universidade de aveiro deti departamento de eletrónica, telecomunicações e informática

#43835 - Sensores e Sinais
Licenciatura em Engenharia de Computadores e Informática

Guião Prático
Introdução ao Octave

Paulo Pedreiras (pbrp@ua.pt),
Pedro Fonseca (pf@ua.pt)
Luis Silva (lems@ua.pt)
UA/DETI/IT

Objectivo

O objectivo deste guião é permitir aos alunos conhecerem algumas das funcionalidades básicas da ferramenta e linguagem de Programação Octave, que será usada na primeira parte das aulas práticas.

O Octave é extremamente útil no contexto desta UC, porque lida com facilidade com a álgebra linear e matemática vectorial por trás dos sinais, permitindo a visualização instantânea de conceitos matemáticos. Dispõe de ferramentas integradas para análise espectral e filtragem de sinais, visualização de dados, entre muitas outras funcionalidades, substituindo cálculos manuais morosos por cálculos imediatos (ou quase), permitindo assim o foco no estudo do comportamento dos sinais, em vez de apenas na resolução matemática.

Dada a formação anterior dos alunos, bem como a excelente documentação disponível, a maior parte do estudo e do trabalho presente neste guia deverá ser realizado autonomamente, fora da aula prática.

Operação básica – vectores e matrizes

1. Crie um vector linha a = [1 2 3 4 5]. Mostre a sua transposta. Crie um vector coluna b = [0; 1; 2; 3; 4]. Mostre a sua transposta. A menos que seja dito algo em contrário, assume-se de agora em diante que todos os vectores são vectores-coluna.
2. Crie um intervalo de valores r1 de 1 a 10 com a sintaxe ":". Repita com um passo não inteiro (r2). Crie um vector-linha r3, usando o "linspace", com 100 pontos entre 0 e 2π, incluindo os limites 0 e 2π. Verifique a capacidade de suprimir outputs não desejados com o operador ";" no final dos comandos.
3. Multiplique o vector "a" por si mesmo (a*a). Explique o resultado.
4. Multiplique o vector "a" por si mesmo, usando a multiplicação elemento a elemento. Tente também a operação "a.^3" e analise os resultados.
5. Calcule o produto interno dos vectores "a" e "b".
6. Liste as variáveis que existem no seu espaço de trabalho digitando "whos". Deverão haver pelo menos a, b, r1, r2, r3
7. Crie uma matriz M com 3 linhas e 4 colunas, com conteúdo aleatório (ver função rand). Mostre a segunda linha de M. Em seguida mostre as duas primeiras linhas e 3 primeiras colunas de M. Faça uso, onde adequado, dos operadores ":" e "end".
8. Calcule o produto da matriz M por ela própria. Calcule também, o produto de M pela sua transposta. Interprete ambos os resultados.
9. Explore as funções eye(), zeros(), ones()
10. Pode efectuar-se a remoção de colunas e linhas de uma matriz atribuindo-lhes o valor "[]". Digite o comando N=M(:,2:4) e analise o resultado.

Polinómios

Existe em Octave um conjunto de funções para manipular polinómios. A definição de polinómios é feita criando vectores cujos elementos são os coeficientes do polinómio ordenados por potência decrescente tal como o exemplo seguinte ilustra. Para representar o polinómio $p(x) = x^2 - 3x + 2$, cria-se o vector:

>>p=[1 -3 2];

e para calcular as suas raízes existe a função

>>r= roots(p)

ans =

2

1

Para obter os coeficientes de um polinómio a partir das raízes pode-se utilizar a função

>>poly(r)

ans =

1 -3 2

Também é possível calcular o valor de um polinómio num conjunto de pontos utilizando a função polyval, tal como o exemplo ilustra:

polyval(p,r)

ans=

0

0

e que, como se pode ver, calculou o valor do polinómio nos seus zeros.

