Apuntes-Examen-Practico-Matemati...

AlexMeriina

Matemáticas I

1º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingeniería
Universidad de Huelva

MATLAB:

Practica 1:

- El comando >> clear all borra todas las variables con las que se ha trabajado hasta ahora.
- Si escribimos >> clear x borrara solo la variable x.
- El comando de ayuda de Matlab es >> help .
- Al teclear >> help cos, nos proporcionar´a informacion sobre como funciona la funcion coseno y una lista de funciones relacionadas.
- Utilizamos la orden >> diary nombredelarchivo.txt para guardar (en un archivo de texto dentro del directorio actual) lo que vayamos escribiendo en cada sesion.
- Si queremos dejar de guardar los comandos y salidas en el archivo de texto creado, usaremos el comando >> diary off.
- Sqrt(x) nos da la raiz cuadrada.
- sin(x), cos(x), tan(x), … nos dan las razones trigonometricas de x.
- exp(x) nos da e^x. El numero e no esta definido previamente, asi que para obtenerlo habria que escribir exp(1).
- log(x) nos da el algoritmo neperiano de x. loga (x) , hay que aplica las transformaciones habituales: loga (x) = ln(x) /ln(a) .
- abs(x) nos da el valor absoluto de x.
- Pi es el numero pi
- inf es infinito
- NaN indica una indeterminacion.
- Eps nos da el numero mas pequeño que conoce matlab.
- Con ezplot hacemos la representacion de una grafica.
- Para expresar el resultado en diferente formatos hay que poner format(rat), o format(long), o short

Ejercicio:

A) $\sqrt{3^2+1}-\sin(\pi)$

sqrt(3^2+1)-sin(pi)

B) $e^{2^2+1}+\frac{3}{2}$

(((exp(1)^2)^2)+1)+(3/2)

C) $\log_3(9)+\ln(e)+|log_{10}(0,1)|$

(log(9)/log(3))+log(exp(1))

ans+(abs(log(0.11)))

D) $\frac{3^2+5}{\frac{1}{2}+2}$

((3^2)+5)/((1/2)+2)

E) $\sqrt[5]{\sqrt{3}+2}$

((sqrt(3))+2)^(1/5)

Ejercicio 2:

1.
   a) $\frac{(3-2i)(2+3i)}{3-4i}$

((3-2i)*(2+3i))/(3-4i)

   b) i^{2019}

i^2019

2. $(1+i)^4$

   a) En forma polar:

complex(1,1)^4

   b) En binomio de Newton

nchoosek(1,1)^4

3. Dados $z = 1 + i$ y $w = \frac{9\pi}{4} + i \ln \sqrt{2}$ se pretende hallar el argumento del número complejo z^w cuyo módulo es 1. Para ello seguiremos los siguientes pasos:

   a) Calcular r = |z| y α = arg(z), y comprobar los resultados con Matlab usando los comandos abs y angle. Recuerda usar el comando >> sym( ).

abs(1+i)

angle(1+i)

sym(1+i)

b) Hallar la expresion exponencial del modulo de z^w

(1+i)^(((9*pi)/4)+((i*log(sqrt(2)))))

c) Hallar k para que $|z^w|$ sea igual a 1.

abs((1+i)^(((9*pi)/4)+(i*log(sqrt(2)))))

Practica 2. Ecuaciones en C:

Para resolver ecuaciones con Matlab debemos primero declarar como simbolicas las variables con las que vamos a trabajar. Para ello usaremos el comando >> syms seguido de los nombres de las variables separados por espacios.

Ejercicio 1: $z^6 - 9z^3 + 8 = 0.$

Comprueba con el comando >> **solve** de Matlab que la resolución de la ecuación es correcta. Puedes usar el comando >> **pretty(ans)** para obtener una expresión más clara de la respuesta.

La solución se asigna por defecto a una variable llamada *ans*, pero si queremos llamar a la solución de otra forma, por ejemplo *sol*, debemos escribir lo siguiente:

>> sol = solve(z^6 − 9 * z^3 + 8)

Ejercicio 2:

Resuelve la ecuación $(1 + i)z^3 - 2i = 0.$

Después, utilizando Matlab, da una expresión aproximada con cuatro decimales de la parte real y de la parte imaginaria de cada solución hallada, y comprueba que las tres soluciones son correctas usando el comando >> **subs**.

ejdos=solve((1+i)*z^3-2i)

Ejercicio 3:

Indica la región del plano que satisface cada una de las siguientes condiciones:

a) $|z-2| = |z-1+i|$,

b) $|z-2| < 2$,

c) $|z-i| + |z+i| > 4$.

Para comprobar usando Matlab si una igualdad es cierta, se escribe con doble igual (==), y la respuesta será `logical 1` si es cierta y `logical 0` si no lo es.

