Prácticas Matlab

Práctica 3 (19/10/2012)

Objetivos

- Repasar, mediante ejemplos, la definición de polinomio de Taylor.
- Ayudar a comprender la aproximación local que proporcionan los polinomios de Taylor observando la incidencia que tiene en la aproximación el grado del polinomio de Taylor y la cercanía al punto en el que se hace el desarrollo.

Comandos de Matlab

1.- Para representar funciones

plot(x,y)

dibuja una línea que une los puntos de abscisas el vector "x" y ordenadas"y".

plot(x,y,s)

Realiza el gráfico con el estilo indicado en "s". Para ello "s" debe ser una cadena de caracteres formada por uno o ningún elemento de las tres columnas siguientes:

y yellow	. point	- solid
m magenta	o circle	: dotted
c cyan	x x-mark	-. dashdot
r red	+ plus-- dashed	
g green	* star	
b blue	s square	
w white	d diamond	
k black	v triangle (down)	
	^ triangle (up)	
	< triangle (left)	
	> triangle (right)	
	p pentagram	
	h hexagram	

figure(n)

Para crear una ventana de dibujo

Ejemplo:

```
>>  x=-pi : 0.1: pi;
>>  figure(1);
>>  plot(x,sin(x),'b. ');
>>  figure(2);
>>  plot(x,cos(x), 'gd-');
```

hold on hold off

Permite dibujar dos gráficas en una misma ventana de dibujo.

Ejemplo:

```
>>  x=-pi : 0.1: pi;
>>  hold on
>>  figure(1);
>>  plot(x,sin(x),'b. ');
>>  plot(x,cos(x), 'gd-');
>>  hold off
>>  %para ver más opcioes teclea la orden:
>>  help plot
```

2.- Para construir objetos simbólicos:

```
syms arg1 arg2 ...
```

Es la forma abreviada de escribir:

```
arg1 = sym('arg1');
arg2 = sym('arg2'); ...
```

Si se quiere indicar el tipo del objeto simbólico se puede escribir:

```
syms arg1 arg2 ... real
```

Es la forma abreviada de escribir:

```
arg1 = sym('arg1','real');
arg2 = sym('arg2','real'); ...
```

```
syms arg1 arg2 ... positive
```

Es la forma abreviada de escribir:

```
arg1 = sym('arg1','positive');
arg2  =  sym('arg2','positive');
...
```

syms arg1 arg2 ... unreal

Es la forma abreviada de escribir:

```
arg1 = sym('arg1','unreal');
arg2 = sym('arg2','unreal'); ...
```

Ejemplo:

```
>>    syms x
>>    y=sin(x)+3^x+8/(x+1)
```

3.- Para dibujar la gráfica de una función simbólica

```
ezplot(f, [a,b], fig)
```

Ejemplo:

```
>>    syms x
>>    y=sin(x)+3^x+8/(x+1)
>>    %El  segundo  y  el  tercer  parámetro  son
      opcionales.
```

```
>>    ezplot(y, [-2,2])
```

Ejemplo:

```
>>   %Representar una función implícita
>>   % xy^2 - x^2 - 2y = 0 ,  x ∈ [-6,6]
>>   ezplot('x*y^2-x^2-2*y',[-6,6])
```

4.- Para obtener el límite de una expresión simbólica "f" cuando la variable "n" tiende al valor "a"

```
limit(f,n,a)
```

Ejemplo:

```
>> syms n
>> limit(1/n,n,inf)
```

5.- Para obtener la derivada de orden n una función simbólica respecto de la variable x.

```
diff(f,x,n)
```

Ejemplo:

```
>> syms x y
>> f=sin(x*y)/x; diff(f,x,3)
```

6.- Para simplificar las expresiones simbólicas:

`collect(p)`	Reúne los términos iguales
`horner(p)`	Cambia a la representación anidada o de Horner
`expand(p)`	Expande los productos en sumas
`factor(p)`	Factoriza la expresión (a veces) si el argumento es una función simbólica. Si se trata de un número proporciona la factorización en números primos.
`simplify(p)`	Simplifica una expresión mediante la aplicación de diversas identidades algebraicas.
`simple(p)`	Utiliza diferentes herramientas de simplificación y selecciona la forma que tiene el menor número de caracteres.
`pretty(p)`	Visualiza la expresión de una manera similar a la utilizada en la escritura habitual.

