

# Algorítmica y Lenguajes de Programación

## MATLAB (iii)

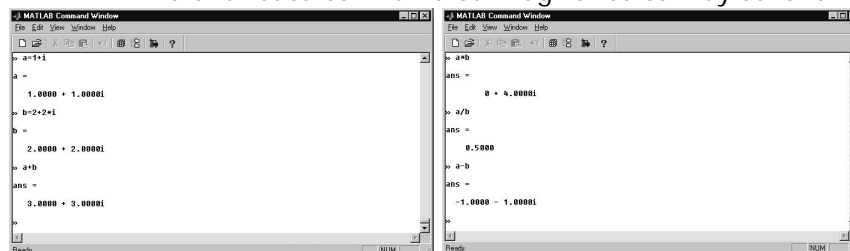
### Números complejos (i)

- La variable `i` ya está definida como la raíz cuadrada de  $-1$ . Podemos verificarlo calculando `i * i`.



```
i=i  
ans =  
-1
```

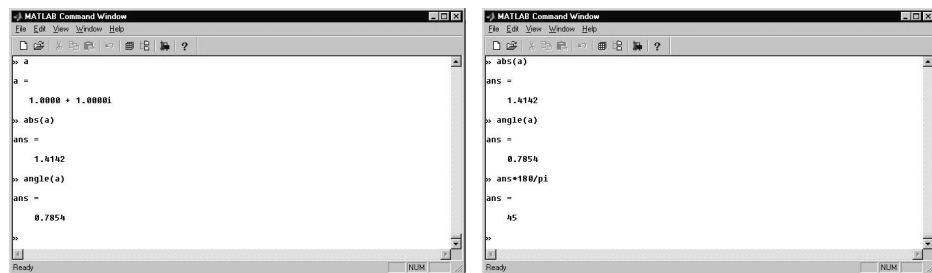
- La aritmética con números imaginarios es muy sencilla.



```
a=1+i  
a =  
1.0000 + 1.0000i  
b=2+2*i  
b =  
2.0000 + 2.0000i  
a+b  
ans =  
3.0000 + 3.0000i  
  
a=b  
ans =  
0 + 0.0000i  
a/b  
ans =  
0.5000  
a-b  
ans =  
-1.0000 - 1.0000i
```

## Números complejos (ii)

- Las funciones **abs** y **angle** nos permiten convertir un número complejo de forma rectangular a polar.
- **angle** retorna la fase en radianes, si lo convertimos a grados vemos que la respuesta es la que se esperaba.



The image shows two side-by-side screenshots of the MATLAB Command Window. The left window shows the definition of a complex number `a = 1.0000 + 1.0000i` and the calculation of its magnitude `abs(a)` resulting in `1.4142` and its phase `angle(a)` resulting in `0.7854`. The right window shows the same magnitude calculation, followed by the phase calculation `angle(a)` resulting in `0.7854`, and then the conversion of the phase to degrees `ans*180/pi` resulting in `45`.

```
MATLAB Command Window
>> a =
    1.0000 + 1.0000i
>> abs(a)
ans =
    1.4142
>> angle(a)
ans =
    0.7854

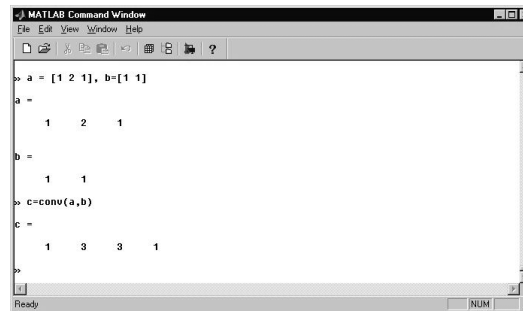
MATLAB Command Window
>> abs(a)
ans =
    1.4142
>> angle(a)
ans =
    0.7854
>> ans*180/pi
ans =
    45
```

## Polinomios. Introducción

- Matlab también proporciona herramientas para manipular polinomios y funciones racionales. Para usar estas herramientas, el polinomio debe representarse como un vector siendo el número del extremo izquierdo la potencia más alta y el número del extremo derecho la constante.
- Por ejemplo,  $x^2 + 2x + 1$  se representaría como:  
**[1 2 1]**
- La función **roots** da las raíces del polinomio mientras que **polyval** evalúa el polinomio en un valor dado. La multiplicación y división de polinomios puede llevarse a cabo con **conv** y **deconv**.

