Tema 2 - Operaciones matemáticas básicas

Técnicas de Interpolación II

M. en C. Gustavo Contreras Mayén

13 de septiembre de 2017

Contenido

Contenido

Contenido

Contenido

Contenido

Interpolación de Newton

Aunque el método de interpolación de Lagrange es conceptualmente sencillo, no es en sí, un algoritmo eficiente.

Un mejor método computacional se obtiene con el Método de Newton, donde el polinomio de interpolación se escribe de la forma:

$$P_n(x) = a_0 + (x - x_0)a_1 + (x - x_0)(x - x_1)a_2 + \ldots + \\ + (x - x_0)(x - x_1)\ldots(x - x_{n-1})a_n$$

Este polinomio nos permite contar con un procedimiento de evaluación más eficiente.

Por ejemplo, con cuatro pares de datos ($n = 3$), tenemos que el polinomio de interpolación es:

$$\begin{aligned} P_3(x) &= a_0 + (x - x_0)a_1 + (x - x_0)(x - x_1)a_2 + \\ &+ \quad (x - x_0)(x - x_1)(x - x_2)a_3 \\ P_3(x) &= a_0 + \{a_1 + (x - x_1)\left[a_2 + (x - x_2)a_3\right]\} \end{aligned}$$

que puede ser evaluado hacia atrás con las siguientes relaciones de recurrencia:

$$
\begin{aligned}
P_0 &= a_3 \\
P_1 &= a_2 + (x - x_2)P_0(x) \\
P_2 &= a_1 + (x - x_1)P_1(x) \\
P_3 &= a_0 + (x - x_0)P_2(x)
\end{aligned}
$$

$$
\begin{aligned}
P_0 &= a_3 \\
P_1 &= a_2 + (x - x_2)P_0(x) \\
P_2 &= a_1 + (x - x_1)P_1(x) \\
P_3 &= a_0 + (x - x_0)P_2(x)
\end{aligned}
$$

Para un n arbitrario, tenemos:

$$
\begin{aligned}
P_0 &= a_3 \\
P_1 &= a_2 + (x - x_2)P_0(x) \\
P_2 &= a_1 + (x - x_1)P_1(x) \\
P_3 &= a_0 + (x - x_0)P_2(x)
\end{aligned}
$$

Para un n arbitrario, tenemos:

$$
\begin{aligned}
P_0(x) &= a_n \\
P_k &= a_{n-k} + (x - x_{n-k})P_{k-1}(x), \qquad k = 1, 2, \ldots, n
\end{aligned}
$$

Definimos `xDatos` para las coordenadas x del conjunto de puntos y n al grado de polinomio, podemos usar el siguiente algoritmo para calcular $P_n(x)$:

```
1 p = a[n]
2 for k in range(1,n+1):
3     p = a[n-k] + (x -xDatos[n-k])*p
```

Los coeficientes de P_n se calculan forzando que el polinomio pase a través del conjunto de puntos $y_i = P_n(x_i), \quad i = 0, 1, \ldots, n$. De tal manera que tenemos un sistema de ecuaciones simultáneas:

$$
\begin{aligned}
y_0 &= a_0 \\
y_1 &= a_0 + (x_1 - x_0)a_1 \\
y_2 &= a_0 + (x_2 - x_0)a_1 + (x_2 - x_0)(x_2 - x_1)a_2 \\
&\vdots \\
y_n &= a_0 + (x_n - x_0)a_1 + \ldots + \\
&\quad + (x_n - x_0)(x_n - x_1)\ldots(x_n - x_{n-1})a_n
\end{aligned}
$$

Diferencias divididas

Se introducen las diferencias divididas, de la siguiente forma:

$$\nabla y_i = \frac{y_i - y_0}{x_i - x_0}, \qquad i = 1, 2, \ldots, n$$

Diferencias divididas

Se introducen las diferencias divididas, de la siguiente forma:

$$\nabla y_i = \frac{y_i - y_0}{x_i - x_0}, \qquad i = 1, 2, \ldots, n$$

$$\nabla^2 y_i = \frac{\nabla y_i - \nabla y_1}{x_i - x_1}, \qquad i = 1, 2, \ldots, n$$

