Interpolación Polinomial: Forma modificada de Lagrange vs diferencias divididas de Newton.

Walter Mora F.
wmora2@yahoo.com.mx
Escuela de Matemática
Instituto Tecnológico de Costa Rica

Palabras claves: Métodos numéricos, interpolación polinomial, polinomio de Lagrange, diferencias divididas de Newton, polinomio de TChebyshev.

Introducción

Usualmente se reserva la forma de Lagrange del polinomio interpolante para trabajo teórico y diferencias divididas de Newton para cálculos. La realidad es que hay una forma modificada de Lagrange que es tan eficiente como diferencias divididas de Newton en cuanto a costo computacional y además es numéricamente mucho más estable. Hay varias ventajas que hacen de esta forma modificada de Lagrange, el método a escoger cuando de interpolación polinomial se trata. La exposición se basa principalmente en ([9]) y ([11]).

La interpolación polinomial es la base de muchos tipos de integración numérica y tiene otras aplicaciones teóricas, de ahí nuestro interés en este tópico.
En la práctica a menudo tenemos una tabla de datos obtenida por muestreo o experimentación. Suponemos que los datos corresponden a los valores de una función f desconocida (a veces es conocida, pero queremos cambiarla por una función más sencilla de calcular). El "ajuste de curvas" trata el problema de construir una función que aproxime muy bien estos datos (es decir, a f). Un caso particular de ajuste de curvas es la interpolación polinomial: En este caso se construye un polinomio P que pase por los puntos de la tabla. La interpolación polinomial consiste en estimar un valor ausente $f(x^*)$ en la tabla con el valor $P(x^*)$.

EJEMPLO 1.1 Considere los siguientes datos para el nitrógeno(N_2):

$T(K)$	100	200	300	400	450	500	600
$B(cm^3/mol)$	-160	-35	-4.2	9.0	?	16.9	21.3

Tabla 1.1 Segundos Coeficientes viriales $B(cm^3/mol)$ para el nitrógeno

donde T es la temperatura y B es el segundo coeficiente virial[1]. ¿Cuál es el segundo coeficiente virial a $450K$?. Para responder la pregunta, usando interpolación polinomial, construimos un polinomio P que pase por los seis puntos de la tabla (ya veremos cómo), tal y como se muestra en la figura (1.1). Luego, el segundo coeficiente virial a $450K$ es aproximadamente $P(450) = 13.5cm^3/mol$.

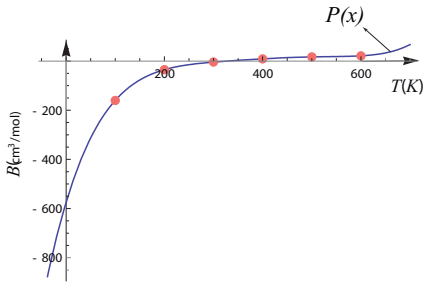

Figura 1.1 Polinomio interpolante P para la tabla de segundos coeficientes viriales.

[1] El comportamiento de gases no ideales se describe a menudo con la *ecuación virial de estado*

$$\frac{PV}{RT} = 1 + \frac{B}{V} + \frac{C}{V^2} + ...,$$

donde P es la presión, V el volumen molar del gas, T es la temperatura Kelvin y R es la constante de gas ideal. Los coeficientes $B = B(T)$, $C = C(T)$,... son el segundo y tercer coeficiente virial, respectivamente. En la práctica se usa la serie truncada

$$\frac{PV}{RT} \approx 1 + \frac{B}{V}$$

EJEMPLO 1.2 Consideremos la función f definida por

$$f(x) = \int_5^\infty \frac{e^{-t}}{t-x}\,dt, \quad -1 \leq x \leq 1$$

La integral que define a f es una integral no trivial (no se puede expresar en términos de funciones elementales). La tabla 1.2 nos muestra algunos valores para f.

x	$f(x)$
-1	0.0009788055864607286
-0.6	0.0010401386051341144
-0.2	0.0011097929435687336
0	0.0011482955912753257
0.2	0.0011896108201581322
0.25	?
0.6	0.0012820294923443982
1.	0.0013903460525251596

Tabla 1.2

Podemos usar un polinomio interpolante para interpolar $f(0.25)$.

En el mundillo del ajuste de curvas hay varias alternativas,

- Usar un polinomio interpolante. Es el método de propósito general más usado.
- Usar trazadores (splines). Estas son funciones polinomiales a trozos.
- Usar polinomios trigonométricos en $[0, 2\pi]$. Son la elección natural cuando la función f es periódica de periodo 2π.
- Usar sumas exponenciales. Se usan si conocemos que f presenta decaimiento exponencial conforme $x \longrightarrow \infty$.
- Si los datos son aproximados ("datos experimentales"), lo conveniente sería usar *Mínimos Cuadrados*

Un problema de interpolación polinomial se especifica como sigue: dados $n+1$ puntos $(x_0, y_0), (x_1, y_1), ..., (x_n, y_n)$, siendo todos los x_i's distintos, encontrar un polinomio $P_n(x)$ de grado $\leq n$ tal que

$$P_n(x_i) = y_i, \quad i = 0, 1, 2, ..., n \tag{1.1}$$

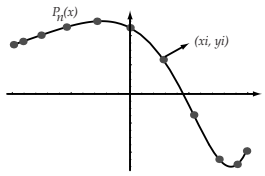

Figura 1.2 $P_n(x)$

A $P_n(x)$ se le llama *polinomio interpolante* y a cada x_i le decimos *nodo de interpolación.*

Teorema 1.1 (Existencia y unicidad) *Sean $x_0, x_1, ..., x_n$ números reales distintos y $n \geq 0$. Si $y_0, y_1, ..., y_n$ son números reales arbitrarios, existe un único polinomio P_n, de grado menor o igual a n tal que $P_n(x_i) = y_i$, $i = 0, 1, ..., n$.*

La unicidad es consecuencia del teorema fundamental del álgebra: Si tenemos dos polinomios P y q de grado a lo sumo n, entonces $P - Q$ tiene grado a los sumo n. Como $P(x_i) - Q(x_i) = 0$, $i = 0, 1, ..., n$ este polinomio tendría $n+1$ ceros distintos, es decir, debe ser el polinomio nulo, $\therefore P = Q$.

La existencia se puede probar exhibiendo el polinomio en la forma de Lagrange, por ejemplo:

$$P_n(x) = y_0 L_{n,0}(x) + y_1 L_{n,1}(x) + ... + y_n L_{n,n}(x) \text{ con } L_{n,k}(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x - x_i}{x_k - x_i}$$

EJEMPLO 1.3 El polinomio de grado ≤ 2 que pasa por $(0,1), (1,3), (2,0)$ es

$$\begin{aligned} P_2(x) &= y_0\, L_{2,0}(x) + y_1\, L_{2,1}(x) + y_2\, L_{2,1}(x) \\ &= 1 \cdot L_{2,0}(x) + 3 \cdot L_{2,1}(x) + 0 \cdot L_{2,2}(x) \\ &= 1 \cdot \frac{(x-1)(x-2)}{(0-1)(0-2)} + 3 \cdot \frac{(x-0)(x-2)}{(1-0)(1-2)}. \quad \text{Simplificando,} \\ &= -\frac{5x^2}{2} + \frac{9x}{2} + 1 \end{aligned}$$

EJEMPLO 1.4 (Tipos de nodos) El polinomio interpolante se ve afectado por los nodos $\{x_i\}_{i=0,1,\ldots,n} \subseteq [a,b]$.