Por ejemplo, para comprobar que $z = i$ forma parte de la región del apartado a), escribiremos >> `abs(i - 2) == abs(i - 1 + i)` y la respuesta debe ser en este caso `logical 1`. Para comprobar que $z = 1 + i$ no forma parte de la región del apartado c), escribiremos >> `abs(1 + i - i) + abs(1 + i + i) > 4` y la respuesta será `logical 0`.

a) abs(i-2)==abs(i-1+i)

b) abs(i-2)<2

c) abs(1+i-i)+abs(1+i+i)>4

Practica 3. Limites, continuidad, asintotas y graficas:

Para calcular límites con Matlab disponemos de la orden >> `limit`. Por ejemplo, si queremos calcular

$$\lim_{x\to 0} \frac{\sin(x)}{x}$$

escribiremos >> `limit(sin(x)/x, x, 0)`.

Para calcular límites en el infinito, por ejemplo

$$\lim_{x\to \infty} \frac{x}{e^{x^2}}$$

usaremos >> `limit(x/exp(x^2), x, Inf)`.

Si queremos hacer límites laterales, añadiremos la opción `'right'` o `'left'`. Por ejemplo, para calcular

$$\lim_{x\to 0^+} \frac{\sin(x)}{x^2}$$

escribiremos >> `limit(sin(x)/x^2, x, 0, 'right')`.

Ejercicio 1:

a) $$\lim_{x \to 1} \frac{x^3 - 1}{(x-1)^2}$$

limit(((x^3)-1)/(x-1)^2,x,1)

b) $$\lim_{x \to 0} \frac{6}{4 + e^{-\frac{1}{x}}}$$

limit(6/(4+(exp(-1/x))),x,0)

c) $$\lim_{x \to \infty} (x - \sqrt{x^2 + 2x})$$

limit(x-(sqrt(x^2)+2*x),x,Inf)

d) $$\lim_{x \to 0} (1 + 3\tan^2 x)^{\cotg^2 x}$$

limit ((1+3*(tan(x))^2)^((cot(x))^2),x,0)

Ejercicio 2:

Clasifica las discontinuidades de la función:

$$f(x) = \begin{cases} \dfrac{cos\frac{\pi}{x-1}}{1 + e^{\frac{1}{x}}} & \text{si} \quad x \neq 1 \text{ y } x \neq 0, \\ 0 & \text{si} \quad x = 0. \end{cases}$$

limit(cos(pi/(x-1))/(1+exp(1/x)),x,0)

limit(cos(pi/(x-1))/(1+exp(1/x)),x,0, 'left')

limit(cos(pi/(x-1))/(1+exp(1/x)),x,0, 'right')

limit(cos(pi/(x-1))/(1+exp(1/x)),x,1)

limit(cos(pi/(x-1))/(1+exp(1/x)),x,1, 'left')

limit(cos(pi/(x-1))/(1+exp(1/x)),x,1, 'right')

Ejercicio 3:

a) $f(x) = \dfrac{2x}{\sqrt{x^2-1}}$.

f=2*x/sqrt(x*2-1)

subs(x^2-1,x,-2)

subs(x^2-1,x,0)

subs(x^2-1,x,2)

subs(f,x,-2)

b) $f(x) = \ln\left(\dfrac{x+1}{x-1}\right)$.

limit(log((x+1)/(x-1)),x,1,'right')

limit(log((x+1)/(x-1)),x,1,'left')

limit(log((x+1)/(x-1)),x,1)

Estas son las asintotas verticales, tb hay que hacer las horizontales, se hacen sustituyendo por infinito.

limit(log((x+1)/(x-1)),x,inf)

limit(log((x+1)/(x-1)),x,-inf)

limit(log((x+1)/(x-1)),x,1)

c) c) $f(x) = \dfrac{1}{e^x-1}$.

Asintotas verticales

limit(1/(exp(x)-1),x,0)

limit(1/(exp(x)-1),x,0,'left')

limit(1/(exp(x)-1),x,0,'right')

Asintotas horizontales

limit(1/(exp(x)-1),x,inf)

limit(1/(exp(x)-1),x,-inf)

Para dibujar funciones en Matlab podemos usar la orden >> **ezplot**. Por ejemplo, si queremos ver en pantalla la función

$$f(x) = \frac{1}{e^x - 1}$$

con valores de x comprendidos entre -25 y 25, escribiremos

>> $\mathtt{ezplot(1/(exp(x)-1),[-25,25])}$.

Recordamos que para sustituir el valor $x = x_0$ en una función $y = f(x)$, podemos usar la orden >> $\mathtt{subs(y, x_0)}$.

Practica 4:

En Matlab se pueden crear archivos de extensión .m para definir nuevas funciones que se podrán usar como se usa cualquiera de las que ya estaban definidas en Matlab.

Para crear este tipo de archivos podemos usar el botón *New* y, en el menú que despliega, elegir *Function*. De esta manera, aparecerá una ventana en la que escribiremos las instrucciones de nuestra nueva función que deberemos *"guardar como"* con el mismo nombre que le demos a la función creada (`nombre.m`). Este nombre debe empezar con una letra y no debe coincidir con el nombre de ninguna de las funciones de las que ya dispone Matlab.

