text e gtext**

  Este comando permite escrever texto dentro de um gráfico.

  No comando *text* é necessário dar a posição em coordenadas do gráfico onde quer que o texto apareça, no comando *gtext* basta clickar com o rato sobre o gráfico, aquando da sua execução.

```
text(xposi,yposi,' Texto ')
gtext(' Texto')
```

Para acrescentar ao seu gráfico, usando o comando *text*, valores numéricos que já tenha calculado, terá que passar primeiro esses valores para uma string. Um dos comandos para esse efeito é *num2str* (number to string).

Suponha que o valor a passar para string está na variável x.
Deverá fazer então:

```
strx = num2str(x);
gtext(strx)
```


- **Cores, tipos de linha e marcas**

Linhas	**Símbolo**	**Marcas**	**Símbolo**
Linha sólida	-	ponto	.
Linha Tracejada	--	sinal mais	+
Ponteado	:	asterisco	*
Traço ponto	-.	círculos	o
		xis	x

Cores	Símbolo
Red	r
green	g
blue	b
white	w
invisible	I
black	k

- **Ciclos for**

  A sintaxe do comando é a seguinte:

```
for varname=start:step:finish
    comandos ...
end
```

1) Suponha que pretende calcular $\sum_{n=1}^{20} \frac{1}{n^2}$, usando um **ciclo for**. Então poderá usar o seguinte programa, a que poderá dar o nome **loop1**:

```
clear all
s=0;
for n=1:1:20
    s=s+1/n^2;
end
sum=s
```

R: sum = 1.5962

- **Ciclos while**

A sintaxe do comando é a seguinte:

```
while expressão
      comandos ...
end
```

2) Suponhamos agora que pretende calcular uma aproximação para a série $\sum_{n=1}^{\infty} \frac{1}{n^2}$, com um erro inferior a 0.0005.

Para isso é mais conveniente usar um ciclo **while** do que um ciclo for.

Altere o seu programa **loop1** do seguinte modo e guarde-o com o nome **loop2** (usando *Save As*):

```
clear all
   format long %Escreve os números com mais casas decimais
   s=0;
   s1=0;
   s2=1000;
   n=0;
   while (abs(s2-s1))>=0.00000005
         n=n+1;
         s=s+1/n^2;
         s1=s2;
         s2=s;
   end
   niter=n
   sum=s2
   erro=abs(s2-s1)
```

R: niter = 4473
sum = 1.64471052823287

Compare este valor com o obtido no exercício anterior. Comente.

- **if-else-end e if-elseif-else-end**

Algumas variantes da sintaxe deste comando if são as seguintes:

```
if expressão
      comandos ...
end
```

ou

```
if expressão
      comandos ...
else
      comandos ...
end
```

ou

```
if expressão 1
      comandos ...
elseif expressão 2
      comandos
else
      comandos
end
```

3) Suponha que uma loja dá descontos de 1%, 5% e 10% consoante a despesa for inferior a 100€ , entre 100€ e 200€ ou superior a 200€.

Faça um programa para calcular o valor a pagar por cada cliente.

Dê-lhe o nome **decide1**.

Experimente para 78.55€, 120.2€ e 223.76€.

R:

```
clear all
despesa=100
if despesa>200
```

```
  despesa=despesa*(1-0.1);
elseif despesa>100
  despesa=despesa*(1-0.05);
else
  despesa=despesa*(1-0.01);
end
format bank %Utiliza 2 decimais
paga=despesa
```

R: paga=77.76 , paga= 114.19, paga= 201.38

- **Introdução de dados através do teclado durante a execução de um programa**

  Até agora, cada vez que pretendemos alterar um parâmetros , temos alterado o próprio programa, após o que, obrigatoriamente para a modificação fazer efeito, temos feito *Save*. Este método é pouco prático, pelo que vamos agora aprender a fazer o "input" dos valores pelo teclado.

4) No programa **decide1** que acabou de criar, substitua a 2ª linha pelo seguinte:

```
despesa=input(' Valor da despesa: ')
```

Faça *Save* e corra o programa.

5) Suponha agora que tem sempre muitos clientes e que quer ter o programa sempre a postos para a introdução de uma despesa diferente de zero (se for zero o programa termina).

Para isso pode por exemplo usar um ciclo while . Chame ao programa **decide2**.

R:

```
clear all
despesa=input(' Valor da despesa: ')
while despesa~=0
```



```
  if despesa>200
        despesa=despesa*(1-0.1);
  elseif despesa>100
        despesa=despesa*(1-0.05);
  else
        despesa=despesa*(1-0.01);
end
    format bank %Utiliza 2 decimais
    paga=despesa
  despesa=input(' Valor da despesa (para terminar 0): ')
end
```

Corra o programa para diversos valores da despesa.

Para terminar o programa basta introduzir zero.