O produto entre dois polinómios é obtido através da função "conv()" (convolução). O exemplo seguinte ilustra o produto entre os polinómios $x^2 + 1$ e $x^3 + x - 1$ dado por $x^5 + 2x^3 - x^2 + x - 1$

>>p1= [1 0 1]

>>p2= [1 0 1 -1]

>>conv(p1,p2)

ans=

1 0 2 -1 1 -1

Exercícios:

1. Calcule o seno, o coseno, a tangente, a raiz quadrada e a raiz cúbica de π/2.
2. Calcule o logaritmo e a raiz quadrada de -1.
3. Calcule o valor da função e^x em 100 pontos do intervalo [−1 . . . 1].
4. Calcule o valor da função sin(x + pi/10) ∗ cos(x) em 100 pontos do intervalo [−π . . . π].
5. Calcule o produto dos polinómios $x^6 + 10$ e $x^2 - 2x + 3$.
6. Obtenha o polinómio cujas raízes são os números inteiros 1, 2 e 3.
7. Calcule os zeros do seguinte polinómio p(x) = x3 + 4x2 − 3x + 1. Calcule o valor do polinómio em 100 pontos da forma $x = e^{j\omega}$ com $\omega \in [0 \ldots 2\pi]$.

Gráficos

Gráficos elementares.

Exercícios:

1. Faça o plot da função seno para dois períodos.
2. Compare o gráfico da função seno com o da função coseno para ângulos entre 0 e 2π e um passo de pi/150. O seno deve ser representado por círculos azuis e o coseno por uma linha verde a cheio.
3. Desenhe o gráfico da função sin (θ) /θ com θ a variar de −2π a 2π e passos de π/10.
4. Visualize o gráfico da função e^x para 100 valores de $x \in [0 \ldots 5]$.
5. Visualize o gráfico da função e^{-x} para 100 valores de $x \in [0 \ldots 5]$.
6. Visualize o gráfico da função $\log_e x$ para valores de $x \in [1 \ldots 5]$.
7. Construa uma matriz X em que a primeira coluna é constituída com 100 valores da função e^x e a segunda coluna por 100 valores da função $\log_e x$ para o intervalo $x \in [1 \ldots 2]$. Execute o comando plot(X).

Gráficos de variáveis complexas

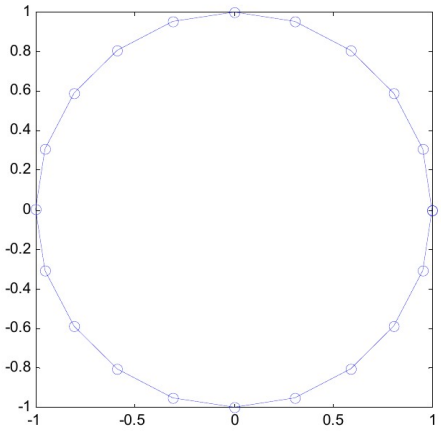
Quando o argumento da função plot é complexo, em abcissa é colocada a parte real do vector e em ordenadas a parte imaginária. Pode-se pensar neste processo como uma forma compacta de representação. O comando que se segue

>> plot(z)

é equivalente a fazer

>>plot(real(z),imag(z))

Vejamos um exemplo em que se utiliza a função axis square, para que a figura obtida seja uma circunferência.




```
>>w= 0:pi/10:2*pi;
>>plot(exp(i*w),'-o')
>>axis square
```

Exercício:

1. Visualize em gráfico a parte real e a parte imaginária da função $e^{j\omega}$, para valores de $\omega \in [0 \ldots 2\pi]$ e passo de $\pi/10$.

Programação em Ocatve

O Matlab possui um conjunto de instruções que permitem a construção de programas (scripts e funções).

A instrução IF

Esta instrução permite a execução condicional de código tendo em conta uma dada condição lógica.

A estrutura é a seguinte:

```
if cond,
        instrução 1
        instrução 2
        ...
else
        instrução 3
        instrução 4
        ...
end
```

Se a condição "cond" for verdadeira são executadas as instruções 1 e 2, senão serão executadas as instruções 3 e 4. A directiva end no final indica o fim da instrução if.