La primera línea del archivo de instrucciones es de la forma
`function[y, z, ...] = nombre(a, b, ...)`
donde `nombre` es el que le hayamos dado al archivo .m; entre corchetes, porque es opcional, aparecen la o las variables de salida (*outputs*) y entre paréntesis aparecen la o las variables de entrada (*inputs*). Si en la siguiente línea incluimos un comentario precedido de %, ésa será la explicación que dé Matlab al solicitar ayuda con el comando >> `help` seguido del nombre de la función.

Para usar la función creada, el directorio actual (o *Current Folder*) debe ser la carpeta en la que se haya guardado el archivo de extensión .m creado. Para que al ejecutar nuestro archivo de función no aparezca, además de las salidas que hayamos programado, la respuesta `ans`, debemos poner punto y coma al final de la orden que escribimos para usar la función.

La sintaxis usada para `if`, `for` y `while` es la siguiente:

◇ `if` *expresión lógica*
  instrucciones
 `elseif` *expresión lógica*
  instrucciones
 `elseif` *expresión lógica*
  instrucciones
 ...
 `else`
  instrucciones
 `end`

◇ `for` *índice = inicio : incremento: final*
  instrucciones

Ejemplo 1.- Dados los coeficientes de una ecuación de segundo grado, vamos a crear un archivo de función que nos diga el número de soluciones reales que tiene la ecuación, y cuáles son dichas soluciones.

```
function[x1, x2] = ecu2(a, b, c)
%Para resolver la ecuación de coeficientes a, b y c escribiremos la orden >> ecu2(a, b, c);
  Delta = b^2 - 4 * a * c;  %Se calcula el discriminante
    if Delta < 0
      disp('No hay soluciones reales ')
    elseif Delta > 0
      x1 = (-b - sqrt(Delta))/(2 * a);
      x2 = (-b + sqrt(Delta))/(2 * a);
      disp('Hay dos soluciones reales '), x1, x2
    else
      x1 = -b/(2 * a);
      x2 = -b/(2 * a);
      disp('Solo hay una solucion real '), x1, x2
    end
end
```

1. Resuelve las siguientes ecuaciones de segundo grado mediante la función ecu2:

   (1.a) $x^2 - 4x - 5 = 0$

   (1.b) $x^2 - x + 1 = 0$

   (1.c) $x^2 + 6x + 9 = 0$

Ejemplo 2.- Vamos a crear una función que corresponda a

$$f(x) = \begin{cases} \dfrac{e^x - 1}{x} & \text{si} \quad x \neq 0 \\ 1 & \text{si} \quad x = 0 \end{cases}$$

```
function[y] = fun(x)
   if x == 0
     y = 1;
   else
     y = (exp(x) - 1)/x;
   end
end
```

Ya podemos usar esta función como cualquier otra de Matlab simplemente llamándola por su nombre. Así, por ejemplo, si escribimos >> `fun(2)`, nos dará el valor que toma la función para $x = 2$.

Para usarla como input de otra función, debemos escribir su nombre como cadena de caracteres, es decir, `'fun'`. Por ejemplo, >> `ezplot('fun', [-2, 5])` dará su representación gráfica en el intervalo indicado.

Ejemplo 3.- Basado en el Teorema de Bolzano, el *Método de la Bisección* aproxima una raíz de una función localizada en un intervalo. Vamos a crear un archivo llamado `bisection.m`:

```
function[c, err, yc, iter] = bisection(f, a, b, delta)
% Para usar esta función bisection primero tenemos que haber definido la f,
% por ejemplo: >> f1(x) = sin(x) - 1/2
% Escribiremos las entradas de esta forma: >> bisection(f1, 0, 2, 0.001);
% siendo en este caso el intervalo [a, b] = [0, 2]
% y delta=0.001, la precisión, es decir, tres cifras decimales exactas.
% c es el cero buscado
% yc será el valor de f en c que debe ser muy próximo a 0
% err es el error de la aproximación de c
% iter es el número de iteraciones realizadas para dar c.
  ya = feval(f, a);
  yb = feval(f, b);
    if ya * yb > 0
      disp('No se puede aplicar el Teorema de Bolzano en este intervalo.')
      return
    end
  err = b - a;
  iter = 0;
    while err >= delta
      c = (a + b)/2;
      yc = feval(f, c);
       if yc == 0
        a = c;
        b = c;
       elseif yb * yc > 0
        b = c;
        yb = yc;
       else
        a = c;
        ya = yc;
       end
      err = b - a;
      iter = iter + 1;
    end
  disp('El cero es '), c
  err = abs(b - a);
  disp('El error es menor que '), err
  yc = vpa(feval(f, c), 10);
  disp('El valor de la funcion en c es '), yc
  disp('El numero de iteraciones ha sido '), iter
end
```