- **Formatação de "outputs" alfanuméricos**

format short	4 decimais sem expoente.
format short e	4 decimais na mantissa e com expoente
format short g	4 decimais, mas o programa escolhe a melhor opção de representação.
format long	14 decimais sem expoente.
format long e	14 decimais na mantissa e com expoente
format long g	14 decimais, mas o programa escolhe a melhor opção de representação.
Format bank	2 decimais
'%5.2f'	Formato f, 5 dígitos no total dos quais 2 são decimais
'\n'	Mudança de linha

- **Function "M-files"**

Quase todos os comandos que temos usado estão guardados em "M-files".

Existem vantagens em guardar funções em ficheiros separados (M-function), de programas. De entre elas, a possibilidade de essa função poder ser usada tanto noutros programas, como mais vezes

dentro do próprio programa. Por outro lado a construção de algumas funções é bastante complexa, pelo que o melhor é criá-las em ficheiros separados.
No MATLAB as "M-functions" são compiladas em vez de simplesmente interpretadas, o que torna a sua execução mais rápida.

A primeira instrução de um "M-function" deve ser a indicação de que o que se segue é uma função: *function*, seguida das variáveis de saída às quais se atribui o nome da função com os seus argumentos. Por exemplo, o comando linspace:

```
function y = linspace(d1, d2, n)
```

A construção de "M-function" obedece a um conjunto de regras:

- Os nomes de função e do ficheiro que a contém têm que se iguais.

- O primeiro comando executável deve ser :

function <variáveis de saída> = <nome da função>(variáveis de entrada)

- Não se deve incluir nenhum comando *clear* numa função.

- Não é usual nas funções termos comandos para imprimir resultados, embora não seja proibido.

- A execução de uma função termina no final de lista de comandos dados no ficheiro ou quando encontra o comando return.

1) Construa a seguinte função: $fun1(x) = \frac{x - cos(x)}{2 + x^2}$

Para isso entre no editor de "M-files" e introduza os seguintes comandos:


```
function y=fun1(x)
y=(x-cos(x))./(2+x^2);
```

Guarde agora o ficheiro com o nome **fun1**.

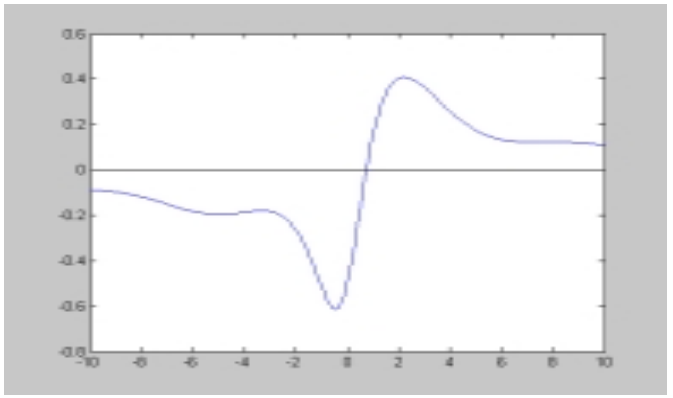
2) Construa agora um programa para desenhar esta função e chame-lhe **plfun1**.

```
clear all
hold on
fplot(@fun1,[-10,10])
plot([-10  10],[0  0], 'k')
hold off
x_intersec=fzero(@fun1,[0  1])
[xmin,ymin]=fminbnd(@fun1,-3,3)
```

De notar o sinal @ antes do nome da nossa função nos comandos *fplot* e *fminbnd*. Este sinal é necessário para o MATLAB saber que vai calcular uma função dentro de outra.

A função fminbnd calcula o ponto de mínimo da função.

R:



Neste programa, usando a instrução fplot não foi necessário definir os vector das abcissas. Se usasse o comando plot teria que definir esse vector e calcular os valores das ordenadas do seguinte modo:

```
clear all
x=linspace(-10,10,200);
y=fun1(x);
hold on
plot(x,y])
plot([-10  10],[0  0], 'k')
hold off
```

- **Polinómios**

Os comandos mais usados em operações com polinómios são:

roots - calcula as raizes
poly - dadas as raizes constroi o polinómio
polyval - calcula o valor de um polinómio
polyder- diferencia um polinómio
conv - multiplica polinómios
deconv- divide polinómios

1) Considere o seguinte polinómio:

$$p(x) = 3x^5 - 4x^4 + 7x^2 - 9x + 3$$

A sua representação em MATLAB é [3 -4 0 7 -9 3].

Considere ainda os polinómios cujas representações são:

q=[4 5 -6] e qp=[0 0 0 4 5 -6].

Obtenha um novo polinómio que é a soma de p com o dobro de qp.

Calcule as raizes de p e de q e chame-lhes proots e qroots respectivamente.

Calcule os polinómios cujas raizes são as mesmas que as de p e de q. Que conclui?

Determine o valor de p para x=2.31.

Calcule as derivadas de p e de q.

Multiplique p por q.

Divida p por q, obtendo o quociente e o resto desta divisão.

Desenhe o gráfico do polinómio p(x).