Diferencias divididas

Se introducen las diferencias divididas, de la siguiente forma:

$$\nabla y_i = \frac{y_i - y_0}{x_i - x_0}, \qquad i = 1, 2, \ldots, n$$

$$\nabla^2 y_i = \frac{\nabla y_i - \nabla y_1}{x_i - x_1}, \qquad i = 1, 2, \ldots, n$$

$$\nabla^3 y_i = \frac{\nabla^2 y_i - \nabla^2 y_2}{x_i - x_2}, \qquad i = 1, 2, \ldots, n$$

Diferencias divididas

Se introducen las diferencias divididas, de la siguiente forma:

$$\nabla y_i = \frac{y_i - y_0}{x_i - x_0}, \qquad i = 1, 2, \ldots, n$$

$$\nabla^2 y_i = \frac{\nabla y_i - \nabla y_1}{x_i - x_1}, \qquad i = 1, 2, \ldots, n$$

$$\nabla^3 y_i = \frac{\nabla^2 y_i - \nabla^2 y_2}{x_i - x_2}, \qquad i = 1, 2, \ldots, n$$

$$\vdots$$

Diferencias divididas

Se introducen las diferencias divididas, de la siguiente forma:

$$\nabla y_i = \frac{y_i - y_0}{x_i - x_0}, \qquad i = 1, 2, \ldots, n$$

$$\nabla^2 y_i = \frac{\nabla y_i - \nabla y_1}{x_i - x_1}, \qquad i = 1, 2, \ldots, n$$

$$\nabla^3 y_i = \frac{\nabla^2 y_i - \nabla^2 y_2}{x_i - x_2}, \qquad i = 1, 2, \ldots, n$$

$$\vdots$$

$$\nabla^n y_i = \frac{\nabla^{n-1} y_n - \nabla^{n-1} y_{n-1}}{x_n - x_{n-1}}$$

La solución al sistema de ecuaciones es:

$$
\begin{aligned}
a_0 &= y_0 \\
a_1 &= \nabla y_1 \\
a_2 &= \nabla^2 y_2 \\
&\vdots \\
a_n &= \nabla^n y_n
\end{aligned}
$$

Si los coeficientes se calculan a mano, es conveniente escribirlos con el siguiente formato: (con $n = 4$)

x_0	y_0				
x_1	y_1	∇y_1			
x_2	y_2	∇y_2	$\nabla^2 y_2$		
x_3	y_3	∇y_3	$\nabla^2 y_3$	$\nabla^3 y_3$	
x_4	y_4	∇y_4	$\nabla^2 y_4$	$\nabla^3 y_4$	$\nabla^4 y_4$

Los términos en la diagonal $(y_0, \nabla y_1, \nabla^2 y_2, \nabla^3 y_3, \nabla^4 y_4)$ son los coeficientes del polinomio.

Si los puntos de datos se enumeran en un orden diferente, las entradas de la tabla van a cambiar, pero el polinomio resultante será el mismo, recordemos que un polinomio de interpolación de grado n con $n + 1$ datos diferentes, es único.

Las operaciones en la computadora se pueden realizar con un arreglo unidimensional a, usando el siguiente algoritmo (tomando la notación $m = n + 1 =$ número de puntos):

```
1 a = yDatos.copy()
2 for k in range(1,m):
3     for i in range(k,m):
4         a[i] = (a[i] - a[k-1])/(xDatos[i] -
              xDatos[k-1])
```

Inicialmente el arreglo a contiene las coordenadas y del conjunto de datos, es decir, la segunda columna de la tabla.

Cada vez que pasa por el bucle externo, se genera la siguiente columna, por lo que se sobre-escriben los elementos de a, por tanto, al concluir el bucle, a contiene los elementos de la diagonal, que son los coeficientes del polinomio.

Módulos con Python

Para facilitar el mantenimiento y la lectura los programas demasiado largos pueden dividirse en módulos, agrupando elementos relacionados.

Los módulos son entidades que permiten una organización y división lógica de nuestro código. Los archivos son su contrapartida física: cada archivo de Python almacenado en disco equivale a un módulo.

Módulo `newtonPoli`

Este módulo incluye dos funciones que se requieren para la interpolación de Newton.

