PRÁCTICAS DE CÁLCULO NUMÉRICO I

PRÁCTICA 3: interpolación mediante diferencias divididas

El objetivo de esta práctica es la de implementar el algoritmo estudiado en clase para la evaluación de diferencias divididas. En un primer lugar, consideraremos el caso más simple de interpolación de Lagrange, en la que todos los nodos son distintos. En la segunda parte de esta práctica mejoraremos el algoritmo de diferencias para que también pueda aplicarse al caso de interpolación de Hermite.

Recordemos el algoritmo para generar diferencias divididas:

Algoritmo: Diferencias divididas
Input: x_j, $j = 0, ..., n$, $f_j = f(x_j)$.
Output: $f_j = f[x_0...x_j]$, $j = 0...n$.

(1) i=0;

(2) Repetir mientras $i < n$

(3) j=n+1;i=i+1

(4) Repetir mientras $j > i$

(5) j=j-1

(6) $f_j = (f_j - f_{j-1})/(x_j - x_{j-i})$

(7) Ir a **(4)**

(8) Ir a **(2)**

1 Interpolación de Lagrange

Describimos a continuación los tres ficheros MATLAB a desarrollar en esta primera parte de la práctica.

1. El programa de cálculo de diferencias divididas tendrá la siguiente sintaxis

   ```
   c=difdiv(x,y);
   ```

   donde x es el vector fila conteniendo los nodos de interpolación $x(i)$; y es el vector fila conteniendo los valores de la función que se interpola ($y(i) = f(x(i))$) y $c(i)$ es el vector fila de diferencias divididas $c(i) = f[x(1), ..., x(i)]$.

   Se recomienda utilizar algún ejemplo sencillo de los expuestos en clase como prueba del funcionamiento del algoritmo. Para facilitar la tarea de comprobación, proporcionamos a continuación un ejemplo explícito (no tan sencillo de calcular "a mano"):

   ```
   >> format long e
   >> c=difdiv([-2.2 -1 2.5 3 6 6.3],[0 -1.5 3 2 6.6 -11.2]);
   >> c
   ```

```
c =

  Columns 1 through 3

                         0    -1.250000000000000e+00     5.395136778115501e-01

  Columns 4 through 6

    -2.617196633154080e-01     6.381515721533514e-02    -9.422853510900782e-02
```

Hasta que no obtengamos el mismo resultado más vale que no continuemos con la práctica.

2. Construiremos también un programa para evaluar el polinomio de interpolación, una vez obtenidas las diferencias divididas. Esta función, que llamaremos **evalpol.m**, dado un conjunto de valores de las abscisas X (como vector) proporciona los valores Y (como vector) que alcanza el polinomio de interpolación para esos valores de las abscisas. La sintaxis será

   ```
   Y=evalpol(X,c,xi);
   ```

   donde X es el vector de las abscisas donde se va a evaluar el polinomio, c es el vector de las diferencias divididas (generadas por la función **difdiv.m**) y xi es el vector de los nodos.

   Por ejemplo,

   ```
   >> xi=[1 2 3];c=difdiv(xi,[1 4 9]);
   >> Y=evalpol([0:7],c,xi)

   Y =

        0     1     4     9    16    25    36    49
   ```

3. Finalmente realizaremos un ejercicio práctico para ilustrar gráficamente el funcionamiento de los dos anteriores programas. Se trata de construir el polinomio de interpolación que pase por un conjunto de puntos (al menos 4), que escogeremos libremente, y representar en la misma gráfica el polinomio de intepolación y los puntos (x_i, y_i) de partida, comprobando así que el polinomio construido mediante **difdiv.m** y evaluado mediante **evalpol.m** es el polinomio de interpolación para esos puntos.

2 Interpolación de Hermite

Se trata ahora de construir el algoritmo de diferencias divididas para interpolación de Hermite. La sintaxis de la rutina será:

```
c=difdivH(x,y);
```

donde x es el vector fila de los nodos (contando repeticiones); si hay n nodos y condiciones hasta la k-ésima derivada, y será la matriz $(k+1)\times n$ que en su primera fila tiene los valores de $f(x)$ en cada nodo, en la segunda los valores de la derivada en cada nodo, etc.

Así, por ejemplo, para el problema de interpolar $f(x)=1/(1+x)$ en los nodos $x_0=0$ y $x_1=1$ exigiendo que el polinomio interpolador coincida hasta la primera derivada con la función en cada nodo, las entradas serían: $x=[x_0\,x_0\,x_1\,x_1]=[0\,0\,1\,1]$, $y=[f(x_0)\,f(x_0)\,f(x_1)\,f(x_1)\,;\,f'(x_0)\,f'(x_0)\,f'(x_1)\,f'(x_1)]=[1\,1\,0.5\,0.5\,;\,-1\,-1\,-0.25\,-0.25]$. En este caso, el programa debería dar la siguientes salidas:

```
>> x=[0 0 1 1];y=[1 1 0.5 0.5;-1 -1 -0.25 -0.25];
>> c=difdivH(x,y);
>> c

c =

    1.0000   -1.0000    0.5000   -0.2500
```

y el polinomio de interpolación sería $P_3(x)=1-(x-x_0)+0.5(x-x_0)^2-0.25(x-x_0)^2(x-x_1)=1-x+0.5x^2-0.25x^2(x-1)$.

Actividades: Combinando la utilización de **difdivH.m** y **evalpol.m**, deberemos calcular las anteriores diferencias divididas y dibujar en la misma gráfica el polinomio de interpolación correspondiente y $f(x)=1/(1+x)$. Dibujar en otra gráfica el error, es decir, la diferencia entre $f(x)$ y el polinomio de interpolación de Hermite. Repetir con el polinomio de grado 5 que interpola en 0 y 1 con condiciones hasta la segunda derivada. Discutir el resultado y comparar con las cotas de error a partir del teorema del resto.