6. Matemáticas Simbólicas

Hasta ahora se ha utilizado a Matlab para realizar cálculos de tipo numérico con mucha eficiencia y sencillez. En esta sección se estudiarán sus capacidades de manipulación de expresiones matemáticas sin utilizar cálculos numéricos; es decir, el tratamiento de las llamadas ***expresiones simbólicas***.

Este tema guarda íntima relación con la llamada ***álgebra simbólica***, que se usa matemáticamente para factorizar, simplificar, determinar soluciones, derivar, integrar, etc.

Las funciones simbólicas de Matlab, agrupadas en lo que se conoce como el "***symbolic math toolbox***" (caja de herramientas de matemáticas simbólicas), se basan en el software ***Maple V*** de la compañía Waterloo Maple Software, Inc. De Canadá.

La instrucción para crear, acceder o modificar un escalar, vector o matriz simbólica es:

[132] >>**sym (** *x* **)** ; ó >>**sym ('***expresión***')** ;

Este comando convierte al escalar, vector o matriz *x* a su forma simbólica. También se usa para convertir a la **'***expresión***'** (entre apóstrofes) en simbólica.

>>**sym ('** [s_{11},s_{12},...,s_{1n}; s_{21},s_{22},...; ...,s_{mn}] **')** ;

Esta versión de *sym*, crea una matriz simbólica de tamaño *m* x *n* , usando los elementos simbólicos s_{11}, s_{12}, ..., s_{mn}.

Si de actualizar una matriz simbólica se trata, tenemos que:

>>**sym (** *x* **,** *i* **,** *j* **,** '*expr*' **)** ;

Es la versión simbólica para *x* (*i* , *j*) = 'expr' .

Así como también:

>>y = **sym (** *x* **,** *i* **,** *j* **)** ;

Es la version simbólica para y = *x* (*i* , *j*) .

Para ambos comandos anteriores, *i* y *j* representan las coordenadas de fila y columna del elemento simbólico deseado.

6.1 Expresiones simbólicas

Las matemáticas simbólicas se basan en las llamadas expresiones simbólicas que no son más que ***cadenas de caracteres encerradas entre apóstrofos***.

Por ejemplo:

' x * exp (2 * x) **'**

' sin (x) * cos (y) **'**

' x ^ 4 – 3 * x ^ 3 + x ^ 2 – 4 * x +1 **'**

' 2 * x ^ 2 + 3 * y ^ 3 – 4 **'**

' (x – 2) ^ 2 + (y – 3) ^ 2 **'**

Cuando la expresión simbólica tiene más de una variable, es importante saber cual es la variable que Matlab considera como la ***variable independiente***.

En muchas funciones de Matlab el usuario puede especificar la variable independiente a usar, especificándola como un parámetro más de la llamada a la función.

Si el usuario no especifica ninguna variable independiente, Matlab la escoge de manera automática, de acuerdo con los siguientes criterios:

a) La que sea una sola letra minúscula, distinta de *i* y *j*, que alfabéticamente esté más cerca a *x*.

b) Si hubiera empate, se escoge la letra más adelante en el alfabeto.

c) Si no hay tal letra, se escoge a *x.*

❖ **Para obtener la variable independiente**

Dada una expresión simbólica, existe una función Matlab que nos dice a cuál de las variables se le considera la independiente:

[133] >>**symvar ('***exp_simb***')** ;

Donde '*exp_simb*' es la expresión simbólica para la cuál queremos saber su variable independiente.

❖ **Para graficar funciones simbólicas**

Por otro lado, Matlab incluye una función con la que podemos graficar una expresión simbólica de una variable. Dicho comando es:

[134] >>**ezplot ('***exp_simb***' , [***xmin* **,** *xmax* **])** ;

Esta función genera la gráfica de '*exp_simb*' donde la variable independiente varía entre *xmin* y *xmax*. Si no se especifica un rango de valores, se asume que la variable varía en el intervalo [-2 π , 2 π].

6.2 Simplificación de expresiones simbólicas

Para manipular las expresiones simbólicas, Matlab ofrece varias funciones que nos permiten simplificar expresiones, expandir términos ó factorizar. Algunas de éllas son:

❖ **Para agrupar términos semejantes**

[135] >>**collect ('***exp_simb***')** ;

ó

>>**collect ('***exp_simb***' , ' *v* ')** ;

En su primera forma, esta función sirve para agrupar términos semejantes de la expresión simbólica '*exp_simb*'.