Na forma mais simples a directiva else pode ser omitida.

A instrução FOR

Esta instrução permite a repetição de um conjunto de instruções. A sintaxe de utilização é a seguinte:

```
for variável= início:passo:fim,
        instrução 1
        instrução 2
        ...
end
```

Exemplo:

```
>>r= 10;
>>for n= 1:10,
>>      r= r+10
>>end
```

Note-se que a abordagem algorítmica só deve ser utilizada quando não se encontra forma de realizar as operações de forma vectorizada, pois o Octave (tal como o Matlab) utiliza uma linguagem interpretada, e assim relativamente lenta.

Exercícios:

1. Gere um vector x com 1000 elementos utilizando a função rand. Extraia para um outro vector todos os elementos de x menores que 0.3. Utilize as instruções for e if.
2. Neste exercício pretende-se visualizar a função $e^{\alpha x}$ quando se varia o parâmetro α segundo a expressão $\alpha = 1 + 0.1 \times n$, com $n \in [1, 2, \ldots, 10]$. Armazene cada uma das variantes da função numa coluna de uma matriz A (100 × 10) considerando 100 pontos para a variável $x \in [0 \ldots 2]$. Para visualizar as curvas simultaneamente faça apenas plot(A) e experimente mesh(A).
3. A expansão em série de Taylor da função seno é dada pela expressão:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots + \frac{x^n}{n!} \sin\left(n\frac{\pi}{2}\right).$$

   Calcule a aproximação para 5 termos da série em 100 pontos do intervalo $[0 \ldots 2\pi]$ e faça um gráfico comparando a aproximação com a função seno.
4. Usando a instrução for, construa uma matriz de 10 × 10 em que cada coluna é um vector v cujos elementos são os números inteiros de 1 a 101 .

Scripts e funções

O Octave permite a criação de scripts e funções.

Os scripts são ficheiros simples que contêm uma sequência de comandos Octave que são executados exactamente como se fossem digitados um a um na janela de comandos. Eles partilham o "Espaço de Trabalho Global", o que significa que qualquer variável criada num script permanece na memória após a sua conclusão. São úteis para automatizar um fluxo de trabalho específico, gerar

um gráfico ou executar uma experiência rápida em que não é necessário passar dados de um lado para o outro.

As funções são blocos modulares de código que recebem entradas específicas, processam-nas e retornam saídas. Ao contrário dos scripts, elas têm o seu próprio "Espaço de Trabalho Local", portanto, as variáveis definidas dentro de uma função não existem no ambiente principal nem interferem com outras partes do seu programa. São úteis pois permitem escrever uma parte da lógica uma vez e reutilizá-la infinitamente em diferentes projectos, tornando o código mais rápido de desenvolver, mais organizado e potencialmente mais robusto e fácil de depurar.

Cuidados a observar:

- Certifique-se sempre de que o nome do ficheiro é exactamente igual ao nome da função (por exemplo, my_func.m para a função my_func).
- Use pontos e vírgulas ";" dentro das suas funções para impedir que o Octave imprima todos os cálculos intermediários na consola.
- Deve adicionar uma linha de comentário, explicativa da funcionalidade e interface/uso da função, logo abaixo da definição da função. Ao escrever "help your_function_name" na janela de comando, esta linha aparecerá, facilitando o uso das funções.

Por exemplo, a função abaixo converte graus Centígrados em graus Fahrenheit. Note a presença e uso de argumentos de entrada e de saída:

```
function fahrenheit = celsius_to_fahrenheit(celsius)
        % This function converts Celsius to Fahrenheit
        fahrenheit = celsius * (9/5) + 32;
end
```

Exercícios:

1. Crie e valide a função mostrada no exemplo anterior
2. Crie e valide uma função que calcule o perímetro e área de um rectângulo cujos lados são passados como argumentos.
3. Crie e valide uma função que aceite como entrada um vector com N amostras de um dado sensor e devolva o valor médio, máximo, mínimo e desvio padrão das amostras.

