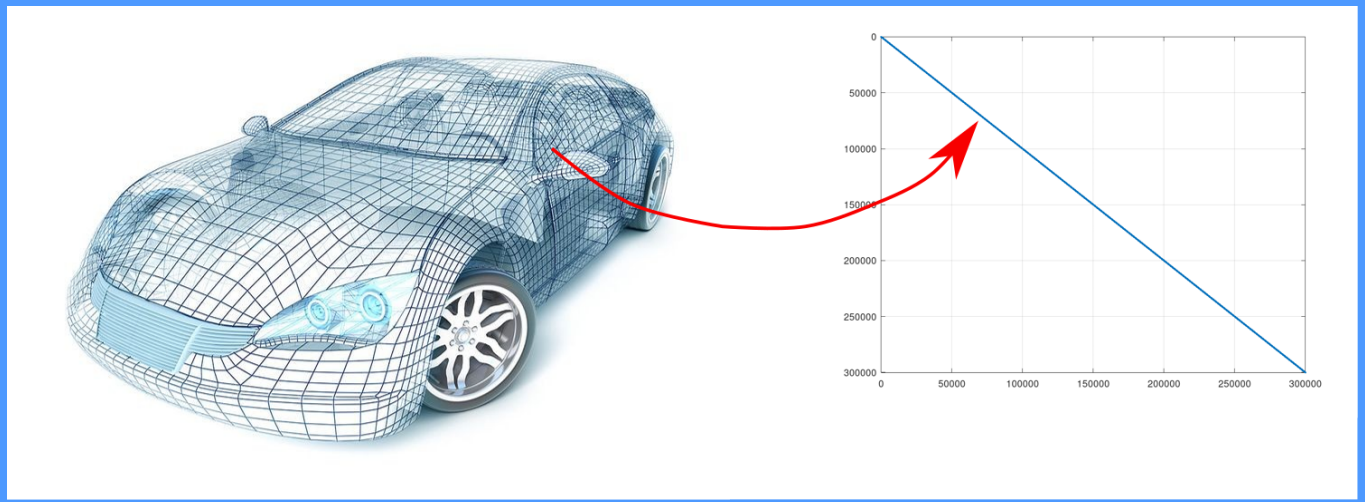
Universidad Politécnica de Cartagena

industriales
etsii UPCT

Introducción a GNU Octave



Rogelio Ortigosa Martínez
Silvestre Paredes Hernández
Departamento de Matemática Aplicada y Estadística, UPCT

Índice

Índice

Bloque 0 (I)

- Comando **cd**

> ***cd*** *Change current working directory.*
> ***cd*** *directory-spec sets the current directory to the one specified.*
> ***cd*** *.. moves to the directory above the current one.*
> ***cd****, by itself, prints out the current directory.*

- Comando **pwd**

> *pwd ()*
> *dir = pwd ()*
> *Return the current working directory.*

- Comando **path**

> path ()
> *STR = path ()*
> *STR = path (PATH1, ...)*
> *Modify or display Octave's load path.*

Bloque 0 (II)

- Comando **genpath**

> *Return a path constructed from DIR and all its subdirectories*

- Añadimos el directorio de trabajo al path:

> *addpath(genpath(pwd))*

Índice

Bloque 1 (I)

Estos estilos controlan el formato de visualización de salida para variables numéricas.