Dados el conjuntos de puntos en los arreglos `xDatos` y `yDatos`, la función `coeffts` devuelve el arreglo a con los coeficientes.

Una vez que ya conocemos los coeficientes, $P_n(x)$ puede evaluarse para cualquier valor de x con la función `evalPoli`.

```
1  def evalPoli(a,xDatos,x):
2      n = len(xDatos) - 1
3      p = a[n]
4      for k in range(1,n+1):
5          p = a[n-k] + (x -xDatos[n-k])*p
6      return p
7
8  def coeffts(xDatos,yDatos):
9      m = len(xDatos)
10     a = yDatos.copy()
11     for k in range(1,m):
12         a[k:m] = (a[k:m] - a[k-1])/(xDatos[k:m
               ] - xDatos[k-1])
13     return a
```

Ejemplo

Los datos que se muestran en la siguiente tabla se obtuvieron de la función

$$f(x) = 4.8 \cos\left(\frac{\pi x}{20}\right)$$

Con ese conjunto de datos, interpola mediante el polinomio de Newton en
$x = 0, 0.5, 1.0, 1.5, \ldots, 7.5, 8.0$ y compara los resultados con el valor "exacto"de los valores
$y_i = f(x_i)$

x	0.15	2.30	3.15	4.85	6.25	7.95
y	4.79867	4.49013	4.2243	3.47313	2.66674	1.51909

¿Qué necesitamos?

Abriendo un archivo en Spyder, llamamos al módulo `numpy` y también al módulo que contiene las funciones para resolver el polinomio de interpolación:

```
1 from numpy import *
2 from newtonPoli import *
```

Hay que crear los arreglos `xDatos` y `yDatos`, el arreglo a se obtiene de la función `coeffts` que está dentro del módulo `NewtonPoli`

```
1 xDatos = array([0.15,2.3,...,7.95])
2 yDatos = array([4.79867,4.49013,...,1.51909])
3
4 a = coeffts(xDatos, yDatos)
```

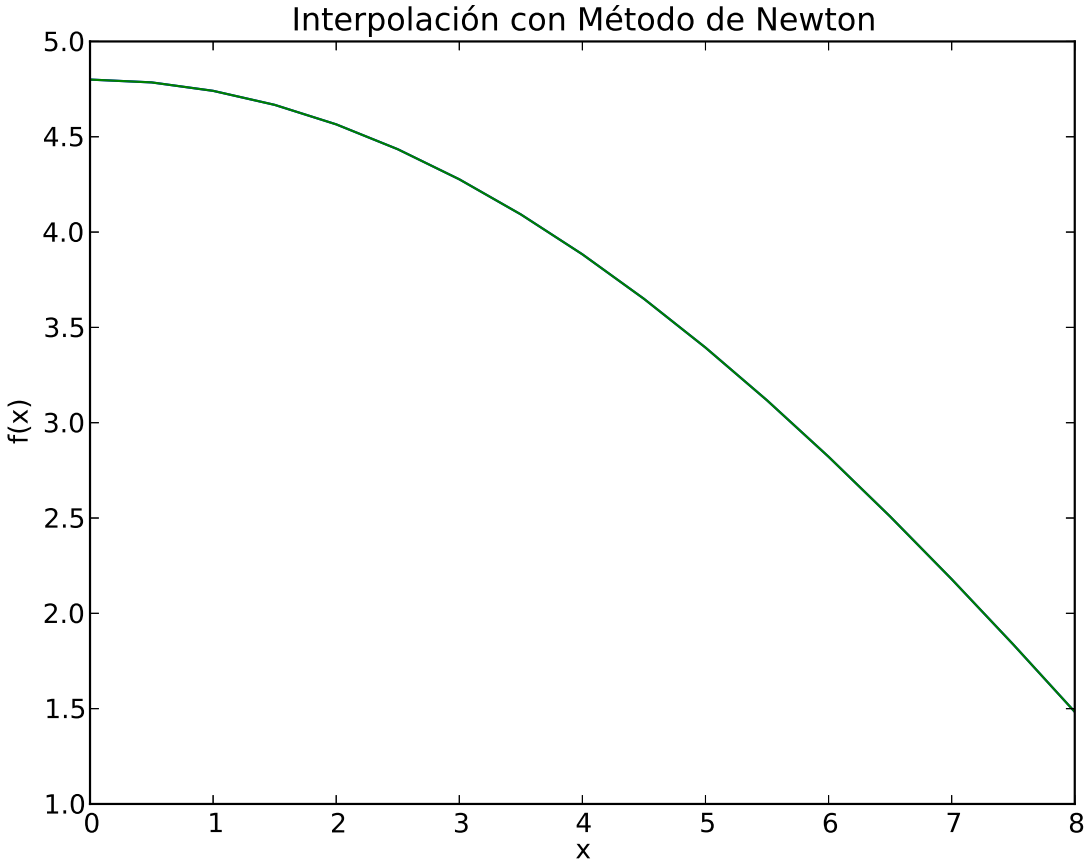
La siguiente parte es proporcionar el rango de puntos y mandar llamar la función `evalPoli`:

```
1 print 'x      yInterp      yExacta'
2 print '-------------------------------'
3 for x in arange(0.0,8.1,0.5):
4     y = evalPoli(a,xDatos,x)
5     yExacta = 4.8*cos(pi*x/20.0)
6     print '%3.1f %9.5f %9.5f' %(x,y,yExacta)
```