R: O seu programa deve-se parecer com o seguinte:

```
clear all
p=[3 -4 0 7 -9 3], q=[4 5 -6], qp=[0 0 0 4 5 -6]
newp=p+2*qp
praizes=roots(p)
qraizes=roots(q)
p1=poly(praizes)
q1=poly(qraizes)
y=polyval(p,2.31)
pderiv=polyder(p)
qderiv=polyder(q)
pq=conv(p,q)
[quoc,resto]=deconv(p,q)
hold on
x=linspace(-1.7,2,300);
grid on
plot(x,polyval(p,x))
hold off
```

R:

```
newp =
  Columns 1 through 6
  3.00  -4.00   0   15.00   1.00  -9.00

praizes =
        -1.37
         0.58
         0.58
         1.00
         0.53

qraizes =

        -2.00
```

```
        0.75

p1 =
  Columns 1 through 6
   1.00  -1.33  -0.00   2.33  -3.00  1.00

q1 =
        1.00         1.25        -1.50

y =
       102.99

pderiv =
       15.00        -16.00            0        14.00        -9.00

qderiv =
        8.00         5.00

pq =
  Columns 1 through 8

 12.00  -1.00  -38.00  52.00  -1.00  -75.00  69.00  -18.00

quoc =
        0.75        -1.94         3.55        -5.59

resto =
  Columns 1 through 5
  0  0  0  0  40.23  -30.54
```

- **Ideias para fazer plots**

Considere o seguinte programa de MATLAB e tente perceber o que ele faz:

```
clear all
k=0
alfa=input(' Valor de alfa (Ar: 64.5eVA^6): ')
beta=input(' Valor de beta (Ar: 99.6eVA^6): ')
n=input(' Valor de n (Ar: 6): ')
m=input(' Valor de m (Ar: 12): ')
while alfa~=0
k=k+1
epsilon=0.1;
```



```
ind1=0;
ind2=0;
ind3=0;
ind4=0;
a=100000;
b=100000;
r=linspace(.5,5,101);
%potencial e forca de atracçao
for i=1:101
uatra(i)=-alfa/(r(i)^n);
fatra(i)=-(n*alfa)/(r(i)^(n+1));
% indice do zero do potencial de atraccao
if uatra(i)>-epsilon
   if ind1==0
      ind1=i;
   end
end
%potencial e forca de repulsao
urep(i)=beta/(r(i)^m);
frep(i)=(m*beta)/(r(i)^(m+1));
%indice do zero do potencial de repulsao
if urep(i)<epsilon
   if ind2==0
      ind2=i;
   end
end
% potencial e forca totais
u(i)=uatra(i)+urep(i);
f(i)=fatra(i)+frep(i);
% minimo do potencial
if u(i)<a
   a=u(i);
   ind3=i;
end
% minimo da forca
if f(i)<b
   b=f(i)
   ind4=i;
end
end
fprintf('|Uat|<0.1 - distancia= %8.4e \n',r(ind1))
```

```
fprintf('|Urep|<0.1 - distancia= %8.4e \n',r(ind2))
fprintf('Umin - distancia= %8.4e \n\n',r(ind3))
sprintf('Umin=%8.3f,r0=%7.3f',u(ind3),r(ind3))
% Passa para string as distancias
aa=ans
sprintf('r=%5.2f',r(ind1))
ab=ans
sprintf('r=%5.2f',r(ind2))
ac=ans
sprintf('r=%5.2f',r(ind4))
bb=ans
sprintf('r=%5.2f',r(ind3))
bc=ans
% figura para o potencial
figure(k)
k=k+1;
hold on
%subplot(2,1,1);
axis([0.5 5 1.5*u(ind3) -1.5*u(ind3)])
%plot(r,uatra, 'k',r,urep, 'b',r,u, 'r')
plot(r,uatra,r,urep,r,u)
%plot()
%plot()
title(' Energia potencial: U(r), Uatr(r), Urep(r)','FontSize',14)
text(r(ind3)+0.05,u(ind3),aa)
text(r(ind1),u(ind1),ab)
text(r(ind2),u(ind2),ac)
legend('Uat','Urep','U')
plot(r(ind1),uatra(ind1),'+')
plot(r(ind2),urep(ind2),'*')
plot(r(ind3),u(ind3),'o')
plot([0.5 5],[0,0],'k')
hold off
%subplot(2,1,2);
% figura para a força
figure(k)
hold on
axis([0.5 5 1.5*min(f) -1.5*min(f)])
plot(r,fatra,r,frep,r,f)
plot([0.5 5],[0,0],'k')
legend('Fatra','Frep','F')
```



```
title('Forças: F(r), Fatra(r), Frep(r)','FontSize',14)
plot(r(ind4),f(ind4),'o')
text(r(ind4),f(ind4),bb)
plot(r(ind3),0,'*')
text(r(ind3),0,bc)
hold off
% pede novos valores para continuar a bonecada
alfa=input(' Valor de alfa (sug: 99.6) (0 - para acabar): ')
if alfa~=0
beta=input(' Valor de beta (sug: 64.5): ')
n=input(' Valor de n (sug: 6): ')
m=input(' Valor de m (sug: 12): ')
end
end
```

Tente reescrever este programa de um modo mais condensado, utilizando instruções MATLAB tais como *min*, *max*, etc....

Bom trabalho