7. Para hacer una sustitución simbólica simple de "var" en "valor" en la expresión "f":

```
subs(f,var,valor)
```

Ejemplo:

```
>> syms x
>> y=sin(x)+3^x+8/(x+1)
>> subs(y, x, 2)
```

8.- Para calcular el polinomio de Taylor de grado n de la función f en el punto a

```
taylor(f,n+1,a)
```

Ejemplo

```
>>syms x
>>f=x*sin(x+1);
>>taylor(f,5,0)
%Devuelve el polinomio de Taylor de f en el punto 0
%de grado 4.
```

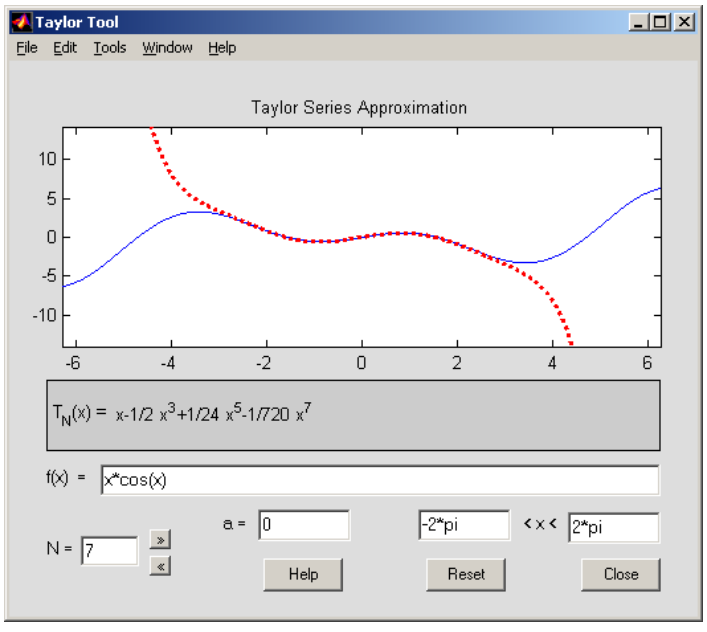
9.- Herramienta taylortool:

Matlab posee una herramienta que permite obtener el polinomio de Taylor de una función y la representación gráfica del polinomio junto con la de la función.

Ejecuta en la ventana de comandos la orden:

>> taylortool

Se abrirá una ventana (ver figura) en la que puedes introducir la función, el grado del polinomio y el intervalo en el que quieres representar la función y el correspondiente polinomio.



En el ejemplo de la figura se trata del polinomio de Taylor centrado en el punto $a = 0$ de grado 7 para la función $f\left(x\right) = x\cos x$ en el intervalo $\left[-2\pi, 2\pi\right]$.

Con los botones en forma de flecha puedes incrementar y/o disminuir el grado del polinomio.

Observa que a medida que el grado del polinomio aumenta el polinomio de Taylor aproxima mejor a la función y en un intervalo más grande.

Ejercicios

1

a) Obtener el polinomio de Taylor de grado 3 de la función $y=\sqrt{1-\frac{x}{2}}$ alrededor del punto a= 0.

b) Obtener, mediante el polinomio anterior, un valor aproximado de $\sqrt{0,5}$.

c) Hallar una cota del error cometido en dicha aproximación.

d) Comprobar con Matlab cómo la aproximación de $\sqrt{0,5}$ es mejor cuánto mayor sea el grado del polinomio de Taylor que se utilice. Escribir una tabla con la aproximación que dan los polinomios de Taylor de grado 1, 3, 5, 20.

Indicaciones

Este ejercicio es el ejercicio propuesto nº11 del tema 2.

a) Calcula el polinomio de Taylor de grado 3 con el comando `Taylor` y guardarlo en una variable de nombre, por ejemplo, pol3.

b) Comprueba que el valor de x para el que se cumple $\sqrt{0,5}=\sqrt{1-\frac{x}{2}}$ es $x=1$. Sustituye después en la expresión de `pol3` ese valor con el comando `subs`.

c) Para acotar el error utiliza la fórmula de Lagrange.

EXPRESIONES DEL RESTO: Sea f es una función derivable $n+1$ veces en un intervalo abierto I, que contenga al punto a. Si $R_n\left[f(x);a\right]$ es el resto enésimo de Taylor correspondiente a la función f en el punto $x=a$ entonces:

$$R_n\left[f(x);a\right]=\frac{f^{(n+1}(t)}{(n+1)!}(x-a)^{n+1}$$

siendo t un punto intermedio entre a y x.