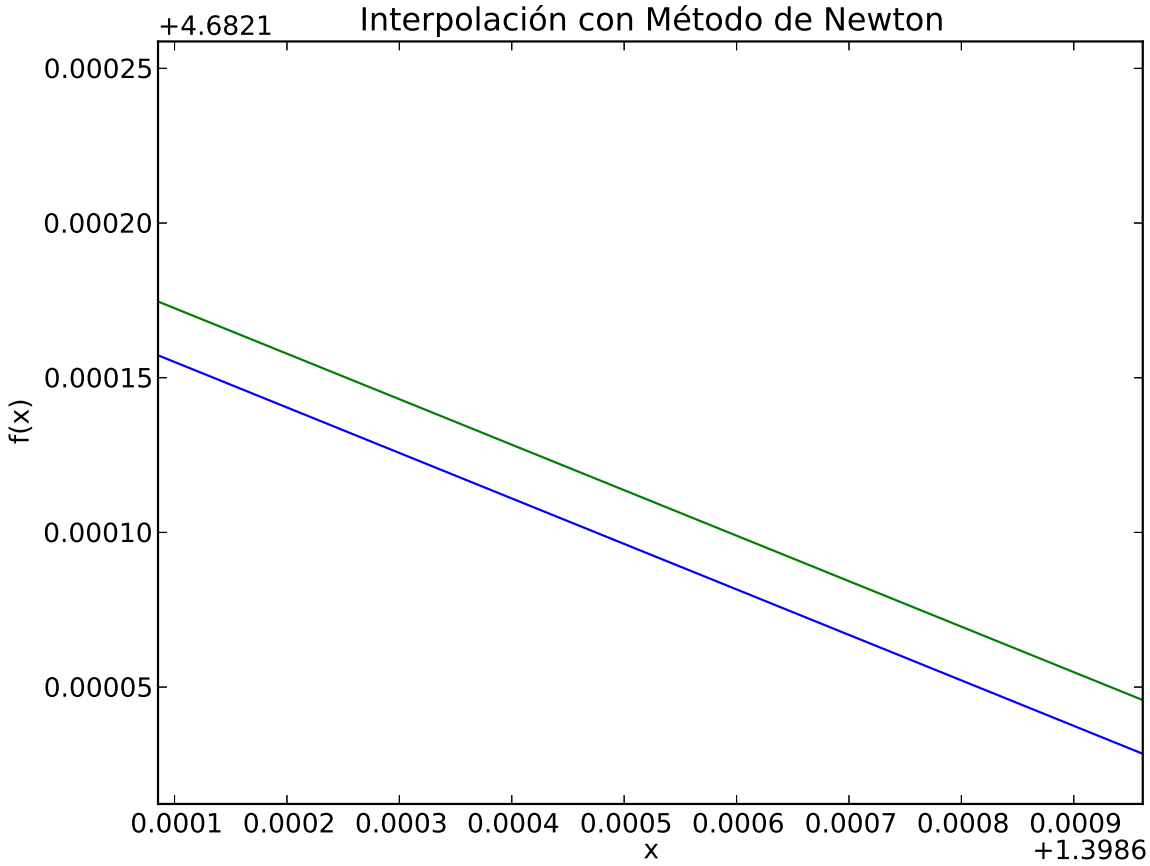
Al ejecutar el código obtenemos lo siguiente:

x	yInterp	yExacta
0.0	4.80003	4.80000
0.5	4.78518	4.78520
1.0	4.74088	4.74090
1.5	4.66736	4.66738
⋮		
7.0	2.17915	2.17915
7.5	1.83687	1.83688
8.0	1.48329	1.48328