Un ejemplo extremo es la función de Runge: Si $f(x) = \dfrac{1}{1+25x^2}$, el polinomio inteporlante presenta problemas de convergencia si tomamos los nodos $\{x_i\}_{i=0,1,\ldots,n}$ igualmente espaciados en $[-1,1]$, es decir $x_i = a + ih$ con $h = 2/n$.

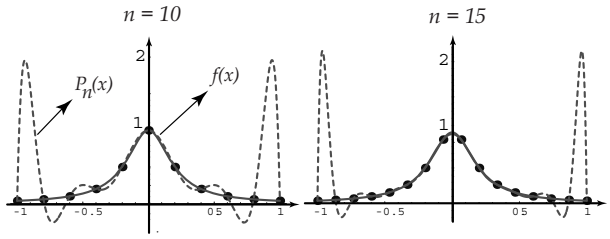

Figura 1.3 $P_n(x)$ obtenido con puntos igualmente espaciados.

Si tomamos nodos de Tchebyshev (sección 1.3) $x_i = \cos\left(\dfrac{2i+1}{2n+2}\,\pi\right)$, se eliminan los problemas de convergencia (y este resultado es válido para cualquier función $f \in C^1[-1,1]$.)

En ejemplo que sigue se muestran las formas de Newton del polinomio interpolante, denotado $PN(x)$ (en rojo) y la forma modificada de Lagrange (en verde)

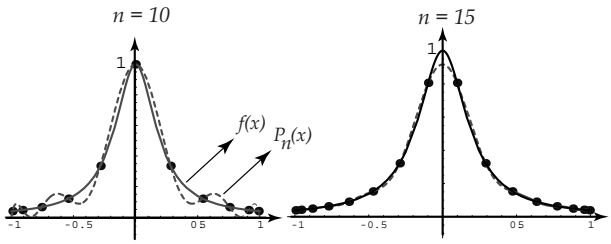

Figura 1.4 $P_n(x)$ con nodos de TChebyshev $x_i = \cos((2i+1)/(2n+2)\,\pi)$.

del polinomio interpolante, denotado $PML(x)$. Ambas representaciones se establecen en la sección (1.1). La forma de Lagrange se confunde con la función mientras que la forma de Newton sufre de problemas de inestabilidad en las cercanías de -1.

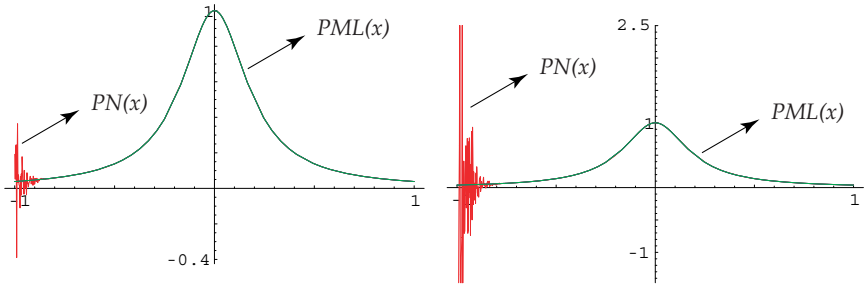

Figura 1.5 $f(x) = 1/(1+25x^2)$, $PN(x)$ y $PML(x)$ con 55 y 60 nodos de TChebyshev

1.1 Forma de Lagrange y Forma de Newton del polinomio interpolante.

Aunque el polinomio interpolante es único, existen varias representaciones como combinación lineal de ciertos polinomios. Sea $\mathbb{R}_{n+1}[x]$ el conjunto de polinomios con coeficientes reales, de grado $\leq n$. $\mathbb{R}_{n+1}[x]$ es un espacio vectorial de dimensión $n+1$. Si $\Omega = \{B_0, B_2, ..., B_n\}$ es una base para $\mathbb{R}_{n+1}[x]$, entonces el polinomio interpolante P_n se puede escribir en la base Ω como

$$P_n(x) = a_1 B_0(x) + ... + a_n B_n(x)$$

Aunque el polinomio interpolante es único, la forma (antes de simplificar) puede cambiar según la base que se tome. Si tomamos la base $\{1, x, ..., x^n\}$ de $\mathbb{R}_{n+1}[x]$, entonces

$$P_n(x) = a_0 + a_1 x + ... + a_n x^n$$

Como $P_n(x_i) = y_i,\ i = 0, 1, ..., n,$ tenemos el sistema de ecuaciones lineales

$$\begin{pmatrix} 1 & x_0 & x_0^2 & ... & x_0^n \\ 1 & x_1 & x_1^2 & ... & x_1^n \\ \vdots & \vdots & & \cdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix} \tag{1.2}$$

La matriz asociada V_n del sistema 1.3 se llama matriz de Vandermonde. Las columnas de V_n conforman un conjunto de vectores linealmente independiente. Luego, podemos obtener los coeficientes a_i usando la regla de Cramer. Aunque teóricamente todo funciona bien, computacionalmente es muy costoso calcular el polinomio interpolante de esta manera.

1.1.1 Forma de Lagrange para el polinomio interpolante.

La forma de Lagrange del polinomio interpolante se obtiene usando la base $L_{n,0}(x), ..., L_{n,n}(x)$ donde $L_{n,k}(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x - x_i}{x_k - x_i}$. En este caso

$$P_n(x) = a_0 L_{n,0}(x) + a_1 L_{n,1}(x) + ... + a_n L_{n,n}(x)$$

Como $P_n(x_i) = y_i,\ i = 0, 1, ..., n$ y como $L_{n,j}(x_i) = \delta_{ij}$ (delta de Kronecker), tenemos,

$$\begin{pmatrix} 1 & 0 & 0 & ... & 0 \\ 0 & 1 & 0 & ... & 0 \\ \vdots & \vdots & & \cdots & \vdots \\ 0 & 0 & & \cdots & 1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix} \implies a_i = y_i,\ i = 0, 1, ..., n. \quad (1.3)$$

Luego $P_n(x) = y_0 L_{n,0}(x) + y_1 L_{n,1}(x) + ... + y_n L_{n,n}(x)$. Esta es la llamada forma de Lagrange de polinomio intepolante.

1.1.2 Forma de Newton para el polinomio interpolante.

La forma de Newton del polinomio interpolante se obtiene usando la base $B_0(x), ..., B_n(x)$ donde

$$\begin{aligned} B_0(x) &= 1 \\ B_1(x) &= (x - x_0) \\ B_2(x) &= (x - x_0)(x - x_1) \\ &\vdots \quad\quad \vdots \\ B_n(x) &= (x - x_0)(x - x_1) \cdots (x - x_{n-1}) \end{aligned}$$

Tenemos entonces

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

Como $P_n(x_i) = y_i,\ i = 0, 1, ..., n$ tenemos un sistema en forma triangular, listo para resolver

$$\begin{array}{rcll} P_n(x_0) & = & a_0 & = y_0 \\ P_n(x_1) & = & a_0 + a_1(x_1 - x_0) & = y_1 \\ P_n(x_2) & = & a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) & = y_2 \\ \vdots & & \vdots & \\ P_n(x_n) & = & a_0 + a_1(x_n - x_0) + ... + a_n(x_n - x_0)\cdots(x_n - x_{n-1}) & = y_n \end{array}$$

Para obtener una fórmula recursiva un poco más limpia, definimos $Q_0 = a_0$ y

$$Q_j(x) = a_0 + a_1(x - x_0) + ... + a_j(x - x_0)\cdots(x - x_{j-1}),\ \ j = 1, ..., n$$

De esta manera,

$$\begin{array}{rcl} a_0 & = & y_0 \\ a_1 & = & \dfrac{y_1 - Q_0(x_0)}{x_1 - x_0} \\ a_2 & = & \dfrac{y_2 - Q_1(x_1)}{(x_2 - x_0)(x_2 - x_1)} \\ \vdots & & \vdots \\ a_n & = & \dfrac{y_n - Q_{n-1}(x_n)}{(x_n - x_0)\cdots(x_n - x_{n-1})} \end{array}$$

Obtenemos una fórmula recursiva: $a_0 = y_0$ y $a_k = \dfrac{y_k - Q_{k-1}(x_k)}{(x_k - x_0)\cdots(x_k - x_{k-1})},\ k = 1, ..., n$

Diferencias divididas.