Style	Resultado	Ejemplo
short (default)	Formato corto y decimal fijo con 4 dígitos después del punto decimal.	3.1416
long	Formato decimal fijo y largo con 15 dígitos después del punto decimal para los valores de double y 7 dígitos después del punto decimal para los valores de single .	3.141592653589793
shortE	Notación científica corta con 4 dígitos después del punto decimal.	3.1416e+00
longE	Notación científica larga con 15 dígitos después del punto decimal para los valores de double y 7 dígitos después del punto decimal para los valores de single .	3.141592653589793e+00
shortG	Formato corto, fijo decimal o notación científica, cualquiera que sea más compacto, con un total de 5 dígitos.	3.1416
longG	Formato decimal largo, fijo o notación científica, cualquiera que sea más compacto, con un total de 15 dígitos para los valores de double y 7 dígitos para los valores de single .	3.14159265358979
shortEng	Notación de ingeniería corta (exponente es un múltiplo de 3) con 4 dígitos después del punto decimal.	3.1416e+000
longEng	Notación de ingeniería larga (exponente es un múltiplo de 3) con 15 dígitos significativos.	3.14159265358979e+000
+	Formato positivo/negativo con +, -y caracteres en blanco mostrados para elementos positivos, negativos y cero.	+
bank	Formato de moneda con 2 dígitos después del punto decimal.	3.14
hex	Representación hexadecimal de un número binario de doble precisión.	400921fb54442d18
rat	Proporción de números enteros pequeños.	355/113

format long
pi

format longe
pi

format short
pi

format shorte
pi

Bloque 1 (II)

- **Suma, resta, multiplicación y división** de dos números

```
1+5
a=1; b=5;
c=a+b;
c=a-b;
c=a*b;
c=a/b;
```

- Algunas **funciones elementales**

```
exp(1)          sin(pi/2)       cos(pi/2)
tan(pi)         asin(1)          acos(0)
atan(1)          exp(1)          log(2)
log(exp(1))      log10(10)       log10(100)
sqrt(2)         sqrt(-2)         factorial(6)
```

Índice

Bloque 2 (I)

- Comando **zeros**.

```
A=zeros(4);          %Creamos matriz cuadrada de ceros 4x4
A=zeros(3,6);        %Creamos matrix rectangular de ceros 3x6
V=zeros(4,1);        %Vector columna 4x1
V=zeros(1,4);        %Vector fila 1x4
```

- Comandos **rand** y **randi**.

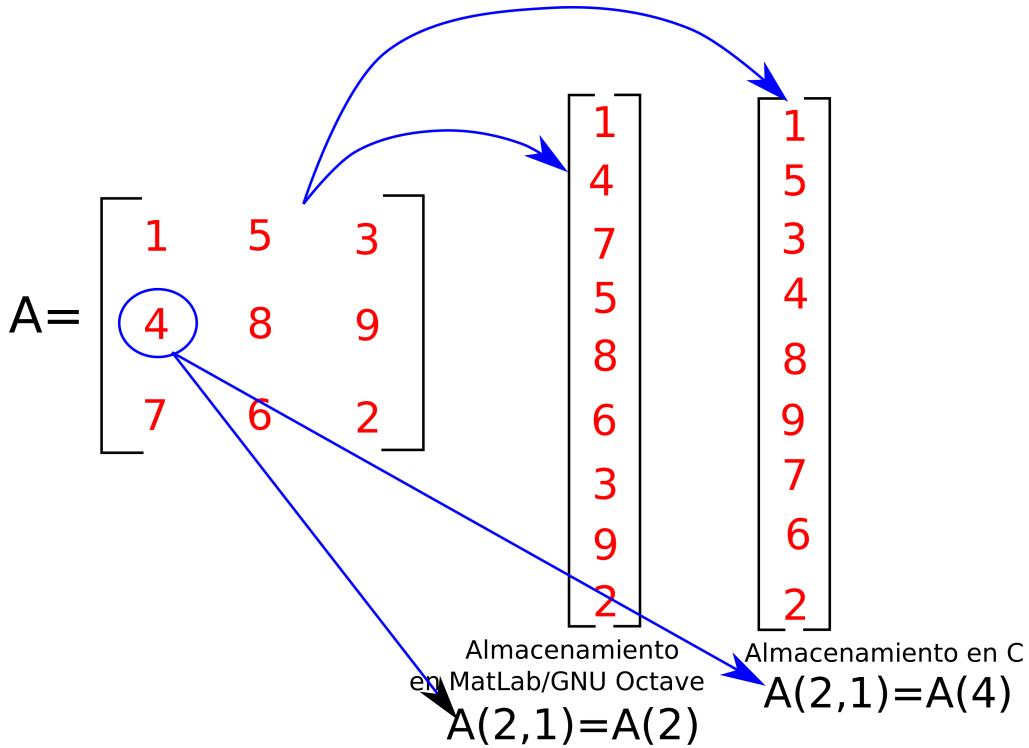
```
A=rand(3,4);         % Creamos matriz 3x4 con números reales aleatorios entre [0, 1]
A=randi(100,3,3);    % Creamos matrix 3x3 con números naturales entre [1, 100]
```

