Prácticas de Matemáticas I – Sesión IV

Ingeniería Técnica Industrial con Intensificación en Diseño Industrial

UPNA - Campus de Tudela

Víctor Domínguez Báguena

Noviembre 2009

Sesión 4

Derivabilidad

En esta sesión introduciremos la derivación numérica en Matlab y las herramientas asociadas: cálculo de extremos y el polinomio de Taylor.

1.1. Derivación simbólica

La derivación de una función en Matlab se calcula vía el comando `diff`. Se trata de una derivación *analítica* y por tanto la variable respecto a la que se deriva debe haber sido declarada previamente como simbólica. La misma instrucción, añadiendo un argumento adicional, puede utilizarse para calcular derivadas de mayor orden. `diff`

Un simple ejemplo ilustra las características de este comando:

```
>> f=vectorize(inline('cos(x)*exp(3*x)','x'));
>> syms t  % t es variable simbolica
>> diff(f(t),t)
ans =
-sin(t)*exp(3*t)+3*cos(t)*exp(3*t)
>> diff(f(t),t,4) % derivada cuarta
ans =
 28*cos(t)*exp(3*t)-96*sin(t)*exp(3*t)
```

Más engorroso resulta definir la función derivada. Para ello hay que proceder como sigue

nombre función derivada= `vectorize(inline(char(diff(`*función por derivar*`)),`*variable*`))`

(Toma buena nota de la colocación de los paréntesis). Por ejemplo

```
>> f=vectorize(inline('cos(x)*exp(3*x)','x'));
>> syms t  % t es variable simbolica
>> f1=vectorize(inline(char(diff(f(t),t)),'t'));
>> f4=vectorize(inline(char(diff(f(t),t,4)),'t'));
```

Evidentemente se pueden utilizar los comandos para simplificar y manipular las expresiones: `simple, factor, expand`,,...

Nota

La declaración de una función derivada es tan engorrosa debido a que la función `inline` requiere como primer argumento una variable carácter (texto) o bien una constante carácter, que viene definido necesariamente entre comillas.

Las instrucciones anteriores proceden de la siguiente forma: primero se deriva la función (requiere una variable simbólica) que da como resultado un símbolo. Luego, se transforma a texto (`char`), y finalmente se inserta dentro de `inline` para declarar la función en la forma estándar.

`char`

Evidentemente, esto se puede hacer en dos pasos:

```
>> f=vectorize(inline('cos(x)*exp(3*x)','x'));
>> f1= diff(f(t),t)  % f1 es un simbolo
f1 =
3*exp(3*t)*cos(t) - exp(3*t)*sin(t)
>> f1=vectorize(inline(char(f1),'t')) %f1 es una función
f1 =
     Inline function:
     f1(t) = 3.*exp(3.*t).*cos(t) - exp(3.*t).*sin(t)
>> f4= char(diff(f(t),t,4))  %f4 es un variable carácter
     f4 =
     28*exp(3*t)*cos(t) - 96*exp(3*t)*sin(t)
>> f4=vectorize(inline(f4,'t'))
f4 =
     Inline function:
     f4(t) = 28.*exp(3.*t).*cos(t) - 96.*exp(3.*t).*sin(t)
```

1.2. Cálculo de máximos y mínimos

A continuación vamos a mostrar como resolver el problema de cálculo de máximos y mínimos para una función derivable recurriendo a los comandos de Matlab. El cálculo de los posibles extremos, esto es, la resolución de la ecuación $f'(x) = 0$, se hace vía el comando `solve`.

El problema que resolveremos es:

$$\text{Encontrar los extresmos relativos de } f(x) := \frac{x^2 - 3x + 10}{x + 1}.$$

Comenzamos introduciendo la función en Matlab

```
>> f=vectorize(inline('(x^2-3*x+10)/(x+1)','x'));
>> syms x  % x variable simbolica
>> pretty(f(x))

   2
  x  - 3 x + 10
  -------------
      x + 1
```

Hemos utilizado `pretty` para visualizar la función de una forma más desplegada y comprobar si la función está bien introducida, especialmente la colocación de los paréntesis. Calculamos las dos primeras derivadas

```
>> f1=vectorize(inline(char(diff(f(x),'x'))));
>> f2=vectorize(inline(char(diff(f(x),'x',2))));
```