Si $y_k = f(x_k)$, la fórmula anterior nos muestra que cada a_k depende de $x_0, x_1, ..., x_k$. Desde muchos años atrás se usa la notación "$a_n = f[x_0, x_1, ...x_n]$" para significar esta dependencia.

Al símbolo $f[x_0, x_1, ...x_n]$ se le llama *diferencia divida* de f.

Si consideramos $f[x_0, x_1, ...x_n]$ como una función de $n+1$ variables, entonces es una función *simétrica*, es decir, permutar las variables de cualquier manera no afecta el valor de la función. Esto es así porque el polinomio que interpola los puntos $\{(x_i, y_i)\}_{i=0,...,n}$ es único, por lo tanto sin importar el orden en que vengan los puntos, el coeficiente principal siempre es $a_n = f[x_0, x_1, ...x_n]$.

En notación de diferencias divididas, la forma de Newton del polinomio interpolante es

$$P_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + \cdots + f[x_0, x_1, ...x_n](x - x_0)\cdots(x - x_{n-1}),$$

El nombre *diferencia divida* viene de la agradable propiedad

Teorema 1.2 *La diferencia dividida $f[x_0, x_1, ...x_n]$ satisface la ecuación*

$$f[x_0, x_1, ...x_n] = \frac{f[x_1, x_2, ...x_n] - f[x_0, x_1, ...x_{n-1}]}{x_n - x_0} \tag{1.4}$$

Prueba. Sea $P_k(x)$ el polinomio que interpola f en $x_0, x_1, ..., x_k$. Aquí solo necesitamos $P_n(x)$ y $P_{n-1}(x)$. Sea $R(x)$ el polinomio que interpola f en $x_1, x_2, ..., x_n$. Entonces (ejercicio 1.1)

$$P_n(x) = R(x) + \frac{x - x_k}{x_k - x_0}\left[R(x) - P_{n-1}(x)\right] \tag{1.5}$$

Como en la ecuación (1.5) el polinomio de la izquierda y el de la derecha son idénticos, entonces su coeficiente principal debe ser el mismo, es decir,

$$
\begin{aligned}
f[x_0, x_1, ...x_n]x^n \ + \ ... \ &= \ f[x_1, x_2, ...x_n]x^{n-1} \ + \ ... + \ \frac{x\,R(x) - x\,P_{n-1}(x) \ + \ ...}{x_k - x_0} \\
&= \ f[x_1, x_2, ...x_n]x^{n-1} \ + \ ... + \ \frac{f[x_1, x_2, ...x_n]x^n \ + \ ... \ - f[x_0, x_2, ...x_{n-1}]x^n \ + \ ...}{x_k - x_0} \\
&= \ \frac{(f[x_1, x_2, ...x_n] \ - \ f[x_0, x_2, ...x_{n-1}])x^n}{x_k - x_0} \ + \ ...
\end{aligned}
$$

de donde se obtiene (1.4).

Interpolando sobre $\{(x_i, y_i)\}_{i=k-j,...,k}$ tenemos

$$
f[x_{k-j}, x_{k-j+1}, \cdots, x_k] = \frac{f[x_{k-j+1}, ..., x_k] - f[x_{k-j}, x_1, ..., x_{k-1}]}{x_k - x_{k-j}}
$$

Este esquema recursivo se puede arreglar en forma matricial como sigue,

$$
\begin{array}{llll}
y_0 & = \boldsymbol{a_0} & & \\
y_1 & f[x_0, x_1] = \boldsymbol{a_1} & & \\
y_2 & f[x_1, x_2] & f[x_0, x_1, x_2] = \boldsymbol{a_2} & \\
 & f[x_2, x_3] & f[x_1, x_2, x_3] & \\
\vdots & \vdots & \vdots & \ddots \\
y_n & f[x_{n-1}, x_n] & f[x_{n-2}, x_{n-1}, x_n] & f[x_0, x_1, ..., x_n] = \boldsymbol{a_n}
\end{array}
$$

EJEMPLO 1.5 Consideremos los puntos $(0,1), (1,3), (2,0)$. La matriz de diferencias divididas es

$$
\begin{array}{lll}
1 = a_0 & & \\
0 & 2 = \boldsymbol{a_1} & \\
3 & -3 & -5/2 = \boldsymbol{a_2}
\end{array}
$$

El polinomio interpolante, en la forma de Newton, es

$$P(x) = 1 + 2(x-0) - 5/2(x-0)(x-1) \ = \ -\frac{5x^2}{2} + \frac{9x}{2} + 1$$

1.2 Estimación del error.

Sea $P_n(x)$ el polinomio que interpola f en los puntos $x_0, x_1, ..., x_n$. La fórmula de error debe ser exacta para $x = x_i,\ i = 0, 1, ..., n$ y también debe ser exacta en el caso de que f sea un polinomio de grado inferior o igual a n. Esto sugiere que en la fórmula de error deben aparecer los factores $(x-x_0)(x-x_1)\cdots(x-x_n)$ y f^{n+1}. Probando con $f(x) = x^{n+1}$ se observa que debe aparecer el factor $1/(n+1)!$.

Teorema 1.3 *Sea $f \in C^{n+1}[a,b]$. Sea $P_n(x)$ el polinomio de grado $\leq n$ que interpola f en los $n+1$ puntos (distintos) $x_0, x_1, ..., x_n$ en el intervalo $[a,b]$. Para cada valor fijo $x \in [a,b]$ existe $\xi(x) \in]a,b[$ tal que*

$$f(x) - P_n(x) = \frac{f^{n+1}(\xi(x))}{(n+1)!}(x-x_0)(x-x_1)\cdots(x-x_n)$$

Prueba. Este es un razonamiento muy elegante, debido a Cauchy[2]. Si $x = x_i,\ i = 0, 1, ...n,$ la fórmula de error es correcta. Consideremos un valor fijo $x \in [a,b]$, diferente de cada uno de los nodos $x_i,\ i = 0, 1, ..., n$. Definamos una función g, en la variable t, de la siguiente manera,

2

Agustín Louis Cauchy (1789-1857), padre del análisis moderno. Estableció las bases del análisis matemático basándolo en un concepto riguroso de límite. Fue el creador del análisis complejo. También trabajo en ecuaciones diferenciales, geometría, álgebra, teoría de números, probabilidad y física matemática

$$g(t) = f(t) - P_n(t) \; - \; \frac{f(x) \; - \; P_n(x)}{\prod_{i=0}^{n}(x - x_i)} \prod_{i=0}^{n}(t - x_i)$$

$g \in C^{n+1}[a,b]$ y $g(x) = 0$ y $g(x_i) = 0,\ i = 0, 1, ...n.$

Por tanto g tiene $n+2$ ceros distintos en $[a,b]$. El teorema de Rolle dice que si h es continua en $[a,b]$ y derivable en $]a,b[$, y si $h(a) = h(b) = 0$, entonces $h(\varepsilon) = 0$ para algún $\varepsilon \in]a,b[$. Aplicando repetidamente el teorema de Rolle a la función g en los intervalos $[x_0, x_1]$, $[x_1, x_2]$, ..., concluimos que g' tiene *al menos* $n+1$ ceros distintos en $]a,b[$. De manera similar concluimos que g^{n+1} tiene *al menos* 1 cero distinto en $]a,b[$ (g^{n+1} es continua en [a,b]).