## Polinomios. Multiplicación y división (i)

- Para multiplicar  $x^2 + 2x + 1$  and  $x + 1$ , utilizamos



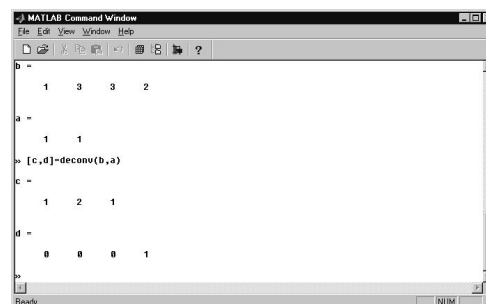
```
MATLAB Command Window
> a = [1 2 1], b=[1 1]
a =
    1     2     1
b =
    1     1
> c=conv(a,b)
c =
    1     3     3     1
```

- Nótese que **deconv** retornará dos vectores, el primero contiene los coeficientes del polinomio cociente y el segundo los coeficientes del polinomio resto.

5

## Polinomios. Multiplicación y división (ii)

- El siguiente ejemplo divide  $x^3 + 3x^2 + 3x + 2$  by  $x + 1$
- Si la parte izquierda de la ecuación no contiene dos variables la respuesta sólo consistirá en el cociente siendo descartado el resto.



```
MATLAB Command Window
> b = [1 3 3 2]
b =
    1     3     3     2
> a = [1 1]
a =
    1     1
> [c,d]=deconv(b,a)
c =
    1     2     1
d =
    0     0     0     1
```

6

## Polinomios. Cálculo de las raíces de un polinomio conociendo los coeficientes (i)

- Para calcular las raíces de un polinomio del que se conocen los coeficientes se deben introducir estos coeficientes en un vector en orden descendente. Es necesario incluir ceros en caso necesario.

```
>> p1 = [ 1 6 7 -6 -8 ]; % The roots of x^4 + 6*x^3 + 7*x^2 - 6*x - 8 = 0
>> r1 = roots(p1)
r1 =
    -4.0000
    -2.0000
    -1.0000
     1.0000
```

- Los coeficientes también pueden introducirse directamente en el comando **roots**. Se obtendría la misma respuesta anterior utilizando la expresión siguiente.

```
>> r1 = roots([ 1 6 7 -6 -8 ])
r1 =
    -4.0000
    -2.0000
    -1.0000
     1.0000
```

7

## Polinomios. Cálculo de las raíces de un polinomio conociendo los coeficientes (ii)

- El comando **roots** puede encontrar raíces imaginarias.

```
>> p2 = [ 1 -6 18 -30 25 ];
>> r2 = roots(p2)
r2 =
    1.0000 + 2.0000i
    1.0000 - 2.0000i
    2.0000 + 1.0000i
    2.0000 - 1.0000i
```

- También puede encontrar raíces repetidas. Nótese que la porción imaginaria de las raíces repetidas se muestra como cero.

```
>> p3 = [ 1 7 12 -4 -16 ];
>> r3 = roots(p3)
r3 =
    -4.0000
    -2.0000 + 0.0000i
    -2.0000 - 0.0000i
     1.0000
```

8

## Polinomios. Cálculo del valor de un polinomio de coeficientes conocidos

- La sintaxis para determinar el valor de un polinomio en cualquier punto es la siguiente.

```
>> s1a = polyval(p1, 3)
s1a =
    280
```

- Donde **p1** es el vector que contiene los coeficientes del polinomio. De forma similar, los coeficientes pueden introducirse directamente en el comando **polyval**.

```
>> s1b = polyval([1 6 7 -6 -8], 3)
s1b =
    280
```

- Se puede también calcular el valor en múltiples puntos.

```
>> z = [ 3 5 7 ];
>> s1c = polyval(p1,z)
s1c =
    280    1512    4752
```