En la segunda forma agrupa términos semejantes de '*exp_simb*' pero con respecto a la variable independiente ' *v* ' especificada.

❖ **Para expandir una expresión**

[136] >>**expand ('***exp_simb***')** ;

Esta función realiza una expansión de la expresión simbólica '*exp_simb*'.

❖ **Para factorizar una expresión**

[137] >>**factor ('***exp_simb***')** ;

Esta función intenta factorizar a '*exp_simb*'.

❖ **Para simplificar una expresión**

[138] >>**simple ('***exp_simb***')** ;

Esta función simplifica la forma de '*exp_simb*' a una forma más corta, si fuera posible hacerlo.

Por ejemplo, dadas las siguientes expresiones simbólicas,

e1 = ' 3 * x ^ 3 – 2 * x ^ 2 + 3 * x - 1 '

e2 = ' (x – 2) ^ 2 + (y + 1) ^ 2 '

e3 = ' sqrt (x ^ 2 * y ^ 3) '

e4 = ' 3 * y ^ 3 / (3 * x * y) '

El resultado de aplicar las funciones es:

Función:	*Resultado*:
factor (e1)	3 * x ^ 3 – 2 * x ^ 2 + 3 * x - 1
expand (e2)	x ^ 2 – 4 * x + 5 + y ^ 2 + 2 * y
collect (e2)	x ^ 2 – 4 * x + 4 + (y + 1) ^ 2
collect (e2, 'y')	y ^ 2 + 2 * y + (x – 2) ^ 2 + 1
simple (e4)	y ^ 2 / x

6.3 Operaciones con expresiones simbólicas

Las operaciones aritméticas normales pueden ser aplicadas a expresiones simbólicas. La toolbox de matemáticas cuenta con varias funciones para llevar a cabo aritmética simbólica y conversiones, veamos:

❖ **Despliegue con tipografía matemática**

[139] >>**pretty ('***exp_simb***')** ;

Esta función despliega a '*exp_simb*' con una forma de salida de tipografía matemática mas clara.

❖ **Convertir de simbólico a valor numérico**

[140] >>**numeric ('***exp_simb***')** ;

Esta función convierte a '*exp_simb*' de una forma simbólica a valor numérico. Con la única restricción de que la expresión simbólica '*exp_simb*' no debe de contener variables simbólicas.

❖ **Convierte un vector a polinomio simbólico**

[141] >>**poly2sym (** *v* **)** ;

Esta función convierte el vector *v* de coeficientes que representa a un polinomio a un polinomio simbólico.

❖ **Convierte polinomio simbólico a vector**

[142] >>**sym2poly (** '*exp_simb*' **)** ;

Esta función convierte el polinomio simbólico representado por '*exp_simb*' a un vector de coeficientes que representa a un polinomio.

❖ **Realizar operaciones aritméticas simbólicas**

Suponemos que *e1* y *e2* son dos expresiones simbólicas válidas, definidas previamente. Se tiene, entonces las funciones:

[143] >>**symadd (** *e1* **,** *e2* **)** ;

Realiza la suma simbólica *e1* **+** *e2*.

[144] >>**symsub (** *e1* **,** *e2* **)** ;

Obtiene la resta simbólica *e1* **-** *e2*.

[145] >>**symmul (** *e1* **,** *e2* **)** ;

Regresa el producto simbólico *e1* ***** *e2*.

[146] >>**symdiv (** *e1* **,** *e2* **)** ;

Proporciona el cociente simbólico *e1* **/** *e2*.

[147] >>**sympow (** *e1* **,** *p* **)** ;

Esta función realiza la elevación a una potencia simbólica, *e1* ^ *p* .

Por ejemplo, dadas las siguientes expresiones simbólicas,

e1 = ' 1 / (x + 1) '

e2 = ' x ^ 2 – 2 * x + 4 '

e3 = ' (x – 2) * (x + 3) '

El resultado de aplicar las funciones es:

Función:	*Resultado*:
symadd (e1 , e3)	1 / (x + 1) + (x - 2) * (x + 3)
sympow (e3 , 2)	(x - 2) ^ 2 * (x + 3) ^ 2
symmul (e1 , e2)	1 / (x + 1) * (x ^ 2 – 2 * x + 4)

6.4 Resolución de ecuaciones

Matlab cuenta con funciones que permiten la solución de una sola ecuación simbólica, de un sistema de ecuaciones simbólicas y de ecuaciones diferenciales simbólicas.