Ahora bien, sea $\xi(x)$ un cero de g^{n+1} en $]a,b[$. Como $\left.\frac{d^{n+1}g}{dt^{n+1}}\right|_{t=\xi(x)} = 0$, tenemos

$$0 = f^{n+1}(\xi(x)) \; - \; \frac{f(x) \; - \; P_n(x)}{\prod_{i=0}^{n}(x - x_i)} \, (n+1)!$$

de donde, $f(x) - P_n(x) = \frac{f^{n+1}(\xi(x))}{(n+1)!}(x - x_0)(x - x_1)\cdots(x - x_n)$.

Observe que $P_n(t)$ es de grado a lo sumo n, así que $\frac{d^{n+1}}{dt^{n+1}} P_n(t) = 0$. Por otra parte $Q(t) = \prod_{i=0}^{n}(t - x_i) = x^{n+1} + b_1 x^{n-1} + ...$, es un polinomio mónico de grado $n+1$, así que $\frac{d^{n+1}}{dt^{n+1}} Q(t) = (n+1)!$

EJEMPLO 1.6 Sea $f(x) = \frac{x^8}{84} - \frac{3\cos(2x)}{8}$.

Considere el conjunto de nodos igualmente espaciados $\{(x_i, f(x_i))\}_{i=0,1,2,3,4}$ con $x_i = i \cdot 0.2$.

Una estimación del error cometido al aproximar $f(0.65)$ con $P_4(0.65)$ es

$$|f(x) - P_4(x)| \leq \frac{M_5}{5!} |(x - x_0)(x - x_1)(x - x_2)(x - x_2)(x - x_4)|$$

donde M_5 es el máximo absoluto de $f^{(5)}$ en $I = [0, 0.8]$ y $x = 0.65$.

La función $f^{(5)}(x) = 80x^3 + 12\,\mathrm{sen}(2x)$ es creciente en I (pues $f^{(6)}(x) = 240x^2 + 24\cos(2x) \geq 0$ en I), entonces $M_5 = f^5(0.8)$ con lo cual, el error estimado es ≈ 0.00024202.

1.3 Nodos de TChebyshev.

Los polinomios de TChebyshev[3] (de primera especie) se definen, de manera recursiva, de la siguiente manera:

$$\begin{cases} T_0(x) = 1, \ T_1(x) = x \\ T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \ n \geq 1. \end{cases}$$

Así, $T_{n+1}(x)$ tiene coeficiente principal 2^n. Sea $\overline{T}_{n+1}(x) = 2^{-n}T_{n+1}(x)$.

[3]

Pafnuti Lvóvich Chebyshov (1821 - 1894). El más prominente miembro de la escuela de matemáticas de St. Petersburg. Hizo investigaciones en teoría de la aproximación de funciones, teoría de los números, teoría de probabilidades y teoría de integración. Sin embargo escribió acerca de muchos otros temas: formas cuadráticas, construcción de mapas, cálculo geométrico de volúmenes, etc.

Si $x \in [-1,1]$ y $x = \cos\theta$, $T_{n+1}(\cos\theta) = \cos n\theta$. De aquí se puede probar que

$$\overline{T}_{n+1}(x) = (x - r_0)(x - r_1)\cdots(x - r_n) \quad \text{con} \quad r_i = \cos\left(\frac{2i+1}{2n+2}\,\pi\right)$$

A los números $r_i = \cos\left(\frac{2i+1}{2n+2}\,\pi\right)$ les llamamos *nodos de TChebyshev*. Pueden ser definidos a un intervalo $[a,b]$ por medio de un cambio lineal de variable que transforme $[-1,1]$ en $[a,b]$.

El resultado principal es este: Es conocido que si M_{n+1} es el máximo absoluto de de f^{n+1} en [a,b] entonces

$$\begin{aligned} f(x) - P_n(x) &= \frac{f^{n+1}(\xi(x))}{(n+1)!}(x - x_0)(x - x_1)\cdots(x - x_n) \\ |f(x) - P_n(x)| &\leq \frac{M_{n+1}}{(n+1)!}|(x - x_0)(x - x_1)\cdots(x - x_n)| \end{aligned}$$

La cota del error se puede minimizar con $\overline{T}_{n+1}$: Si $P_n(x)$ es el polinomio que interpola a f en los nodos de TChebyshev $r_i = \cos\pi(2i+1)/(2n+2)$, $i = 0, 1, ..., n$,

$$|f(x) - P_n(x)| \leq \frac{M_{n+1}}{(n+1)!}|\overline{T}_{n+1}(x)| \leq \frac{M_{n+1}}{(n+1)!}|\prod_{i=0}^{n}(x - x_i)|, \quad x \in [-1,1]$$

En particular, como en $[-1,1]$ máximo absoluto de $\overline{T}_{n+1}$ es 2^{-n}, tenemos

$$|f(x) - P_n(x)| \leq \frac{M_{n+1}}{(n+1)!2^n}, \quad x \in [-1,1]$$

EJEMPLO 1.7 Consideremos la función $f(x) = |x| + 0.5x - x^2$. En las figuras (1.6), (1.7) se muestra la representación de f contra el polinomio interpolante con $n = 15$ nodos. En el primer caso, el polinomio interpolante se construyó con nodos de TChebyshev. En el segundo caso, el polinomio interpolante se construyó con nodos igualmente espaciados.

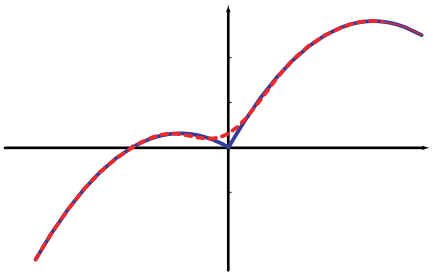

Figura 1.6 Polinomio interpolante con nodos de TChebyshev

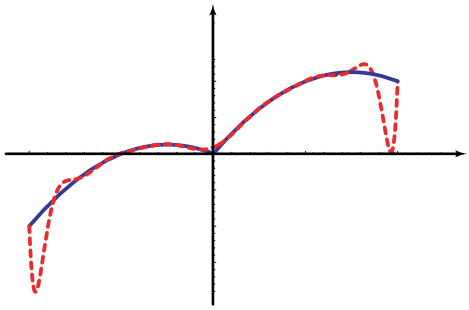

Figura 1.7 Polinomio interpolante con nodos igualmente espaciados.

