Algorítmica y Lenguajes de Programación

MATLAB (ii)

Cálculo con MATLAB. Introducción

- En esta lección se presentarán algunos de los aspectos principales relacionados con el uso de MATLAB para el cálculo.
- El concepto central es el de **función**. En primer lugar veremos cómo se representan funciones y después las formas en que se puede trabajar con las mismas.
- MATLAB permite, básicamente, lo siguiente:
  - cálculo simbólico,
  - cálculo numérico y
  - visualización
- A lo largo de esta lección se presentarán ejemplos concretos para cada uno de estos conceptos.
- Por el momento, trataremos con funciones de una sola variable.

2

Cálculo con MATLAB. Funciones y cálculo diferencial (i)

- Hay dos nociones distintas aunque relacionadas que son importantes para el Cálculo:
  - Una son las expresiones simbólicas como *sen x* o x^2.
  - La otra son las "reglas" o algoritmos que permiten obtener una salida numérica a partir de unas entradas también numéricas.
- La segunda "definición" es más general; sirve, por ejemplo, para definir una función *f(x)* como x^2 si *x* es negativa o 0 y *sen x* si *x* es positiva.
- Por otro lado, cualquier expresión simbólica implica una regla de evaluación. Es decir, si sabemos que $f(x)=x^2$ entonces sabemos que $f(4)=4^2=16$.
- En MATLAB, la diferencia fundamental entre una función y una expresión simbólica radica en que una función puede ser invocada con argumentos y una expresión simbólica no.
- Por otra parte, una expresión simbólica puede ser derivada mientras que una función no puede.
- En MATLAB las funciones son creadas de dos formas: como ficheros .m y como funciones *inline*.

3

Cálculo con MATLAB. Funciones y cálculo diferencial (ii)

- La forma típica de definir una expresión simbólica es la siguiente:

```
syms x
f=x^2-sin(x)
f =
x^2-sin(x)
```

- Las dos líneas siguientes muestra que podemos derivar *f*, pero que no podemos evaluarla, al menos de una forma obvia. Nótese que MATLAB reconoce la variable. En caso de que haya varias variables simbólicas, podemos especificar aquella respecto a la cual queremos derivar.

```
diff(f)
ans =
   2*x-cos(x)
f(4)
??? Index exceeds matrix dimensions.
```

- Podemos evaluar *f(4)* sustituyendo *x* por *4* de la siguiente forma:

```
subs(f,x,4)
ans =
   16.7568
```

4

Cálculo con MATLAB. Funciones y cálculo diferencial (iii)

- Podemos también convertir *f* en una función *inline* con el comando:

```
fin=inline(char(f))
fin =
   Inline function:
   fin(x) = x^2-sin(x)
```

- Lo que está sucediendo aquí es que el comando **`inline`** requiere una cadena como entrada y **`char`** convierte *f*, expresión simbólica, en la cadena **`'x^2-sin(x)'`**. (Si simplemente hubieramos escrito **`fin=inline(f)`** obtendríamos un mensaje de error puesto que *f* no es una cadena). La función *inline* **`fin`** acepta ahora argumentos:

```
fin(4)
ans =
   16.7568
```

- De forma similar podemos construir una función a partir de la derivada de *f:*

```
fxin=inline(char(diff(f)))
fxin =
   Inline function:
   fxin(x) = 2*x-cos(x)
```

- La función MATLAB **`char`** reemplaza el argumento que recibe por la cadena que lo representa, haciéndolo así accesible a funciones que requieren cadenas como argumento, como por ejemplo **`inline`**.

5

Cálculo con MATLAB. Funciones y cálculo diferencial (iv)

- Sin embargo, cuando **`char`** es aplicado a una expresión simbólica, el resultado aún es una expresión simbólica y puede ser derivada:

```
diff(char(f))
ans =
   2*x-cos(x)
```

- La otra forma de crear una función evaluable es escribiendo una función en un fichero .m
- Esta es la forma principal de definir funciones en la mayor parte de aplicaciones de MATLAB. Recordemos que un fichero .m se crea de forma separada.
- Supongamos que hemos escrito el fichero .m **`fun1.m`** que contiene lo siguiente:

```
function out=fun1(x)
out=x^2-sin(x);
```

- Ahora la función **`fun1`** puede ser invocada con un argumento:

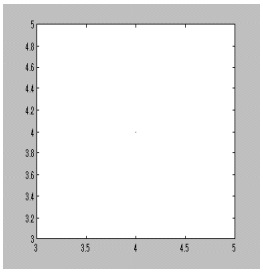
```
fun1(4)
ans =
   16.7568
```

6

Cálculo con MATLAB. Visualización (i)