- Comando **size**.

```
size(A)  % Calcula el tamaño de la matriz o el vector A
```

Bloque 2 (II)

- **Almacenamiento interno de matrices en GNU/Octave.**



Bloque 2 (III)

- Supongamos la matriz 3x3 en la transparencia anterior. Practica en tu ordenador los siguientes comandos:

A(2,3) %Extraemos elemento (2,3) de la matriz

A(8) %Extraemos elemento 8 (para matriz 3x3 coincide con elemento (2,3))

A(:,1) %Extraemos columna 1 de la matriz, generando un vector 3x1

A(:,3) %Extraemos columna 3 de la matriz, generando un vector 3x1

A(2,:) %Extraemos fila 2 de la matriz, generando un vector fila 1x3

A(1:2,2) %Extraemos columna 2 y filas 1 y 2, generando un vector 2x1,

A(2:end,end) %Extraemos filas de la 2 hasta el final (en este caso 3) y la última columna (en este caso 3), generando en este caso un vector 2x1

Bloque 2 (IV)

- **Traspuesta** de matriz y vector: supongamos **A** una matriz 3x4 y **V** un vector 4x1:

```
A' %Calcula la traspuesta de A, resultando una matriz 4x3

transpose(A) %Hace exactamente lo mismo que el comando anterior

V' Calcula la traspuesta de V, resultando un vector fila 1x4
```

- **Producto escalar** de vectores: Sean V1 y V2 vectores de tamaño 3x1

```
V1'*V2 %Calcula el producto escalar de V1 y V2

V1(1)*V2(1) + V1(2)*V2(2) + V1(3)*V2(3) %Calcula el producto escalar de V1 y V2.
No recomentado si el tamaño de los vectores es grande
```

Bloque 2 (V)

- **Módulo** de un vector

```
norm(V1) %Calcula el módulo de V1

sqrt(V1'*V1) %Calcula el módulo de V1

sqrt(V1(1)*V2(1) + V1(2)*V2(2) + V1(3)*V2(3)) %Calcula el módulo de V1. No recomendado
```

- Sean A1 y A2 dos matrices del mismo tamaño:

```
A1 + A2 %Calcula la suma de A1 y A2

A1 - A2 %Calcula la resta de A1 y A2
```

- **Vectorización** de una matriz. Definamos A=[1 3 4;2 5 7;9 0 6]:

```
A(:) %Creamos el vector [1;2;9;3;5;0;4;7;6]
```

- Comando **eye**

```
eye(4) %Creamos matriz identidad de tamaño 4x4
```

Bloque 2 (VI)

- **Matrices simétricas**: consideremos $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ (matriz cuadrada). Una manera de conseguir una matriz simétrica a partir de $\boldsymbol{A}$ es la siguiente:

$$\boldsymbol{B} = \boldsymbol{A} + \boldsymbol{A}^T \qquad (\boldsymbol{B} = \boldsymbol{B}^T)$$

- **Matrices simétricas**: consideremos $\boldsymbol{A} \in \mathbb{R}^{N\times N}$. Otra manera de conseguir una matriz simétrica a partir de $\boldsymbol{A}$ es

$$\boldsymbol{B} = \boldsymbol{A}\boldsymbol{A}^T; \qquad (\boldsymbol{B} = \boldsymbol{B}^T)$$