Seguidamente procedemos a buscar los ceros de la primera derivada: `solve`

```
>> sol=solve(f1(x),'x')que presenta en cada uno de ellos.

sol =
 -1+14^(1/2)
 -1-14^(1/2)
```

(`sol` es una variable simbólica). En la variable `sol` se almacenan las raíces de la primera derivada, en este caso hay dos, $\{-1+\sqrt{14}, -1-\sqrt{14}\}$. Comprobamos el signo de la segunda derivada para cada uno de las raíces:

```
>> f2(sol(1))  % f2 evaluada en la primera raiz
ans=
 -1/7*14^(1/2)+5/7+1/196*(2*(-1+14^(1/2))^2+26-6*14^(1/2))*14^(1/2)
```

Al evaluar un símbolo, el resultado es una expresión, exacta, pero de la que es difícil saber qué signo tiene. Vamos a pasar las soluciones a números reales, esto es, a evaluarlas

```
>> sol1=double(sol(1)),...
   sol2=double(sol(2)) % las dos soluciones evaluadas
sol1 =
  2.7417
sol2 =
 -4.7417
>> f2(sol1)
ans =
    0.5345
>> f2(sol2)
ans =
  -0.5345
```

Por tanto `sol(1)` (esto es, $x = -1 + \sqrt{14}$) es mínimo mientras que `sol(2)` corresponde a un máximo.

Finalmente podemos visualizar la función para comprobar que efectivamente ésos son los máximos y mínimos de `f`:

```
>> t=linspace(-9,9,5000);
>> plot(t,f(t))
>> ylim([-40,40]) % limitamos el rango en la "y"
>> hold on
>> plot([sol1 sol2],f([sol1 sol2]),'o',...
         'markersize',10,'markerfacecolor','r')
```

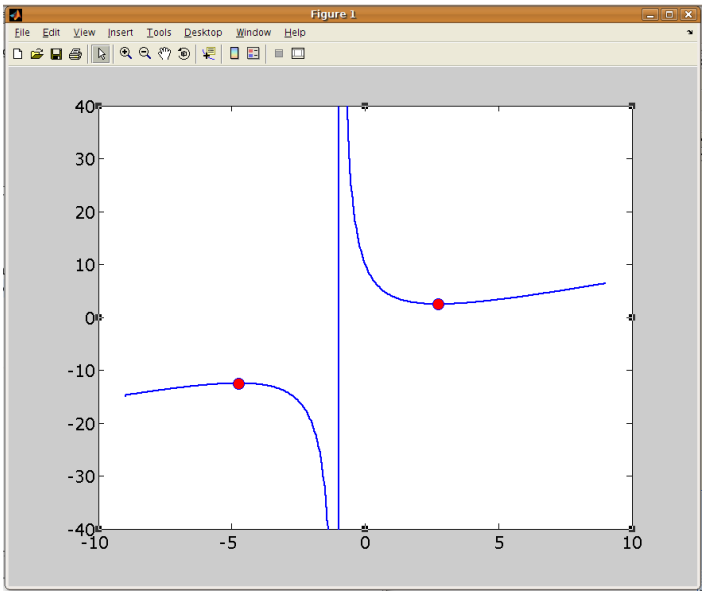

Figura 1.1: Solución del problema de máximos y mínimos.

En la penúltima instrucción, activamos la superposición (`hold on`) de forma que nuevas gráficas se añaden sobre las existentes en lugar de borrarlas. Seguidamente, con el último comando, marcamos los extremos calculados con un círculo (`'o'`), de tamaño (`markersize`) 10 y con color (`markerfacecolor`) rojo. El resultado se muestra en la Figura 1.1.