Graficando la función exacta y los puntos



Graficando la función exacta y los puntos



Contenido

Método de Neville

Ya vimos que el método de interpolación de Newton consiste en dos pasos:

1. El cálculo de los coeficientes.
2. La evaluación del polinomio.

Esto funciona bien si la interpolación se lleva a cabo repetidamente para diferentes valores de x usando el mismo polinomio. Si sólo hay un punto interpolado, el método de Neville es un método que calcula la interpolación en un solo paso, siendo una mejor opción.

Sea

$$P_k[x_i, x_{i+1}, x_{i+k}]$$

el polinomio de orden k que pasa a través de los $k+1$ puntos (x_i, y_i),(x_{i+1}, y_{i+1}),. . .,(x_{i+k}, y_{i+k}).

Para un sólo punto, tenemos

$$P_0[x_i] = y_i$$

La inteporlación con dos puntos es

$$P_1[x_i, x_{i+1}] = \frac{(x - x_{i+1})P_0[x_i] + (x_i - x)P_0[x_{i+1}]}{x_i - x_{i+1}}$$

La inteporlación con dos puntos es

$$P_1[x_i, x_{i+1}] = \frac{(x - x_{i+1})P_0[x_i] + (x_i - x)P_0[x_{i+1}]}{x_i - x_{i+1}}$$

La inteporlación con tres puntos es

La inteporlación con dos puntos es

$$P_1[x_i, x_{i+1}] = \frac{(x - x_{i+1})P_0[x_i] + (x_i - x)P_0[x_{i+1}]}{x_i - x_{i+1}}$$

La inteporlación con tres puntos es

$$P_2[x_i, x_{i+1}, x_{i+2}] = \frac{(x - x_{i+2})P_1[x_i, x_{i+1}] + (x_i - x)P_1[x_{i+1}, x_{i+2}]}{x_i - x_{i+2}}$$

Para mostrar que esta interpolación intersecta los puntos, sustituimos primero $x = x_i$, para obtener

$$P_2[x_i, x_{i+1}, x_{i+2}] = P_1[x_i, x_{i+1}] = y_i$$

Para mostrar que esta interpolación intersecta los puntos, sustituimos primero $x = x_i$, para obtener

$$P_2[x_i, x_{i+1}, x_{i+2}] = P_1[x_i, x_{i+1}] = y_i$$

del mismo modo, $x = x_{i+2}$

Para mostrar que esta interpolación intersecta los puntos, sustituimos primero $x = x_i$, para obtener

$$P_2[x_i, x_{i+1}, x_{i+2}] = P_1[x_i, x_{i+1}] = y_i$$

del mismo modo, $x = x_{i+2}$

$$P_2[x_i, x_{i+1}, x_{i+2}] = P_1[x_i, x_{i+2}] = y_{i+2}$$

Para mostrar que esta interpolación intersecta los puntos, sustituimos primero $x = x_i$, para obtener

$$P_2[x_i, x_{i+1}, x_{i+2}] = P_1[x_i, x_{i+1}] = y_i$$

del mismo modo, $x = x_{i+2}$

$$P_2[x_i, x_{i+1}, x_{i+2}] = P_1[x_i, x_{i+2}] = y_{i+2}$$

finalmente, cuando $x = x_{i+1}$, tenemos

Para mostrar que esta interpolación intersecta los puntos, sustituimos primero $x = x_i$, para obtener

$$P_2[x_i, x_{i+1}, x_{i+2}] = P_1[x_i, x_{i+1}] = y_i$$

del mismo modo, $x = x_{i+2}$

$$P_2[x_i, x_{i+1}, x_{i+2}] = P_1[x_i, x_{i+2}] = y_{i+2}$$

finalmente, cuando $x = x_{i+1}$, tenemos

$$P_1[x_i, x_{i+1}] = P_1[x_{i+1}, x_{i+2}] = y_{i+1}$$

Entonces

$$P_2[x_i, x_{i+1}, x_{i+2}] = \frac{(x_{i+1} - x_{i+2})y_{i+1} + (x_i - x_{i+1})y_{i+1}}{x_i - x_{i+2}} = y_{i+1}$$