A continuación se describen éstas funciones:

❖ **Solución de una ecuación o un sistema de ecuaciones**

[148] >>**solve (** *f* **)** ;

Este comando resuelve la ecuación simbólica *f*, obteniéndose su variable simbólica.

Si *f* no es una ecuación, sino una expresión simbólica, el comando resuelve para *f* = 0 obteniéndose su variable simbólica.

La variante:

>>**solve (** *f1 , f2 , … , fn* **)** ;

Resuelve el sistema de ecuaciones representado por *f1 , f2 , … , fn.*

Por ejemplo:

>>solve (' x^3 – 2.5 * x^2 – 2 * x + 1.5 ') ;

Regresa los valores: 3, 0.5 y –1 en un vector columna que son las raíces del polinomio simbólico.

❖ **Solución de ecuaciones diferenciales**

Una ***EDO*** (**e**cuación **d**iferencial **o**rdinaria) de primer grado puede definirse como:

$$Y' = \frac{dy}{dx} = g(x, y)$$

Donde *x* es la variable independiente y *y* es una función de *x*. Una solución a una *EDO* de primer grado es una función $y = f(x)$, tal que $f'(x) = g(x, y)$.

Para hallar la solución de este tipo de ecuaciones diferenciales hay que recurrir a la integración con el fin de poder hallar *y* a partir de *y* ' .

La solución a una *EDO* generalmente no es única, sino una familia de funciones. Por ello es necesario especificar lo que se conoce como ***condición inicial*** ó ***condición a la frontera*** para encontrar una solución única.

Existen métodos de aproximación numérica que iteran hacia la solución de las ecuaciones diferenciales. Sin embargo, si existe una solución analítica es preferible usar ésta que la aproximada.

Matlab ofrece una función para la solución de ecuaciones EDO con condiciones iniciales, que es:

[149] >>**dsolve (** '*ecuación*' **,** '*condición*' **)** ;

Este comando resuelve simbólicamente la *EDO* especificada en '*ecuación*'.

El argumento opcional '*condición*' especifica una condición inicial o condición de frontera.

Al escribir la *EDO* simbólica hay que tener en cuenta que hay que respetar la nomenclatura siguiente: Se utiliza una letra **D** mayúscula para denotar derivación respecto a la variable independiente. Una D seguida de un dígito denota derivación repetida.

Entonces, **D*y*** representa a d*y* / d*x*, y tal como se especificó, **D2*y*** representa d^2y / dx^2.

Por ejemplo, si se desea resolver

$$y' = 3 * x * \sin^2 (y)$$

con:

$$y (0) = \pi / 4$$

En Matlab escribiríamos:

```
>>dsolve ( ' Dy = 3*x*sin(y)^2 ' , ' y(0) = pi/4 ' ) ;
```

El comando anterior, después de ejecutarse, regresa la siguiente solución para la *EDO*:

```
- atan ( 2 / ( 3 * x ^ 2 – 2 ) )
```

También podríamos haber definido cada parte por separado como expresiones simbólicas en variables distintas y después ejecutar el comando *dsolve*.

Para el ejemplo que se resolvió anteriormente, podemos escribir en Matlab:

```
>>e = sym ( ' Dy = 3 * x * sin ( y ) ^ 2 ' ) ;
>>c = sym ( ' y ( 0 )  = pi / 4 ' ) ;
>>dsolve ( e , c ) ;
```

Esto nos despliega exactamente la misma respuesta que antes.

En un mismo comando *dsolve* podemos especificar más de una ecuación diferencial y cada una con su respectiva condición inicial. La nomenclatura sería:

$$>>\text{dsolve} (edo_1 , edo_2 , \ldots, edo_n , ci_1, ci_2, \ldots, ci_n) ;$$

En este caso, cada *EDO* se debe escribir con variables distintas y ocupar las mismas variables en su correspondiente condición inicial; pues esta es la manera en que Matlab relaciona a cada EDO con sus condiciones frontera.

De igual forma que antes, cada *EDO* y condición inicial se deben de escribir entre apóstrofos o bien ocupar la opción de definir variables separadas con el comando *sym* para cada parte de las ecuaciones.

6.5 Derivación de expresiones simbólicas

En este caso consideraremos la situación de obtener la derivada de una expresión simbólica, de acuerdo con las reglas del cálculo tradicional.

En matlab se usa la función ***diff*** para determinar la derivada de una expresión simbólica.