- **Matrices definidas positivas**: consideremos $\boldsymbol{A} \in \mathbb{R}^{N\times N}$. $\boldsymbol{A}$ es definida positiva si:

$$\boldsymbol{x} \cdot \boldsymbol{A}\boldsymbol{x} > 0; \quad \forall \boldsymbol{x} \in \mathbb{R}^N \not\geq \boldsymbol{0}$$

- **Descomposición espectral de matrices simétricas**.

$$\boldsymbol{A} = \lambda_1 \boldsymbol{n}_1 \boldsymbol{n}_1^T + \lambda_2 \boldsymbol{n}_2 \boldsymbol{n}_2^T + \cdots + \lambda_N \boldsymbol{n}_N \boldsymbol{n}_N^T = \sum_{i=1}^{N} \lambda_i \boldsymbol{n}_i \boldsymbol{n}_i^T;$$

$$\lambda_i \in \mathbb{R}; \qquad \boldsymbol{n}_i \cdot \boldsymbol{n}_j = 0, \ \forall i \neq j$$

- **Descomposición espectral de matrices simétricas definidas positivas**.

$$\boldsymbol{A} = \lambda_1 \boldsymbol{n}_1 \boldsymbol{n}_1^T + \lambda_2 \boldsymbol{n}_2 \boldsymbol{n}_2^T + \cdots + \lambda_N \boldsymbol{n}_N \boldsymbol{n}_N^T = \sum_{i=1}^{N} \lambda_i \boldsymbol{n}_i \boldsymbol{n}_i^T;$$

$$\lambda_i > 0; \qquad \boldsymbol{n}_i \cdot \boldsymbol{n}_j = 0, \ \forall i \neq j$$

Bloque 2 (VII)

- Comando **eig**.

```
[V,Lambda]=eig(A)  %Calculamos los autovectores (V) y autovalores (Lambda) de la matriz A
```

- Comando **cond**.

```
cond(A)  %Calculamos el número de condición de la matriz A
```

- **Radio espectral** de matriz definida positiva (criterio de convergencia de métodos iterativos).

```
max(abs(Lambda(:)))
```

- Comando **det**

```
det(A);      %Calculamos el determinante de la matriz A
```

Bloque 2 (VIII)

- Comando **inv**.

```
B=inv(A)  %Calculamos los inversa de A mediante cálculo de adjuntos (determinantes) (no recomendado para matrices más grandes de 3x3)
```

- **Solución de sistemas de ecuaciones**: Sea $Ax = b$, con $A \in \mathbb{R}^{nxn}$ (matriz cuadrada de tamaño nxn) y $b \in \mathbb{R}^n$ (vector de tamaño nx1). Resolución a través del **cálculo de la inversa** de A:

```
x=inv(A)*b  %No recomendado para n>3
```

- **Solución de sistemas de ecuaciones**: Resolución mediante **solver directo de Octave**:

```
x=A\b
```

Índice

Bloque 3 (I)

- Polinomio de Taylor para función e^x

$$e^x \approx 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots = 1 + \sum_{i=1}^{N} \frac{x^i}{i!}$$

```
% Función que aproxima el valor de la función exponencial exp(x) a través de su
% polinomio de Taylor centrado en x=0. Las entradas de la función son:
% 1) x: el punto x donde aproximar exp(x)
% 2) tol: tolerancia o diferencia entre aproximaciones consecutivas
% Las salidas de la función son:
% 1) ex: aproximación de exp(x)
% 2) i: número de términos necesarios para que la diferencia entre aproximaciones
% consecutivas sea inferior (en valor absoluto) a tol

function [ex,i] = MyExponential(x,tol)
ex   = 1;
i    = 0; %Contador para número de términos en el sumatorio
diff = 1; %Damos valor a variable diff que permita entrar en el while
while diff>tol
i    = i + 1; %Incrementamos el contador del número de términos en el sumatorio
ex0  = ex; %Valor de la aproximación actual
ex   = ex0 + xˆi/factorial(i); %Añadimos un término a la aproximación
diff = abs(ex-ex0); %Calculamos diferencia entre aproximaciones consecutivas
endwhile
```