9

## Polinomios. Cálculo de los coeficientes de un polinomio conocidas las raíces

- Para calcular los coeficientes a partir de las raíces se utiliza el comando **poly**. Los coeficientes se retornan en orden descendente.

```
>> r1 = [ -4 -2 -1 1 ]
>> t1 = poly(r1)
t1 =
    1.0000    6.0000    7.0000   -6.0000   -
    8.0000
```

- Las raíces también se pueden introducir directamente en el comando **poly**.

```
>> t2 = poly([ -4 -2 -1 1 ])
t2 =
     1     6     7    -6    -8
```

10

## Polinomios. Determinación de los coeficientes para un polinomio que ajusta un conjunto de datos (i)

- **polyfit** determina, por el método de mínimos cuadrados, el polinomio que aproxima un conjunto de datos. El orden del polinomio es indicado por el usuario.

```
>> x = [ 1.0 1.3 2.4 3.7 3.8 5.1 ];
>> y = [ -6.3 -8.7 -5.2 9.5 9.8 43.9 ];
>> coeff = polyfit(x,y,3)           % Fits a third order
polynomial
coeff =
    0.3124    1.5982   -7.3925   -1.4759
```
- Tras determinar los coeficientes del polinomio, se puede utilizar el comando **polyval** para predecir los valores de la variable dependiente para cada valor de la variable independiente.

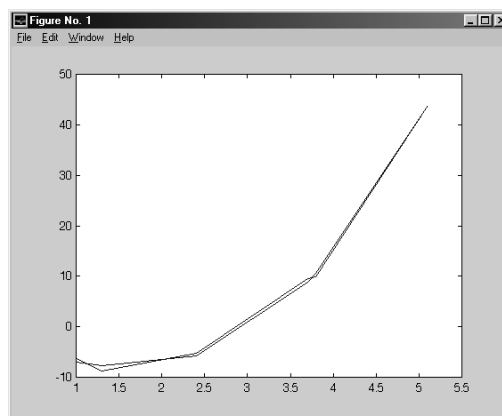
```
>> ypred = polyval(coeff,x)
ypred =
   -6.9579   -7.6990   -5.6943    8.8733   10.6506
   43.8273
```

11

## Polinomios. Determinación de los coeficientes para un polinomio que ajusta un conjunto de datos (ii)

- Los datos experimentales y predichos pueden mostrarse de forma gráfica.

```
>> plot(x,y);
>> hold on;
>> plot(x,ypred);
```

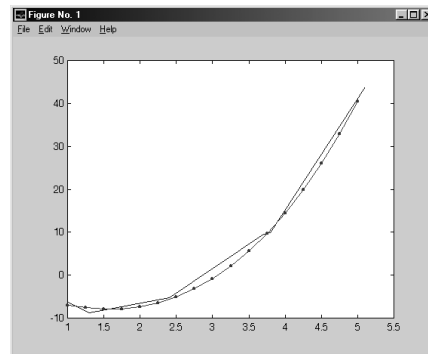


12

Polinomios. Determinación de los coeficientes para un polinomio que ajusta un conjunto de datos (iii)

- Es posible representar el polinomio de una forma más "suave":

```
x = [ 1.0 1.3 2.4 3.7 3.8 5.1 ];  
y = [ -6.3 -8.7 -5.2 9.5 9.8 43.9 ];  
coeff = polyfit(x,y,3);  
nuevosx = [1:0.25:5.1];  
ypred = polyval(coeff,nuevosx);  
plot(x,y);  
hold on;  
plot(nuevosx,ypred,'r.-');
```



13

## Resumen

- Matlab permite:
  - Manejar números complejos de manera sencilla.
  - Trabajar con polinomios y funciones racionales:
    - Multiplicar y dividir polinomios.
    - Calcular las raíces de un polinomio conociendo los coeficientes.
    - Calcular el valor de un polinomio de coeficientes conocidos.
    - Calcular los coeficientes de un polinomio conociendo las raíces.
    - Determinar los coeficientes para un polinomio que ajusta un conjunto de datos.

14