En nuestro caso, la función que estamos considerando es $f(x)=\sqrt{1-\frac{x}{2}}$. Queremos aproximar para $x=1$ el valor de la función por el del polinomio de Taylor de grado $n=3$ desarrollado en el punto $a=0$, esto es,

$$f(1)\approx T_3(1)$$

Considerando el resto de Lagrange la diferencia entre estos dos valores es

$$f(1)-T_3(1)=\frac{f^{(4}(t)}{4!}(1-0)^4 \quad (*)$$

siendo t un punto intermedio entre 0 y 1.

Realiza entonces los siguientes pasos con Matlab para obtener una cota de la aproximación $f(1)\approx T_3(1)$

1. Calcula la derivada cuarta de la función (comando `diff`) y represéntala (comando `plot`).
2. A partir del dibujo, encuentra un valor M que sea cota superior del valor absoluto de esta derivada en el intervalo [0, 1].
3. Escribe finalmente la cota del error utilizando la expresión del resto de Lagrange vista en (*) teniendo en cuenta que:

$$\left|f(1)-T_3(1)\right|=\left|\frac{f^{(4}(t)}{4!}(1-0)^4\right|\le\frac{M}{4!}$$

d) Rellena la siguiente tabla

	Polinomio de Taylor $T_n(x)$	Aproximación $T_n(1)$	$f(1)-T_n(1)$
n=1			
n=3			
n=5			
n=20			

2

Se considera $f(x)=\log\left(\frac{x+2}{2x-2}\right)$. Se pide:

a) Representar el dominio de la función $f(x)$.

b) Calcular el polinomio de Taylor de grado n de $f(x)$ en $a=4$ y determina la expresión del resto enésimo de Lagrange de $f(x)$ en $a=4$.

c) Calcular una cota del error cuando queremos aproximar $\log\left(\frac{6'1}{6'2}\right)$ por el polinomio de grado 3.

d) ¿Cuál es el grado del polinomio de Taylor de $f(x)$ en $a=4$ que habría que considerar para aproximar $\log\left(\frac{6'1}{6'2}\right)$ con un error menor que 10^{-1}?

e) Calcular con Matlab el grado del polinomio de Taylor que aseguraría la aproximación de $\log\left(\frac{6'1}{6'2}\right)$ con un error menor que 10^{-6}

Indicaciones

Este ejercicio es el ejercicio propuesto nº12 del tema 2.

a) Calcula a mano los valores de x para los cuales $\frac{x+2}{2x-2}>0$. Recuerda que para que un cociente sea positivo el numerador y el denominador deben tener el mismo signo.

b) Calcula las primeras derivadas a mano teniendo en cuenta que

$$f(x)=\log\left(\frac{x+2}{2x-2}\right)=\log(x+2)+\log(2x-2) \quad si\ x>1$$

Puedes comprobar con Matlab que no te has confundido utilizando el comando `diff`.

Intenta obtener después una expresión para la derivada enésima.

Escribe a continuación la expresión del resto de Lagrange pedido.

c) Determina en primer lugar el valor de x que cumple que $f(x)=\log\left(\frac{6'1}{6'2}\right)$.

$x=$

Obtén después la expresión del resto de orden 1 (R_1) y acótalo

$R_1=$

$|R_1|\leq$

Si la cota obtenida es menor que 10^{-1}, el grado del polinomio a considerar sería 1, en caso contrario, debes repetir el proceso considerando el polinomio de grado 2. Este proceso deberás repetirlo hasta encontrar un n de forma que el resto enésimo cumpla que es menor que 10^{-1}.

El valor de n es:

d) Repite los pasos seguidos en el apartado c) considerando ahora 10^{-6} en lugar de 10^{-1}.

El valor de n es:

***Observaciones**:*

- La aproximación del polinomio de Taylor es local. En puntos alejados del punto en el que se hace el desarrollo del polinomio el valor de éste y la función pueden no ser próximos.
- Considerando un punto x suficientemente próximo al punto el que se hace el desarrollo, a, la aproximación de $f(x)$ por su polinomio de Taylor es mejor cuanto más grande sea el grado del polinomio, n.
- Considerando un valor n cualquiera, la aproximación de $f(x)$ por la del polinomio de Taylor con dicho grado es mejor cuánto más cerca esté x del punto en el que se hace el desarrollo, a.

Resumen de comandos

Estos son los comandos utilizados en esta práctica que se darán por conocidos en las prácticas siguientes y que conviene retener porque se podrán preguntar en las distintas pruebas de evaluación.

- Para operar con variables simbólicas: `syms, diff, subs`
- Para representar funciones por puntos: `plot`
- Para calcular el polinomio de Taylor: `taylor`