Habiendo deducido el patrón, podemos establecer la fórmula recursiva:

$$P_k[x_i, x_{i+1}, \ldots, x_{i+k}] =$$

$$= \frac{(x - x_{i+k})P_{k-1}[x_i, x_{i+1}, \ldots, x_{i+k}] + (x_i - x)P_{k-1}[x_{i+1}, x_{i+2}, \ldots, x_{i+k}]}{x_i - x_{i+k}}$$

Dando el valor de x, los cálculos se pueden anotar en un formato de tabla (se muestran para cuatro puntos):

	$k=0$	$k=1$	$k=2$	$k=3$
x_0	$P_0[x_0] = y_0$	$P_1[x_0, x_1]$	$P_2[x_0, x_1, x_2]$	$P_3[x_0, x_1, x_2, x_3]$
x_1	$P_0[x_1] = y_1$	$P_1[x_1, x_2]$	$P_2[x_1, x_2, x_3]$	
x_2	$P_0[x_2] = y_2$	$P_1[x_2, x_3]$		
x_3	$P_0[x_3] = y_3$			

Si m es el número de datos, el algoritmo que calcula los elementos de la tabla es:

```
1 y = yDatos.copy()
2 for k in range(1,m):
3     for i in range(m-k):
4         y[i] = ((x-xDatos[i+k])*y[i]+(xDatos[i
              ]-x)*y[i+1])/(xDatos[i] - xDatos[i+
              k])
```

Módulo Neville

La siguiente función implementa el método de interpolación de Neville, devuelve $P_n(x)$:

```
1 def neville(xDatos,yDatos,x):
2     m = len(xDatos)
3     y = yDatos.copy()
4     for k in range(1,m):
5         y[0:m-k] = ((x - xDatos[k:m])*y[0:m-k]
                + \
6         (xDatos[0:m-k] - x)*y[1:m-k+1])/ \
7         (xDatos[0:m-k] - xDatos[k:m])
8     return y[0]
```

Contenido

Ejemplo

Dados los puntos de la siguiente tabla:

x	4.0	3.9	3.8	3.7
y	-0.06604	-0.02724	0.01282	0.05383

Calcular la raíz de $y(x) = 0$ con el método de interpolación de Neville.

Este es un ejemplo de *interpolación inversa*, donde los roles de x e y se intercambian, ya que se calcula un y para un x dado; encontrar ahora x corresponde a un valor de y dado, en este caso $y = 0$.

Usa el módulo de Neville para encontrar la raíz. El valor es 3.83170355972

Contenido

Cuatro Ejercicios

1. Usa el método de interpolación de Neville para calcular y en $x = \pi/4$ del conjunto de datos

x	0	0.5	1	1.5	2
y	-1.00	1.75	4.00	5.75	7.00

2. Dados los puntos

x	0	0.5	1	1.5	2
y	-0.7854	0.6529	1.7390	2.2071	1.9425

calcular y en $x = \pi/4$ y en $x = \pi/2$. Usa el método que consideres más conveniente.

3. Los puntos

x	-2	1	4	-1	3	-4
y	-1	2	59	4	24	-53

se obtuvieron de un polinomio. Usando las diferencias divididas de la tabla del método de Newton, determina el grado del polinomio.

4. Escribe un programa que use el método de interpolación de Neville para que realice el cálculo de varios valores de x. Determina y en $x = 1.1, 1.2, 1.3$ de los siguientes datos:

x	-2.0	-0.1	-1.5	0.5
y	2.2796	1.0025	1.6467	1.0635

x	-0.6	2.2	1.0	1.8
y	1.0920	2.6291	1.2661	1.9896

Respuesta: $y = 1.3262, 1.3938, 1.4693$