Comandos genéricos

Formatação de saída

O comando "format" controla a forma como os valores numéricos são apresentados. Este comando controla apenas a forma com os valores são apresentados e não a forma como estes são representados internamente. Os seguintes exemplos ilustram os formatos mais comuns:

```
>> format short
>> a= [1/5 pi]
   0.2000   3.1416
>> format short e
>> a
    2.0000e-001   3.1416e+000
```

>> format long

>> a

0.20000000000000 3.14159265358979

>> format long e

>> a

2.000000000000000e-001 3.141592653589793e+000

Uma opção útil do comando format é format compact que elimina as linhas em branco extra introduzidas entre a apresentação das variáveis permitindo uma apresentação dos resultados mais compacta.

Pastas e ficheiros

Pode visualizar-se a pasta corrente com o comando “pwd”. O comando “cd” permite navegar pela hierarquia de pastas. A interface gráfica permite navegar visualmente, com “clicks de rato”.

Deve salientar-se que, para serem executados, os scripts e funções devem estar na pasta corrente ou numa pasta que esteja no “search path”/caminho de pesquisa.

No GNU Octave, a função “addpath” adiciona pastas ao caminho de pesquisa para localizar funções e scripts. Use addpath(“nome_do_diretório”) para adicionar uma pasta ou addpath(genpath(“nome_do_diretório”)) para incluir subdiretórios. Essas alterações são temporárias, a menos que sejam guardadas no ficheiro “octaverc” usando o comando “savepath”.

Comandos-chave para gerar caminhos:

- Ver caminho actual: Digitar “path” sem argumentos para ver o caminho de pesquisa actual.
- Adicionar directório: addpath(‘~/Octave’) adiciona uma pasta. Use addpath(‘dir1’,'dir2') para múltiplos directórios.
- Adicionar subdirectórios: addpath(genpath(‘my_folder’)) adiciona my_folder e todas os seus subdirectórios ao caminho.
- Remover directório: rmpath(‘folder_name’) remove um diretório do caminho.
- Guardar caminho permanentemente: use “savepath” para que as alterações permaneçam entre sessões.
- Redefinir caminho: path(default) restaura o caminho padrão.
- Alterações persistentes: para garantir que os caminhos sejam carregados sempre que o Octave for iniciado, adicione o comando addpath ao seu ficheiro de configuração .octaverc.
- Verificar caminhos:
  - pwd mostra o directório de trabalho actual.
  - filesep retorna o separador de directórios do sistema operacional.

Os comandos LOAD e SAVE

Com estes comandos é possível guardar num ficheiro variáveis do Octave que estejam no "workspace". O comando save permite guardar as variáveis para disco e o comando load permite carregá-las para o "workspace". O exemplo seguinte ilustra a funcionalidade destes comandos.

>> A= rand(2)

A =

0.61543
0.92181
0.79194
0.73821

>>b= rand(1,3)

b=

0.9501
0.2311
0.6068

>>save ficheiro A b

>>load ficheiro

Os comandos "load" e save permitem igualmente guardar ficheiros em formato ASCII.

>> save ficheiro A b -ascii

sendo mais fácil para o utilizador verificar o conteúdo com um simples editor de texto.

Exercício:

1. Execute a sequência anterior de comandos e guarde as variáveis A e b em formato ASCII num ficheiro com o nome "teste.txt". Edite o conteúdo desse ficheiro com um editor de texto. Repita a operação para o ficheiro gerado com "save ficheiro A b".

2. Após ter criado as variáveis A e b tal como é indicado no texto, execute o comando "whos" e compare a memória ocupada por cada uma delas. Visualize a variável b nos seguintes formatos numéricos: long, short e e long e. Finalmente, apague as variáveis e verifique com o comando whos.

Agradecimento

Algum do texto e exercícios foram adaptados do documento "Matlab num Instante", Versão 1.3, José Manuel Neto Vieira, Departamento de Electrónica e Telecomunicações/Universidade de Aveiro.