1.4 Forma de Lagrange modificada y forma baricéntrica de Lagrange.

La forma de Lagrange del polinomio interpolante es atractiva para propósitos teóricos. Sin embargo se puede reescribir en una forma que se vuelva eficiente para el cálculo computacional. El resultado principal es analizar las ventajas que tiene reescribir la forma de Lagrange del polinomio interpolante en las formas

Forma Modificada.

$$P_n(x) = \ell(x) \sum_{j=0}^{n} \frac{\omega_j^{(n)}}{x - x_j} y_j \tag{1.6}$$

Forma Baricéntrica.

$$P_n(x) \begin{cases} = & y_i & \text{si} \quad x = x_i, \\ = & \dfrac{\displaystyle\sum_{k=0}^{n} \frac{\omega_k^{(n)}}{x - x_k} y_k}{\displaystyle\sum_{k=0}^{n} \frac{\omega_k^{(n)}}{x - x_k}}, & \text{si} \quad x \neq x_i \end{cases}$$

$\ell(x)$ y $\omega_k^{(n)}$ se definen así: Supongamos que tenemos $n+1$ nodos distintos $x_0, x_1, ..., x_n$. Sea $\ell(x) = \prod_{i=0}^{n} (x - x_i)$ y definimos los *pesos baricéntricos* como

$$\omega_k = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{1}{x_k - x_i}, \quad k = 0, 1, ..., n.$$

Es decir, $\ell(x) = (x - x_0)(x - x_1) \cdots (x - x_n)$ y

$$\omega_k = \frac{1}{x_k - x_0} \cdot \frac{1}{x_k - x_1} \cdots \frac{1}{x_k - x_{k-1}} \cdot \frac{1}{x_k - x_{k+1}} \cdots \frac{1}{x_k - x_n}.$$

La forma modificada se obtiene notando que $L_{n,k}(x) = \dfrac{\omega_k^{(n)}}{x - x_k} \prod_{j=0}^{n} (x - x_j) = \dfrac{\omega_k^{(n)}}{x - x_k} \ell(x)$.

La forma baricéntrica se obtiene dividiendo el polinomio interpolante por $\sum_{k=0}^{n} L_{n,k}(x) \equiv 1$ (es decir, el polinomio interpolante de $f(x) \equiv 1$ es $P_n(x) \equiv 1$), en efecto

$$P_n(x) = \sum_{k=0}^{n} y_k\, L_{n,k}(x) = \frac{\displaystyle\sum_{k=0}^{n} y_k\, L_{n,k}(x)}{\displaystyle\sum_{k=0}^{n} L_{n,k}(x)}$$

$$= \frac{\displaystyle\sum_{k=0}^{n} y_k \frac{\omega_k^{(n)}}{x-x_k} \prod_{j=0}^{n}(x-x_j)}{\displaystyle\sum_{k=0}^{n} \frac{\omega_k^{(n)}}{x-x_k} \prod_{j=0}^{n}(x-x_j)} = \frac{\displaystyle\sum_{k=0}^{n} \frac{\omega_k^{(n)}}{x-x_k} y_k}{\displaystyle\sum_{k=0}^{n} \frac{\omega_k^{(n)}}{x-x_k}} \quad \text{si} \quad x \neq x_k.$$

Esta forma se llama "forma baricéntrica" de Lagrange pues, aunque no todos los "pesos" son necesariamente positivos, expresa este polinomio como un promedio ponderado de los y_k's.

EJEMPLO 1.8 Consideremos la siguiente tabla de datos,

x	$f(x)$
0.2	3.2
0.3	3.3
0.4	3.4
0.5	4.5

Calcule la forma modificada y la forma baricéntrica de Lagrange e interpole con ambos polinomios, $f(0.35)$.

Solución: Primero calculamos $\ell(x) = (x-0.2)(x-0.3)(x-0.4)(x-0.5)$. Ahora, los pesos baricéntricos,

$$\omega_0 = \frac{1}{0.2-0.3} \cdot \frac{1}{0.2-0.4} \cdot \frac{1}{0.2-0.5} = -166.667,$$
$$\omega_1 = \frac{1}{0.3-0.2} \cdot \frac{1}{0.3-0.4} \cdot \frac{1}{0.3-0.5} = 500,$$
$$\omega_2 = \frac{1}{0.4-0.2} \cdot \frac{1}{0.4-0.3} \cdot \frac{1}{0.4-0.5} = -500,$$
$$\omega_3 = \frac{1}{0.5-0.2} \cdot \frac{1}{0.5-0.3} \cdot \frac{1}{0.5-0.4} = 166.667$$

Entonces, la forma modificada de Lagrange es,

$$P_3(x) = (x-0.2)(x-0.3)(x-0.4)(x-0.5)\left(-\frac{533.333}{x-0.2}+\frac{1650.}{x-0.3}-\frac{1700.}{x-0.4}+\frac{750.}{x-0.5}\right),$$

y la forma baricéntrica es,

$$P_3(x) = \frac{-\dfrac{533.333}{x-0.2}+\dfrac{1650.}{x-0.3}-\dfrac{1700.}{x-0.4}+\dfrac{750.}{x-0.5}}{-\dfrac{166.667}{x-0.2}+\dfrac{500.}{x-0.3}-\dfrac{500.}{x-0.4}+\dfrac{166.667}{x-0.5}}$$

En ambos casos, $f(0.35) \approx P_3(0.35) = 3.2875$.

Nodos de TChebyshev.

En el caso de que podamos escoger los nodos, la elección son los nodos de TChebyshev. En este caso el cálculo de los pesos baricéntricos es muy sencillo ([2], p.249),

$$\omega_k^{(n)} = (-1)^k \operatorname{sen} \frac{(2k+1)\pi}{2n+2}$$

Este último resultado se obtiene del siguiente cálculo:

$$\begin{aligned} \omega_k^{-1} &= \lim_{x\to x_k} \frac{\ell(x)}{(x-x_k)} \\ &= \lim_{x\to x_k} \frac{\ell(x)-\ell(k)}{(x-x_k)}, \quad \text{pues} \quad \ell(x_k)=0; \\ &= \ell'(x_k). \end{aligned}$$

1.5 Algoritmos

1.5.1 Forma Modificada y Forma Baricéntrica de Lagrange.

La implementación se centra en el cálculo de los $\omega_k^{(n)}$. Una vez calculados estos números, armar cada polinomio interpolante es algo directo.

- En el algoritmo usamos $w_0^{(0)} = 1$ y $w_k^{(j)} = (x_k - x_j)w_k^{(j-1)}$. Así $\omega_k^{(n)} = 1/w_k^{(n)}$.
- Si usamos nodos de TChebyshev, el cálculo es directo: $\omega_k^{(n)} = (-1)^k \operatorname{sen} \dfrac{(2k+1)\pi}{2n+2}$.

Algoritmo 1.1: Pesos Baricéntricos

Entrada: $n+1$ nodos distintos $\{x_i\}_{i=0,1,...,n}$.

