Tema 4

Cálculo con funciones de una variable

Para el estudio de este tema familiarícese con los comandos empleados en MAXIMA para todas las operaciones habituales en cálculo con funciones de una variable, en particular las siguientes: límites, derivadas, integrales (definidas e indefinidas) y desarrollos en serie de Taylor:

- Para calcular límites:
  - Límites bien definidos: `limit(expresión, variable, valor)`
  - Límites laterales: `limit(expresión, variable, valor, dirección)`
- Para calcular derivadas:
  - Primera derivada: `diff(expresión, variable)`
  - Derivadas de orden superior: `diff(expresión, variable, orden)`
  - Derivadas parciales de orden arbitrario `diff(expresión, variable1, orden1, variable2, orden2, ..., variable`n`, orden`n`)`
- Para calcular integrales:
  - Integral indefinida (primitiva): `integrate(expresión, variable)`
  - Integral definida: `integrate(expresión, variable, límite inferior, límite superior)`
  - Cálculo numérico de una integral definida: `quad_qag`
- Para calcular desarrollos en serie de Taylor:
  - `taylor(expresión, variable, punto en torno al que desarrollamos, orden)`

4.1. Problemas resueltos

1. Realice un programa que proporcione los desarrollos en serie de Taylor sucesivos de una función desde orden cero hasta un orden n arbitrario.
   - input: La función a desarrollar, el punto en torno al que desarrollamos y el orden máximo n.
   - output: La lista de desarrollos en serie de Taylor correspondiente.

Soluciones

Ejercicio 1

Realice un programa que proporcione los desarrollos en serie de Taylor sucesivos de una función desde orden cero hasta un orden n arbitrario

En la mayoría de los problemas de esta parte del curso no vamos a insistir demasiado en la cuestión de la *optimización*, pero en este problema haremos una excepción. La forma más inmediata de hacer este ejercicio sería hacer una función que llamase a la función `taylor(`f`,` x`,` x_0`,` n`)` sucesivamente para valores del orden del desarrollo n entre $n = 0$ y el orden máximo que se desee calcular. A pesar de ser totalmente ineficiente, ya que estamos repitiendo el mismo cálculo muchas veces, eso daría la respuesta correcta. Para hacerlo de esa forma sencillamente haríamos:

```
taylorlistineficiente(f, x, x0, orden) := makelist( taylor(f, x, x0, i),
   i, 0, orden, 1);
```

Esta forma de hacerlo es poco recomendable si queremos evaluar estos desarrollos hasta un orden muy alto, o si la función sobre la que vamos a operar es difícil de evaluar. Por ejemplo, supongamos que la función $f(x)$ cuyo desarrollo queremos conocer no es una función conocida en términos de una expresión matemática cerrada, sino que sólo podemos conocer los valores que esta función, o sus derivadas, toman para valores numéricos concretos de la variable, y supongamos que cada una de esas evaluaciones numéricas es muy costosa computacionalmente (p. ej. porque involucra resolver numéricamente un conjunto de ecuaciones trascendentes). En ese caso haríamos el programa intentando minimizar el número de evaluaciones de la función $f(x)$ o sus derivadas.

Es obvio que para calcular el listado de desarrollos pedidos de una forma eficiente basta con calcular el desarrollo de orden más alto una sola vez. Una posible forma de hacer eso es como se muestra en la función `taylor_list`, cuyo listado incluimos a continuación. Listado del archivo `taylor_list.mc`:

```
/*  FUNCION "taylorlist"
    Calcula la lista de desarrollos en serie de Taylor desde orden 0 hasta
        un orden dado
    Input:
      1: f = func. de una variable cuyo desarrollo se va a calcular
      2: x = variable
      3: x0 = punto en torno al que desarrollamos
      4: n = orden del desarrollo
    Output:
      lista de desarrollos de f(x) en serie de potencias de (x - x0) desde
          orden 0 hasta orden n */

taylorlist(f, x, x0, n) := block( [TL, aux],
```

```
  /* asignamos la func. f a la variable auxiliar aux */

  aux : f,

  /* definimos la variable auxiliar TL como una lista de un solo elemento
     , dado por
     el primer elemento del desarrollo en serie de f(x) (es decir, f(x0))
         */

  TL : [ subst(x0, x, aux) ],

  /* en este bucle vamos evaluando los sucesivos elementos del desarrollo
      y los vamos
     incluyendo en la lista TL como elementos adicionales mediante el
        comando append */

  for i : 1 thru n step 1 do (

    /* vamos calculando sucesivamente las derivadas de orden desde 1
       hasta n */

    aux : diff(aux,x),

    /* cada una de estas derivadas evaluadas en x = x0 y multiplicadas
       por
              (1/i!)*(x - x0)^i
       nos proporciona el elemento siguiente del desarrollo
       construimos la lista de desarrollos pedida annadiendo al
          desarrollo anterior
       el desarrollo siguiente, dado por el anterior mas el nuevo
          elemento */

    TL : append( TL, [ TL[i] + (1/i!) * subst(x0, x, aux) * (x - x0)^i ] )

  ), /* fin del bucle */

  TL

)$

/* fin */
```