- Una de las cosas que podemos querer hacer con una función es representar su gráfica. La operación más elemental en MATLAB es dibujar un punto con unas coordenadas específicas:

```
plot(4,4)
```



- La salida de este comando es el punto azul centrado en la figura. Para dibujar una curva MATLAB dibuja una secuencia de puntos conectados mediante segmentos de recta.
- La entrada para tales dibujos consiste en dos vectores (listas de números). El primer argumento es el vector de coordenadas x y el segundo el vector de coordenadas y.
- MATLAB conecta los puntos cuyas coordenadas aparecen en posiciones consecutivas de los vectores de entrada.
- Dibujemos la función que definimos en la transparencia anterior. En primer lugar se necesita un vector de coordenadas x:

```
X1=-2:.5:2
X1 =
    Columns 1 through 7
    -2.0000   -1.5000   -1.0000   -0.5000   0    0.5000    1.0000
    Columns 8 through 9
    1.5000    2.0000
```

- **X1** es un vector de nueve componentes, comenzando en -2 y avanzando hasta 2 con incrementos de 0,5. Ahora se debe preparar un vector de coordenadas y aplicando nuestra función sobre las coordenadas x. Para ello debemos modificar nuestra función de tal forma que pueda operar sobre las componentes individuales de un vector.

7

Cálculo con MATLAB. Visualización (ii)

- La función MATLAB **vectorize** reemplaza los operadores *****, **^** y **/** por **.***, **.^** y **./** respectivamente.
- Recordemos que MATLAB trabaja fundamentalmente sobre vectores y matrices, y su interpretación por defecto de la multiplicación, división y exponenciazión es que son operaciones sobre matrices.
- El punto antes del operador indica que se debe aplicar componente a componente.

```
fin=inline(vectorize(f))
fin =
   Inline function:
   fin(x) = x.^2-sin(x)
Y1=fin(X1)
   Y1 =
   Columns 1 through 7
   4.9093    3.2475    1.8415    0.7294         0   -0.2294
   0.1585
   Columns 8 through 9
   1.2525    3.0907
plot(X1,Y1)
```

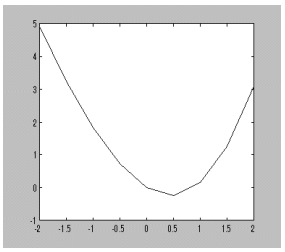
8

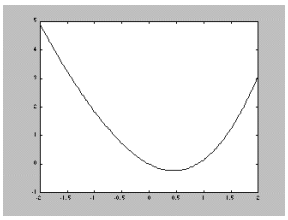
Cálculo con MATLAB. Visualización (iii)

- Esta representación es bastante “mala”; de hecho se pueden ver las “esquinas”. Para solucionar esto es posible reducir el paso empleado. Insertaremos puntos y coma después de las definiciones de **`X1`** e **`Y1`** para suprimir la salida.

```
X1=-2:.02:2;
Y1=fin(X1);
plot(X1,Y1)
```

- En realidad, la visualización de una función simbólica puede lograrse de forma más sencilla con el comando **`ezplot`**. Sin embargo, **`plot`** permite modificar el color, la apariencia de las curvas, etc.





Cálculo con MATLAB. Resolución de ecuaciones (i)

- La gráfica de *f* indica que existen dos soluciones para la ecuación *f(x)=0*, una de las cuales es claramente 0. Disponemos en MATLAB de **`solve`**, un “solucionador” simbólico de ecuaciones, y **`fzero`**, un “solucionador” numérico.
- Ilustraremos **`solve`** con un ejemplo sencillo.

```
g=x^2-7*x+2
g =
x^2-7*x+2
groots=solve(g)
groots =
[ 7/2+1/2*41^(1/2)]
[ 7/2-1/2*41^(1/2)]
```

- Aquí, **`solve`** encuentra todas las raíces que puede y las muestra como las componentes de un vector columna. Ordinariamente, **`solve`** tratará de despejar la x, si está presente, o por la variable alfabéticamente más cercana a x. Esto puede obviarse especificando la variable a despejar. Nótese que el primer argumento de **`solve`** es una expresión simbólica que solve iguala a 0.

```
syms y
solve(x^2+y^2-4,y)
ans =
[  (-x^2+4)^(1/2)]
[ -(-x^2+4)^(1/2)]
```