Nota

El comando `solve` resuelve la ecuación (o sistemas de ecuaciones) de forma simbólica. Por ello las instrucciones anteriores devolvían cantidades como `-1+14^(1/2)` en lugar de la conveccional `2.7417`. La solución de la ecuación se busca mediante procesos analíticos por lo que es muy posible que para ecuaciones algo complicadas no pueda encontrar la solución dada las limitaciones del cálculo analítico. De forma alternativa existe la instrucción `fzero` que busca soluciones numéricas de una ecuación.

fzero

La instrucción `double` utilizado anteriormente transforma un símbolo en un número real, y es necesario para dibujar los puntos extremos

1.3. Polinomio de Taylor

El polinomio de Taylor se calcula en Matlab vía el comando `taylor`. Veámoslo nuevamente con un simple ejemplo

taylor

```
>>  f=vectorize(inline('cos(x)*x','x'))
>>  g=taylor(f(x),x,4,pi)
g =
 -x+1/2*pi*(x-pi)^2+1/2*(x-pi)^3
```

construye el polinomio de Taylor de $f(x)$ en $x_0 = \pi$ **de grado 3**. En general, el formato del comando es

`taylor`(*función*, *variable*, *grado+1*, *punto del desarrollo*)

El resultado es una variable simbólica. Para definir una función que contenga este polinomio de Taylor se procede de la forma habitual

```
>> p3=vectorize(inline(char(g),'x')); % g contiene el polinomio
>> p3(3)-f(3) % error funcion - polinomio en x=4
ans =
   5.0209e-005
```

Lo mismo con un polinomio de grado 6:

```
>> g=taylor(f(x),x,7,pi); % grado 6 en pi
>> p6=vectorize(inline(char(g),'x'));
>> p6(3)-f(3)
ans =
   -3.3564e-008
```

Finalmente, podemos mostrar función y polinomio recurriendo a las instrucciones de dibujo. Así,

```
>> t=linspace(0,6,1000);
>> plot(t,f(t),'k-',t,p3(t),'b--',t,p6(t),'r:')
>> legend('funcion','pol de Taylor de grado 3',...
   'pol de Taylor de grado 6')
```

crea la Figura 1.2

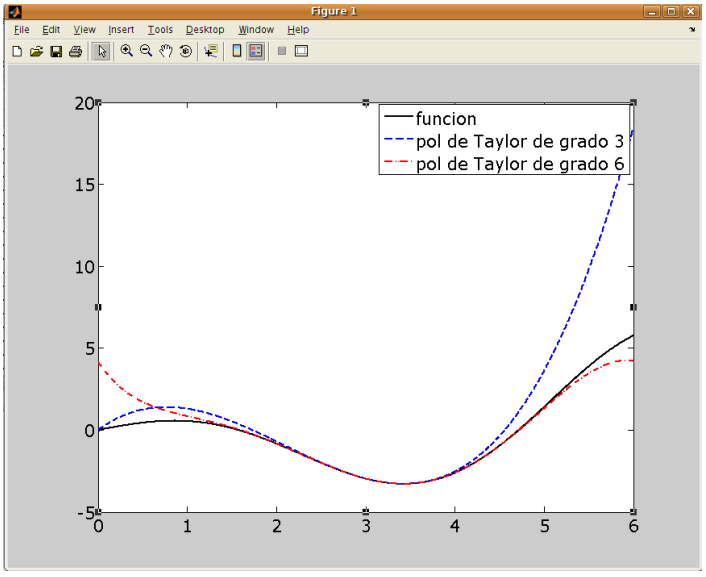

Figura 1.2: Función y dos polinomios de Taylor de grado 3 y 6 en π

Nota También es válido la instrucción

```
>> p6=vectorize(inline(char(taylor(f(x),x,7,pi)),'x'));
```

Obviamente un polinomio se puede desarrollar. Para ello basta utilizar el comando `expand`.

Ejercicios

1.1 Deriva la siguientes funciones

(a) $\dfrac{x+4}{x-3}$

(b) $\dfrac{2^x}{x+1}$

(c) $\cos(e^x \operatorname{sen}(x))$

(d) x^x

(e) $\log\cos(x)$

(f) $\dfrac{x^2+1}{x^2-1}$

(g) x^x

(h) $\cos^{\operatorname{sen}(x)}(x), \quad x \in [0, \pi/2]$

(i) $e^{-\cos(x^2)} + \cos(x^2)e^{-\cos(x^2)}$

(j) $\arctan(\tan(x^2))$

(k) $e^x e^{\cos(x)} 2^{1+x^2}$

(ℓ) $\operatorname{sen}^2(x) - \cos^2(x)$

(m) $|x|$

(n) $\arctan(1/x)x$

(ñ) $\tan(2^x)$

(o) x^{x^x}

(p) $x - |x|$

(q) $\dfrac{\cos(x^2\log(x))}{\operatorname{sen}(x^2\log(x))} - x^3$.

