UNIVERSIDAD CATÓLICA DE LA SANTÍSIMA CONCEPCIÓN
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE MATEMÁTICA Y FÍSICA APLICADAS

Ecuaciones Diferenciales (in1008c)

Laboratorio 3

Cálculo Simbólico y Funciones Matemáticas

Hasta ahora hemos visto como se utiliza Matlab para realizar manipulaciones con variables que almacenan vectores o matrices de **valores fijos**, esto es, sólo procedimientos aritméticos de esas estructuras (principal objetivo del software). En este laboratorio trataremos la definición y el uso de variables en un sentido matemático usual, como literales de expresiones algebraicas.

1 Cálculo simbólico.

Matlab posee un potente motor de cálculo simbólico, llamado Mupad, que permite realizar operaciones con expresiones algebraicas escalares, vectoriales y matriciales compuestas de coeficientes y literales (símbolos o letras que representan valores que son desconocidos o que pertenecen a un conjunto preestablecido) y que entregan resultados del mismo tipo.

Para definir las variables x e y en Matlab, con en el sentido expuesto más arriba, se utilizan alguna de las siguientes sentencias.

```
>> syms x y
```

```
>> x = sym('x');
>> y = sym('y');
```

La sentencia de la izquierda permite una forma rápida de definir varias variables al mismo tiempo, cuyo nombre como variable de Matlab coincida con el literal que guarda, mientras que el método de la derecha se utiliza cuando deseamos definir una única variable simbólica. Para definir variables simbólicas correctamente sólo se debe saber éstas deben comenzar con una letra y pueden contener sólo símbolos alfanuméricos y guiones bajos.

Si se pone atención al *Workspace* luego de ingresar una de las dos opciones presentadas en el *Command Window* se observa que las variables `x` e `y` son de tipo `sym`.

Las variables simbólicas muestran su verdadera utilidad cuando se construyen otras expresiones que dependen de ellas, las cuales pasan a ser automáticamente variables simbólicas.

Ejemplo: Las primeras expresiones algebraicas que son estudiadas dentro del contexto del aprendizaje de las matemáticas son los polinomios. La siguiente instrucción (luego de haber ingresado a x como tipo *sym*) permite definir el polinomio cuadrático $p(x) = 3x^2 - 2x + 1$ introduciendo la linea de comando:

```
>> p = 3*x^2-2*x+1
p =
3*x^2 - 2*x + 1
```

Se debe tener cuidado en escribir todos los productos y otras operaciones elementales de forma explícita porque de lo contrario se puede producir un error.

Operaciones con expresiones algebraicas.

Matlab permite efectuar de forma analítica una gran variedad de cálculos con expresiones simbólicas tales como operaciones elementales, de sustitución, de límite, de diferenciación, de integración, de simplificación, transformadas y resolución de ecuaciones. La siguiente tabla 1 muestra algunas de las capacidades del software.

Función	**Utilidad**
`simplify(f)`	Simplifica la expresión simbólica `f`.
`expand(f)`	Expande la expresión simbólica `f`.
`subs(f,a,b)`	Substituye en la expresión simbólica `f` la sub-expresión `a` por la expresión `b`.
	Se puede utilizar para evaluar la expresión simbólica `f`.
`solve(eq,var)`	Resuelva la ecuación simbólica `eq` respecto a la incógnita `var`.
`limit(f,x,a)`	Obtiene el límite cuando la variable `x` tiende al número `a` en la expresión `f`.
`diff(f,x,n)`	Derivada de orden `n` de la expresión `f` respecto de la variable `x`.
`int(f,x)`	Integral indefinida de la expresión `f` respecto de la variable `x`.
`int(f,x,a,b)`	Integral definida de la expresión `f` respecto de la variable `x` entre `a` y `b`.
`taylor(f,x,a)`	Polinomio de Taylor de grado 5 de la expresión `f` con `x` entorno a `x = a`.
`pretty(f)`	Presenta en formato matemático usual la expresión `f`.

Tabla 1: Algunas operaciones con expresiones simbólicas.