Existen cuatro variantes para emplear el comando *diff* para obtener las derivadas simbólicas:

[150] >>**diff (** '*exp_simb*' **)** ;
ó
>>**diff (** '*exp_simb*' **,** '*v* ' **)** ;
ó
>>**diff (** '*exp_simb*' **,** *n* **)** ;
ó
>>**diff (** '*exp_simb*' **,** '*v* ' **,** *n* **)** ;

La primera versión, regresa la derivada de la expresión simbólica '*exp_simb*' respecto a la variable independiente predefinida, que es *x*.

La segunda opción, regresa la derivada pero con respecto a la variable especificada en '*v* ' .

La tercera forma regresa la *n*-ésima derivada de la expresión con respecto a *x*, que es la predefinida.

La última opción, conjunta todo lo anterior ya que regresa la *n*-ésima derivada de la expresión con respecto a la variable especificada en '*v* ' .

Por ejemplo, si se tienen las funciones:

f1 = ' 3 * x ^ 3 – 2 * x ^ 2 + 3 * x - 1 '

f2 = ' (x – 2) ^ 2 + (y + 1) ^ 2 '

f3 = ' x ^ 4 – 2 * x ^ 3 + 4 * x ^ 2 - 9 '

f4 = ' 3 * y ^ 3 / (3 * x * y) '

El resultado de aplicar las derivadas es:

Derivada:	*Resultado*:
diff (f1)	9 * x ^ 2 – 4 * x + 3
diff (f2 , ' y ')	2 * y + 2
diff (f3 , 2)	12 * x ^ 2 – 12 * x + 8
diff (f4, ' y ' , 2)	2 / x

6.6 Integración de expresiones simbólicas

Aquí se considera la situación de obtener la integral de una expresión simbólica, de acuerdo con las reglas del cálculo tradicional.

En matlab se usa la función ***int*** para determinar la integral de una expresión simbólica '*exp_simb*'. Su propósito es encontrar una expresión simbólica *F* de tal manera que se cumpla que *diff* (*F*) = '*exp_simb*'.

Si por alguna razón Matlab no es capaz de obtener la integral, simplemente regresa la expresión simbólica original, indicando que no le fue posible resolver el problema.

La función *int* puede utilizarse en alguna de las siguientes variantes:

[151] >>**int (** '*exp_simb*' **)** ;
ó
>>**int (** '*exp_simb*' **,** '*v* ' **)** ;

ó también,

>>**int (** '*exp_simb*' **,** *a , b* **)** ;

ó

>>**int (** '*exp_simb*' **,** '*v* ' **,** *a , b* **)** ;

ó

>>**int (** '*exp_simb*' **,** '*m* ' **,** '*n* ' **)** ;

La primera versión, regresa la integral de la expresión simbólica '*exp_simb*' respecto a la variable independiente predefinida, que es *x*.

La segunda opción, regresa la integral pero con respecto a la variable especificada en '*v* ' .

La tercera forma regresa la integral de la expresión con respecto a *x*, que es la predefinida, pero evaluada para los límites [*a*,*b*], donde *a* y *b* pueden ser expresiones numéricas.

La cuarta variante regresa la integral pero con respecto a la variable especificada en '*v* ' y evaluada para los límites [*a*,*b*], donde *a* y *b* pueden ser expresiones numéricas.

La última opción regresa la integral de la expresión con respecto a *x*, que es la predefinida, pero evaluada para los límites [*m*,*n*], donde *m* y *n* son expresiones simbólicas.

Es muy recomendable que al efectuar derivación ó integración simbólicas siempre se tenga muy en claro, y se defina, cuál será la variable independiente para evitar malas interpretaciones de las expresiones.

Por ejemplo, si se tienen las funciones:

f1 = ' 3 * x ^ 2 – 2 * x + 4 '

f2 = ' (x – y) ^ 2 + (y + 1) ^ 2 '

f3 = ' cos (4 * x – 2) '

f4 = ' sqrt (x) '

El resultado de aplicar las integrales es:

Integral:	*Resultado*:
int (f1)	x ^ 3 – x ^ 2 + 4 * x
int (f2 , ' y ')	-1/3 * (x-y)^3 + 1/3 * (y+1)^3
int (f3 , 2 , 4)	¼ * sin(14) - ¼ * sin(6)
int (f4, ' a ' , ' b ')	2/3 * b^(3/2) - 2/3 * a^(3/2)