Resultado: Pesos baricéntricos $\omega_k^{(n)}$, $k = 0, 1, ..., n$

if $\{x_i\}_{i=0,...,n}$ *son nodos de TChebyshev* **then**

  $w_k^{(n)} = (-1)^k \operatorname{sen} \dfrac{(2k+1)\pi}{2n+2}$, $k = 0, ..., n$

else

  $w_0^{(0)} = 1$;

  for $j = 1$ **to** n **do**

    for $k = 0$ **to** $j-1$ **do**

      $w_k^{(j)} = (x_k - x_j)w_k^{(j-1)}$

    $w_j^{(j)} = \prod_{k=0}^{j-1}(x_j - x_k)$;

  for $k = 0$ **to** n **do**

    $w_k^{(n)} = 1/w_k^{(n)}$

return $w_0^{(n)}, w_1^{(n)}, ..., w_n^{(n)}$

El código VBA para Excel para calcular los pesos baricéntricos (caso general)

```
'Recibe XY={(x0,y0),(x1,y1),...,(xn,yn)}, n+1 puntos
'Devuelve w = w_0^(n), w_1^(n),...w_n^(n)
'Uso: Dim Ptos()
```

```
'         Ptos = PesosBaricentricos(XY)

Function PesosBaricentricos(XY)
Dim n, nf
Dim pesos()
Dim w() As Double  'w = w_0^(n), w_1^(n),...w_n^(n)

nf = UBound(XY, 1) '# de datos en matriz XY. Suponemos que
n = nf - 1
ReDim w(nf, nf)  'XY={(x0,y0),(x1,y1),...,(xn,yn)}, n+1 puntos
                   'xi = XY(i,0), yi = XY(i,1)
ReDim pesos(n)
'Pesos Baricentricos
w(0, 0) = 1
For j = 1 To n
    For k = 0 To j - 1
        w(k, j) = (XY(k, 0) - XY(j, 0)) * w(k, j - 1)
    Next k
    prod = 1
    For k = 0 To j - 1
        prod = prod * (XY(j, 0) - XY(k, 0))
    Next k
    w(j, j) = prod
Next j
For k = 0 To n
        pesos(k) = 1 / w(k, n)
Next k
PesosBaricentricos = pesos
End Function
```

En *Mathematica* podemos escribir un código más directo: El módulo BLagrange calcula los pesos y la forma modificada de Lagrange del polinomio interpolante (el notebook de *Mathematica* se puede descargar en

http://www.cidse.itcr.ac.cr/revistamate/HERRAmInternet/v9n1_2008/Lagrange_Vs_Newton.html

```
BLagrange[XY_] :=
  Module[{j, k, n, X, Y},
   (*Xi= xi*)
   X_k  := Transpose[XY][[1,k+1]];
   Y_k  := Transpose[XY][[2,k+1]];
   n = Length[XY] - 1;
   L[x_] := ∏_{j=0}^{n} (x - X_j);
   W[j_] := (∏_{k=0}^{j-1} 1/(X_j - X_k)) (∏_{k=j+1}^{n} 1/(X_j - X_k));
   Return[L[x] * ∑_{j=0}^{n} W[j]/(x - X_j) * Y_j // N]];
```

1.5.2 Forma de Newton del polinomio interpolante.

Para la implementación de la fórmula de diferencias divididas de Newton escribimos $P_n(x) = \sum_{i=0}^{n} F_{i,i} \prod_{j=0}^{i-1}(x - x_j)$. Para el cálculo de los $F_{i,i}$'s usamos la fórmula recursiva:

$$\begin{aligned} F_{i,0} &= y_i, \quad i = 0, 2, ..., n \\ F_{i,j} &= \frac{F_{i,j-1} - F_{i-1,j-1}}{x_i - x_{i-j}}, \; i = 1, 2, ..., n; \quad j = 1, 2, ...i \end{aligned}$$

Algoritmo 1.2: Diferencias Divididas de Newton

Entrada: $\{(x_i, y_i)\}_{i=0,1,...,n}$ con los x_i's distintos.
Resultado: Coeficientes del polinomio interpolante: $F_{0,0}$, $F_{1,1}$, ..., $F_{n,n}$

for $i = 1$ **to** n **do**
    $F_{i,0} = y_i$
;
for $i = 1$ **to** n **do**
    for $j = 1$ **to** i **do**
        $F_{i,j} = \dfrac{F_{i,j-1} - F_{i-1,j-1}}{x_i - x_{i-j}}$
return $F_{0,0}$, $F_{1,1}$, ..., $F_{n,n}$

El código VBA para Excel para calcular las diferencias divididas es

```
' Recibe un array XY y devuelve una matriz M
' xi=XY(i,0) i=0,1,...,nf-1
' yi=XY(i,1) i=0,1,...,nf-1

Function MatrizDifDivNewton(XY() As Double)
Dim M() 'Matriz de diferencias divididas
Dim nf, n

nf = UBound(XY, 1)      ' Número de filas
ReDim M(nf + 1, nf + 1)
n = nf - 1              'XY tiene n+1 puntos

For i = 0 To n          'Columna 0 de M con las yi's
    M(i, 0) = XY(i, 1)  'xi = XY(i,0), yi = XY(i,1)
Next i
                        'Fij=(F_i,j-1 - F_i-1,j-1)/(x_i-x_i-j)
For i = 1 To n
    For j = 1 To i
        M(i,j)=(M(i,j-1)-M(i-1,j-1))/(XY(i,0)-XY(i-j,0))
    Next j
Next i
MatrizDifDivNewton = M
End Function
```

EJEMPLO 1.9

Para imprimir la matriz de diferencias divididas en una hoja Excel, en la celda `cells(fila,col)` se puede usar la siguiente función

```
Sub imprimirMatrizDDN(M, fila, col)
Dim n

n = UBound(M, 1)
Cells(fila, col) ="Matriz de diferencias divididas"+str(n)

For j = 0 To n
 For i = j To n
    Cells(fila + 1 + i, col + j) = M(i, j)
  Next i
Next j
End Sub
```

La construcción del polinomio se podría hacer con algo como

```
Pn = Str(y0)
Qn = " "
For i = 1 To n
    For j = 0 To i-1
        Qn = Qn + "*(x -" + str(xj) + ")"
    Next j
    ai = F(i, i)
    Pn = Pn + "+" + str(ai) + Qn
    Qn = " "
Next i
Pn = Replace(Pn, "--", " + ")
Pn = Replace(Pn, "+-", " - ")
```

En *Mathematica* podemos escribir un código más directo: El módulo DDNewton calcula la forma de Newton del polinomio interpolante (el notebook de *Mathematica* se puede bajar en la revista digital Matemática, Educación e Internet).

```
DDNewton[XY_] :=
  Module[{F, j, i, n, X, Y},
   X = Transpose[XY][[1]];
   Y = Transpose[XY][[2]];
   n = Length[XY] - 1;
   F = Table["", {n + 1}, {n + 1}];
   F[[All, 1]] = Y[[All]];
   For[j = 1, j ≤ n, j++,
    For[i = j, i ≤ n, i++,
     F[[i+1, j+1]] = N[(F[[i+1, j]] - F[[i, j]])/(X[[i+1]] - X[[i+1-j]])]]];
   For[i = 0, i ≤ n, i++,
    p[i + 1, x_] = N[Product[(x - X[[j]]), {j, 1, i}]]];
   Return[Sum[F[[i+1, i+1]] p[i + 1, x], {i, 0, n}] // N];
   ];
```

1.6 Comparación: Forma de Lagrange vs Diferencias Divididas de Newton.

En la forma modificada de Lagrange se deben calcular de los pesos baricéntricos $\omega_k^{(n)}$. Esto se hace, en general, con $O(n^2)$ operaciones[4]. En la forma de Newton, el cálculo de las diferencias divididas requiere $n(n+1)$ sumas y $n(n+1)/2$ divisiones. Así que el esfuerzo computacional en ambos métodos es esencialmente el mismo.

En la forma modificada de Lagrange, agregar un punto (x_n, x_{n+1}) requiere ampliar el ciclo *for* de n a $n+1$. Para calcular P_{n+1} solo necesita cambiar $n+1$ por

[4]Algo es $O(n^2)$ si es $\leq cn^2$ (c constante) a partir de algún momento

n en (1.6). En la forma de Newton, agregar un punto (x_n, x_{n+1}) requiere ampliar el ciclo *for* de n a $n+1$. Para calcular P_{n+1} se debe agregar el nuevo término $a_{n+1}(x-x_0)\cdots(x-x_n)$.