1.2 Halla las derivadas de diferentes órdenes para

(a) $\dfrac{e^x + e^{-x}}{2}$

(b) $\dfrac{e^x - e^{-x}}{2}$

(c) $e^x\cos(x)$

(d) $(x-a)^n$

(e) $\dfrac{1}{1-x}$

(e) $\log(x^2)$

(f) $\operatorname{sen}^2(x) - \cos^2(x)$

(g) $x\cos(x)$

(h) xe^{x^2}

(h) $(1-x)^a$.

(**Ayuda** cuando aparezca un parámetro, hay que definir una función dos variables, por ejemplo 'a', 'x' y derivar sólo respecto a la variable 'x'.)

1.3 Encontrar los extremos relativos de

(a) $x^4 - 2x^3 + 2x + 1$,

(b) $x^3 - 12x - 3, \quad x \in [0, 3]$

(c) $\cos(x) + \operatorname{sen}(x)$,

(d) $\cos^2(x) + 1/2 \operatorname{sen}(2x), \quad x \in [0, \pi]$

(**Ayuda** Utiliza `simple` si las raíces son muy complicadas. Es posible que tengan una expresión más sencilla de lo que parece a simple vista. De todas formas siempre puedes evaluar la expresión utilizando `double` o `vpa` (*variable precision arithmetic* donde se puede evaluar con mayor precisión).

vpa

1.4 Calcula el polinomio de Taylor de

(a) $\log(1 + x)$ en $x_0 = 0$ y grados 4, 8 y 16

(b) $\dfrac{x^2 + x - 1}{x^2 + 2}$ en $x_0 = 1$ y grados 4,6 y 10

(c) $\arcsin(x)$ en $x_0 = 0$ y grados 4,6 y 10

(d) $\tan(\cos(x))$ en $x_0 = 0$ y grados 4, 8.

(e) $\log(1 + \operatorname{sen}^2(x))$ en $x_0 = 0$ y grados 2, 4 y 12

(f) $\sqrt{1 + \cos(x^2)}$ en $x_0 = \pi/2$ y grados 2, 4 y 12

Dibuja función y polinomio cerca del punto del desarrollo y comprueba en qué zonas hay convergencia de la función al polinomio.

1.5 Construir un polinomio tal que permita evaluar la función $\exp(x)$ con un error menor que 10^{-8} en el intervalo $[-1, 1]$. £Se puede utilizar para evaluar la función, digamos, en $x = \pm 4$?. £Cómo solucionarías este problema, es decir, evaluar la exponencial con este polinomio?.

1.6 En este ejercicio estudiamos las fórmulas de derivación numérica. Dada f derivable, se propone como aproximación de la derivada $f'(x_0)$ las siguientes fórmulas

- $\Delta_h(f, x_0) := \dfrac{f(x_0 + h) - f(x_0)}{h}$ **(diferencia progresiva)**.
- $\nabla_h(f, x_0) := \dfrac{f(x_0) - f(x_0 - h)}{h}$ **(diferencia regresiva)**.
- $\delta_h(f, x_0) := \dfrac{f(x_0 + h) - f(x_0 - h)}{2h}$ **(diferencia centrada)**

En todos los casos, $h > 0$ es una cantidad suficientemente pequeña. En vista de lo anterior, ¿cuál te parece mejor?. ¿Cuál es más costosa de evaluar?

Haz un pequeño programa en Matlab que te permita comprobar el comportamiento de estas fórmulas de cuadratura, diferencias progresivas, regresivas y centrales, para la aproximación de la derivada de una función que conozcas exactamente y diferentes valores de h. Observa el error que comete cada una de ellas en función de h y plantéate de nuevo las cuestiones anteriores.