Ejemplo 1: El siguiente código corresponde al cálculo de la derivada de la expresión simbólica en función de `x` dada por $f = \tan(x)$ usando la definición mediante el límite del cociente incremental.

```
>> syms x h
>> f = tan(x);
>> cif = (subs(f,x,x+h)-f)/h;
>> pretty(cif)
tan(h + x) - tan(x)
-------------------
         h
>> df = limit(cif,h,0)
df =
tan(x)^2 + 1
>> Df = diff(f,x)
Df =
tan(x)^2 + 1
```

Ejemplo 2: Para verificar que la función $y(x) = 1 + ce^{-x}$ es una solución explícita de EDO:

$$y' + y = 1$$

```
>> syms x y(x) c
>> y = 1+c*exp(-x)
>> test =simplify(subs(diff(y,x,1)+y == 1,y))
test =
TRUE
```

Ejemplo 3: En vista del ejemplo anterior. Para obtener el valor de la constante c, tal que $y(0) = 2$:

```
>> syms x y(x) c
>> y = 1+c*exp(-x)
>> CI = y(0)==2;
>> C =solve(CI,c)
>> y_p = subs(y,C,c)
```

2 Funciones Matemáticas.

En Matlab se pueden definir funciones matemáticas de diferentes maneras y con diferentes utilidades. Aquí veremos cuatro formas de ingresar este tipo de estructuras.

2.1 Funciones de tipo `inline`.

Se conocen como funciones inline a las funciones que, al compilar, no son llamadas en el código objeto, sino insertadas en la sección del código donde se las llame. No necesitan que se explicite previamente las variables de las cuales dependerá sino sólo al momento de crearlas. Para definirlas se usa la instrucción `inline` en la ventana de comando o en el cuerpo de un rutero.

Ejemplo: Supongamos que deseamos definir la función $f(x, y) = \cos(x^2 + y^2)$. Escribimos entonces:

```
>> f = inline('cos(x.^2+y.^2)','x','y')
f =
     Inline function:
     f(x,y) = cos(x.^2+y.^2)
>> f(sqrt(pi),exp(1))
ans =
   -0.4484
```

Como se observa en el ejemplo, evaluar una función tipo inline es muy simple, principalmente porque se utiliza la misma notación que en matemática analítica. Sin embargo este tipo de funciones presenta la desventaja que no se consideran como expresiones simbólicas y por tanto no se pueden realizar las operaciones usuales de límite y otras vistas en la sección anterior.

Note que para definir la operación de exponenciación se ha utilizado el punto. Esto permite evaluar la función de forma vectorial, es decir, se pueden entregar vectores (de la misma longitud) como argumentos este tipo de funciones.

2.2 Funciones de tipo anónimo (@).

Las funciones anónimas constituyen una forma muy flexible de funciones matemáticas sobre la marcha, bien en la línea de comando, bien en una línea cualquiera de una función o de un fichero de Matlab.

Ejemplo: Definamos la función vectorial dada por:

$$F(x,y) = \begin{pmatrix} x^2 - y^2 \\ \ln\left(\cos(xy) + 2\right) \end{pmatrix}$$

Escribimos entonces en la ventana de comandos (o en un rutero) las instrucciones:

```
>> F = @(x,y)[x.^2-y.^2; log(cos(x.*y)+2)]
F =
    @(x,y)[x.^2-y.^2;log(cos(x.*y)+2)]
>> F(0,1)
ans =
   -1.0000
    1.0986
```

Al igual que el caso anterior, evaluar este tipo de funciones se realiza mediante la notación matemática usual y también admiten argumentos vectoriales (vectores de las mismas dimensiones). Nuevamente la desventaja está en que no se pueden aplicar operaciones simbólicas sobre este tipo de funciones.

En el contexto de los laboratorios de Ecuaciones Diferenciales, las funciones anónimas serán las más utilizadas por su capacidad de definir funciones vectoriales con el formato simple.

2.3 Funciones simbólicas.

Las expresiones simbólicas que se definen mediante el uso de variables simbólicas pueden considerarse como funciones de las mismas, sin embargo, Matlab admite la definición explícita de funciones escalares o vectoriales de variables simbólicas que son tratadas como tal por el software por lo que para ellas se permiten todas las operaciones simbólicas vistas en el tema anterior.

Ejemplo: Definamos la función escalar dada por:

$$f(x) = \text{Arctan}\left(\frac{x^3}{x^2+1}\right).$$

En Matlab se ingresan las instrucciones:

```
>> syms x
>> f(x) = atan(x^3/(x^2+1))
f(x) =
atan(x^3/(x^2 + 1))
>> f1 = f(1)
f1 =
atan(1/2)
>> double(f1)
ans =
    0.4636
>> diff(f,x)
ans(x) =
((3*x^2)/(x^2 + 1) - (2*x^4)/(x^2 + 1)^2)/(x^6/(x^2 + 1)^2 + 1)
```