En la forma de Lagrange, los pesos baricéntricos no dependen de las cantidades y_i, así que una vez calculados estos pesos para un conjunto de nodos $\{x_i\}_{i=0,\ldots,n}$, se puede interpolar cualquier función en $O(n)$ pasos (lo que se gasta en calcular P_n). La forma de Newton requiere el cálculo de la tabla de diferencias divididas para cada nueva función.

La forma modificada de Lagrange no depende del orden de los nodos. Las diferencias divididas si tienen esa dependencia: para valores grandes de n, muchos ordenamientos provocan inestabilidad. Para lograr estabilidad se se requiere usar nodos tales como los generados por el algoritmo de Leja o el de van der Corput (ver [10]). En la sección (1.8) se muestra el efecto de usar estos tipos de nodos.

En la forma modificada de Lagrange, pareciera que podría haber problemas si evaluamos $x \approx x_k$. En este caso $\omega_k^{(n)}/(x-x_k)$ sería grande con el consiguiente riego de inexactitud en el cálculo. Sin embargo, como ambas cantidades aparecen en el numerador y el denominador, hay una cancelación que mantiene la exactitud ([11]).

Aunque los pesos baricéntricos sean calculados con mucho error (P_n dejaría de ser un polinomio) aún así, siguen interpolando los datos: se convierte en un interpolador racional.

Diferencias divididas tiene aplicación en otros temas el análisis numérico: interpolación de Hermite, interpolación matricial, ecuaciones diferenciales, etc. Pero cuando se trata de interpolación polinomial, hay mucha evidencia a favor de la forma modificada de Lagrange.

1.7 Análisis de error.

Informalmente, un cálculo numérico es *inestable* si pequeños errores en algún estado de los cálculos es magnificado en los pasos siguientes de tal manera que afecta

seriamente la exactitud de todo el cálculo.

Por otro lado, se dice que un problema está *mal condicionado* si pequeños cambios en los datos de entrada producen grandes cambios en la respuesta. Para ciertos tipos de problema se puede definir un *número de condición*. Si este número es grande, indica que el problema es mal condicionado.

Sea $||f||_\infty = \max\limits_{x\in[a,b]} |f(x)|$. Si tenemos un conjunto de nodos $X = \{x_j\}_{j-0,\ldots,n}$, la constante de Lebesgue se define como

$$\Lambda_n(X) = \left|\left|\sum_{j=0}^{n} |L_{n,k}(x)|\right|\right|_\infty$$

Sea $\{\widetilde{f}(x_i)\}_{i=0,1,\ldots,n}$ es una perturbación de $\{f(x_i)\}_{i=0,1,\ldots,n}$, $x_i \in [a,b]$. La perturbación puede ser una consecuencia del error de redondeo, por ejemplo. Sean $\widetilde{P}_n$ el polinomio interpolante correspondiente al conjunto $\{(x_i, \widetilde{f}(x_i))\}$ y P_n el polinomio interpolante de f en $\{(x_i, f(x_i))\}$. Entonces,

$$||P_n(x) \ - \ P)n(x)||_\infty = \max_{x\in\,[a,b]} \left|\sum_{j=0}^{n}[(f(x_j) - \widetilde{f}(x_j))L_{n,j}(x)]\right| \ \leq \ \Lambda_n(X) \max_{i=0,\ldots,n} |f(x_i) - \widetilde{f}(x_i)|$$

Esto dice que pequeños cambios en los datos implica pequeños cambios en el polinomio de interpolación si la constante de Lebesgue es pequeña[5]. Esta constante juega el papel de *número de condición* en los problemas de interpolación.

En el caso de nodos igualmente espaciados, se puede probar que $\Lambda_n(X) \simeq \dfrac{2^{n+1}}{e\ n\ \log n}$. Esto indica que, para n grande, esta forma de interpolación se vuelve inestable.

En el caso de la forma baricéntrica, se puede probar que

[5]El error en la construcción de P_n no se toma en cuenta pues es despreciable ([**?**])

$$\Lambda_n(X) \geq \frac{1}{2n^2} \frac{\text{máx}_{j-0,\ldots n} |\omega_k^{(\text{n})}|}{\text{mín}_{j-0,\ldots n} |\omega_k^{(\text{n})}|}$$

Esto dice que si los pesos baricéntricos se vuelven grandes, el problema de interpolación respectivo se vuelve un problema mal condicionado.

En el caso de la forma modificada de Lagrange, el número de condición se define como ([11])

$$Cond(x, n, f) = \lim_{\epsilon \to 0} \sup \left\{ \left| \frac{P_f(x) - P_{f+\Delta f}(x)}{\epsilon \, P_f(x)} \right| \quad \text{tal que} \quad |\Delta f| \leq \epsilon |f| \right\}$$

Aquí, P_g es el polinomio interpolante de g en el conjunto de nodos.

En el caso de la forma baricéntrica de Lagrange, el número de condición se define como ([11])

$$Cond(x, n, f) = \frac{\sum_{j=0}^{n} \left| \frac{\omega_j \, y_j}{x - x_j} \right|}{\left| \sum_{j=0}^{n} \frac{\omega_j \, y_j}{x - x_j} \right|}$$

Si la salida de una algoritmo, que calcula el valor de una función f, se denota $Pro(z)$ y si $Pro(z) = f(z + \delta)$ donde δ es pequeño, decimos que $Pro()$ es *estable hacia atrás* en el sentido de que $Pro(z)$ es el valor exacto de f en una entrada ligeramente perturbada $z + \delta$.

El error *hacia adelante* de un algoritmo es la diferencia entre el resultado y la solución.

El error hacia adelante es a lo sumo tan grande como el número de condición multiplicado por el error hacia atrás.

El análisis de error en ([11]) concluye que la forma modificada de Lagrange es estable hacia atrás respecto a las perturbaciones en los valores de la función. La fórmula baricéntrica no es estable hacia atrás pero es estable hacia adelante para cualquier conjunto de puntos con una constante de Lebesgue pequeña.

En la misma referencia se observa que no hay un análisis de error conocido en el caso de diferencias divididas pero que sí se ha observado de manera teórica y experimental que los errores en este método son muy dependientes del orden de los nodos y son inaceptablemente grandes en el caso de los nodos de TChebyshev.

Alta exactitud relativa.

Decimos que E puede ser obtenida con *alta exactitud relativa* si el error relativo del valor calculado $\widehat{E}$ puede ser acotado como sigue:

$$\frac{||\widehat{E} - E||}{||E||} \leq C \cdot eps$$

C es una constante y eps es la unidad de redondeo ($eps \approx 10^{-16}$ en los computadores actuales).

Por ejemplo, la representación $\widehat{x}$ de $x \in \mathbb{R}$ en el computador tiene alta exactitud relativa pues $\dfrac{||\widehat{x} - x||}{||x||} \leq eps$

Se puede calcular con alta exactitud relativa productos, cocientes y adiciones (de sumandos con el mismo signo) de expresiones que puedan ser calculadas con alta exactitud relativa. Las restas de datos exactos se calculan con alta exactitud relativa. Las restas de números contaminados con algún error y con distinto signo no tienen alta exactitud relativa.

Durante el cálculo de las diferencias divididas, se deben calcular restas de valores previamente calculados y entonces se pierde la alta exactitud relativa. Las diferencias de desempeño entre la forma de Newton y la forma de Lagrange se explica por la ausencia de exactitud en los cálculo de los coeficientes.