Cálculo con MATLAB. Resolución de ecuaciones (ii)

- Probemos ahora **solve** sobre nuestra función *f*:

```
froots=solve(f)
froots =
0
```

- Aquí, **solve** encuentra la raíz en 0 pero no la otra. Podemos probar con **fzero** que resuelve la ecuación numéricamente comenzando en valor inicial para la variable...
- El comando **fzero** no acepta *f* como argumento sino que requiere **char(f)** o **fin** y podrá encontrar la segunda raíz. También aceptaría el nombre del fichero .m **'fun1'** (con las comillas) o la expresión **@fun1** (la arroba, **@**, es un marcador para un nombre de función):

```
newfroot=fzero(char(f),.8)
Zero found in the interval: [0.70949, 0.89051].
newfroot =
   0.8767
newfroot=fzero(fin,.8)
   Zero found in the interval: [0.70949, 0.89051].
newfroot =
   0.8767
newfroot=fzero('fun1',.8)
Zero found in the interval: [0.70949, 0.89051].
newfroot =
   0.8767
```

11

Cálculo con MATLAB. Cálculo integral e integración numérica (i)

- Aún no hemos hablado de la integración. MATLAB dispone de un "integrador" simbólico, denominado **int**, que puede integrar fácilmente f:

```
intsf=int(f,0,2)
intsf =
5/3+cos(2)
```

- Sin embargo, si reemplazamos *f* por la función *h* definida de la siguiente forma:

```
h=sqrt(x^2-sin(x^4))
h =
(x^2-sin(x^4))^(1/2)
```

- **int** será incapaz de evaluar la integral:

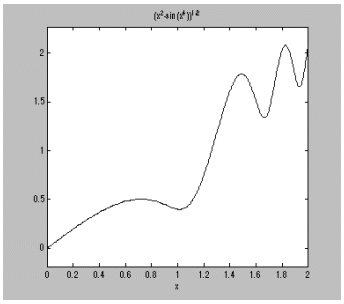
```
int(h,0,2)
Warning: Explicit integral could not be found.
> In C:\MATLABR11\toolbox\symbolic\@sym\int.m at line 58
ans =
int((x^2-sin(x^4))^(1/2),x = 0 .. 2)
```

12

Cálculo con MATLAB. Cálculo integral e integración numérica (ii)

- Sin embargo, si escribimos **`double`** (indicando números de doble precisión) antes de la expresión integral, MATLAB retornará el resultado de una integración numérica.

```
double(int(h,0,2))
Warning: Explicit integral could not be
   found.
> In
   C:\MATLABR11\toolbox\symbolic\@sym\int.m
   at line 58
ans =
    1.7196
```



- Podemos comprobar la plausibilidad de esta respuesta dibujando h entre 0 y 2 y estimando el área bajo la curva:

```
ezplot(h,[0,2])
```

13

Cálculo con MATLAB. Cálculo integral e integración numérica (iii)

- El valor numérico retornado por MATLAB es algo menos que la mitad del área de un cuadrado de 2 unidades de lado lo cual es consistente con nuestro gráfico.
- La integración numérica invocada por la combinación de **`double`** e **`int`** no es nativa, esto es no es propia de MATLAB sino de MAPLE del cual han sido tomadas las rutinas de cálculo simbólico de MATLAB.
- MATLAB también dispone de un integrador numérico denominado **`quadl`**. Las rutinas **`double(int(...))`** y **`quadl(...)`** proporcionan respuestas ligeramente distintas.

```
quadl(inline(vectorize(h)),0,2)
ans =
    1.7196
```

14

Cálculo con MATLAB. Resumen

- Hasta ahora MATLAB ha sido usado como una herramienta de cálculo numérico. Sin embargo, también es posible para realizar cálculo simbólico.
- MATLAB permite, básicamente:
  - cálculo simbólico
  - cálculo numérico
  - representaciones gráficas
- Es necesario distinguir las representaciones simbólicas de las funciones de las funciones como "reglas de evaluación"; las primeras pueden ser derivadas e integradas, las segundas no; las primeras no pueden ser evaluadas, las segundas sí.
- El comando **`diff`** permite derivar una expresión simbólica.
- Para visualizar una función podemos utilizar **`plot`** (emplea vectores) o **`ezplot`** (representa expresiones simbólicas).
- Los comandos **`solve`** y **`fzero`** permiten resolver ecuaciones.
- El comando **`int`** permite realizar integrales simbólicas mientras que **`double(int(...))`** y **`quadl(...)`** permiten calcular integrales numéricas.

15