La forma de Lagrange mantiene todas los cálculos con alta exactitud relativa hasta que se ejecuta la suma final, donde ésta se puede perder.

1.8 Experimentos.

En los experimentos que siguen, se usa interpolación polinomial de una función f en nodos de TChebyshev y en nodos de Leja. Se usan las forma modificada de Lagrange, la forma baricéntrica y diferencias divididas de Newton. Se busca mostrar el fenómeno de inestabilidad de diferencias divididas y como mejora sobre nodos de Leja. También se recomiendan nodos de van der Corpus (estos nodos son números en $[0,1]$). En ([10]) aparece el código (en MatLab) para calcular nodos de Leja. En el notebook de *Mathematica 5.0,* con el cual fueron generados los ejemplos, aparece el código para calcular nodos de van der Corput (la revista digital Matemática, Educación e Internet).

En la mayoría de figuras se muestran las gráficas con el error relativo. Los polinomios interpolantes son PN para la forma de Newton, PML para la forma modificada de Lagrange y PBL para forma baricéntrica.

Para el error relativo se usa EPN cuando es la forma de Newton contra f, $EPML$ cuando es la forma modificada de Lagrange contra f y $EPBL$ cuando es la forma Baricéntrica contra f.

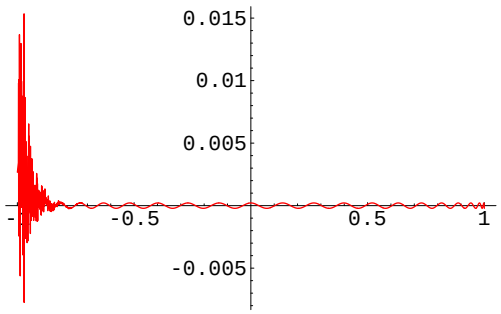

Figura 1.8 $f(x) = 1/(1+25x^2)$ y $PN(x)$. Error relativo sobre 45 nodos de TChebyshev.

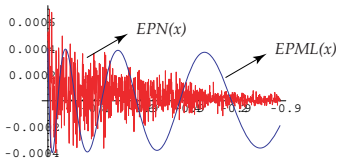

Figura 1.9 $f(x) = 1/(1+25x^2)$ vs $PN(x)$ y $PML(x)$. Error relativo sobre 50 nodos de TChebyshev

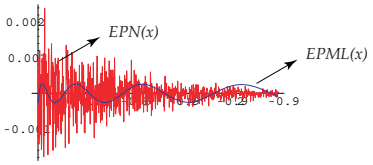

Figura 1.10 $f(x) = 1/(1+25x^2)$ vs $PN(x)$ y $PML(x)$. Error relativo sobre 52 nodos de TChebyshev

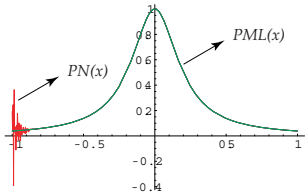

Figura 1.11 $f(x) = 1/(1+25x^2)$ vs $PN(x)$ y $PML(x)$. PN y PML interpolan f sobre 52 nodos de TChebyshev. Note la inestabilidad de PN cerca de -1.

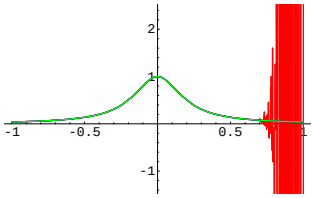

Figura 1.12 $f(x) = 1/(1+25x^2)$ vs $PN(x)$. PN interpola f sobre 70 nodos de TChebyshev, en orden descendente.

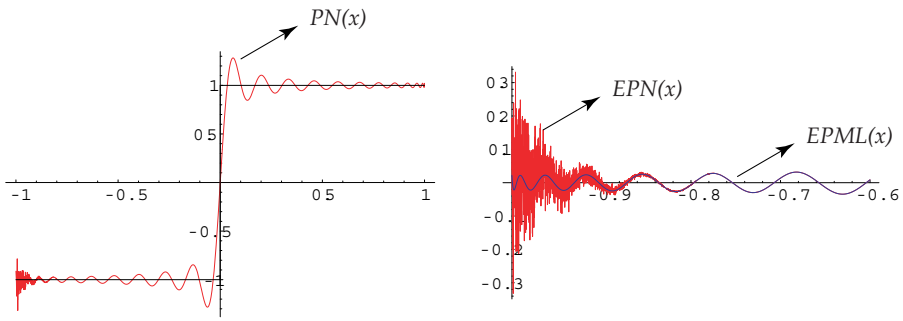

Figura 1.13 $f(x) = e^x$ vs $PN(x)$ y $PML(x)$. PN y PML interpolan f sobre 65 nodos de TChebyshev. A la derecha, gráficas con error relativo.

1.8.1 Sucesiones de Leja.

Para diferencias divididas se recomienda nodos de Leja en vez de nodos de TChebyshev. El interés en las sucesiones de Leja $z_0, z_1, \ldots$ se deriva del hecho de que el polinomio $\prod_{k=0}^{j-1}(z - z_k)$ puede ser evaluado con gran exactitud aún cuando el grado sea alto. Generar la sucesión es costosa computacionalmente.

En el caso más simple, los puntos se calculan así: Sea K un conjunto compacto. $z_0 \in K$ es un punto que satisface $|z_0| = \max_{z \in K}|z|$. Luego, para $j = 1, 2, \ldots$ los puntos z_j satisfacen

$$\prod_{k=0}^{j-1} |z_j - z_k| = \max_{z \in K} \prod_{k=0}^{j-1} |z - z_k|, \; z_j \in K.$$

Ver ([10]).

Bibliografía

[1] W. Gautschi. *Numerical Analysis. An Introduction.* Birkhäuser, 1997.

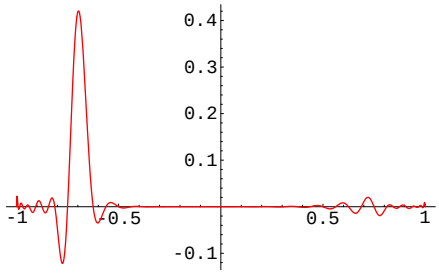

Figura 1.14 $f(x) = 1/(1+25x^2)$ y $PN(x)$. Error relativo sobre 45 nodos de Leja

[2] P. Henrici.*Essentials of Numerical Analysis.* Wiley, New York, 1982.

[3] J. Stoer, R. Bulirsch. *Introduction to Numerical Analysis.* 3rd ed. Springer, 2002.

[4] G. Dahlquist, A. Björk. *Numerical Mathematics in Scientific Computation.*

[5] D. Kahaner, K. Moler, S. Nash. *Numerical Methods and Software.* Prentice Hall.1989.

[6] D. Kincaid, W. Cheney. *Numerical Analysis. Mathematics of Scientific Computing.* Brooks-Cole Publishing Co.1991.

[7] R. Burden, J. Faires. *Análisis Numérico.* 6ta ed. Thomson. 1998.

[8] E. Cheney, *Introduction to Approximation Theory.* Internat. Ser. Pure ans Applied Mathematics. McGraw-Hill. 1966.

[9] J.P. Berrut, L. N. Trefethen. "Barycentric Lagrange Interpolation" Siam Rewiew. Vol. 46, No. 3. 2004.

[10] J. Baglama, D. Calvetti, L. Reichel "Fast Leja Points". Electronic Transactions on Numerical Analysis. Volume 7, pp. 124-140.1998.

[11] J. Higham, "The numerical stability of barycentric Lagrange interpolation". IMA Journal of Numerical Analysis 24. 2004.