18.1.5 Algoritmo computacional para el polinomio de interpolación de Newton

Tres propiedades hacen a los polinomios de interpolación de Newton muy atractivos para aplicaciones en computadora:

1. Como en la ecuación (18.7), es posible desarrollar de manera secuencial versiones de grado superior con la adición de un solo término a la ecuación de grado inferior. Esto facilita la evaluación de algunas versiones de diferente grado en el mismo programa. En especial tal capacidad es valiosa cuando el grado del polinomio no se conoce *a priori.* Al agregar nuevos términos en forma secuencial, podemos determinar cuándo se alcanza un punto de regreso disminuido (es decir, cuando la adición de términos de grado superior ya no mejora de manera significativa la estimación, o en ciertas situaciones incluso la aleja). Las ecuaciones para estimar el error, que se analizan en el punto 3, resultan útiles para visualizar un criterio objetivo para identificar este punto de términos disminuidos.
2. Las diferencias divididas finitas que constituyen los coeficientes del polinomio [ecuaciones (18.8) hasta (18.11)] se pueden calcular eficientemente. Es decir, como en la ecuación (18.14) y la figura 18.5, las diferencias de grado inferior sirven para calcular las diferencias de grado mayor. Utilizando esta información previamente determinada, los coeficientes se calculan de manera eficiente. El algoritmo en la figura 18.7 incluye un esquema así.
3. El error estimado [ecuación (18.18)] se incorpora con facilidad en un algoritmo computacional debido a la manera secuencial en la cual se construye la predicción.

```
SUBROUTINE NewtInt (x, y, n, xi, yint, ea)
  LOCAL fdd(n,n)
  DOFOR i = 0, n
    fdd(i,0) = y(i)
  END DO
  DOFOR j = 1, n
    DOFOR i = 0, n - j
      fdd(i,j) = (fdd(i+1,j-1) - fdd(i,j-1))/(x(i+j) - x(i))
    END DO
  END DO
  xterm = 1
  yint(0) = fdd(0,0)
  DOFOR order = 1, n
    xterm = xterm * (xi - x(order-1))
    yint2 = yint(order-1) + fdd(0,order) * xterm
    ea(order-1) = yint2 - yint(order-1)
    yint(order) = yint2
  END order
END NewtInt
```

Figura 18.7
Un algoritmo para el polinomio de interpolación de Newton escrito en pseudocódigo.

Todas las características anteriores pueden aprovecharse e incorporarse en un algoritmo general para implementar el polinomio de Newton (figura 18.7). Observe que el algoritmo consiste de dos partes: la primera determina los coeficientes a partir de la ecuación (18.7); la segunda establece las predicciones y sus errores correspondientes. La utilidad de dicho algoritmo se demuestra en el siguiente ejemplo.

EJEMPLO 18.5 Estimaciones del error para determinar el grado de interpolación adecuado

Planteamiento del problema Después de incorporar el error [ecuación (18.18)], utilice el algoritmo computacional que se muestra en la figura 18.7 y la información tabulada siguiente para evaluar $f(x) = \ln x$ en $x = 2$:

x	$f(x) = \ln x$
1	0
4	1.3862944
6	1.7917595
5	1.6094379
3	1.0986123
1.5	0.4054641
2.5	0.9162907
3.5	1.2527630

Solución Los resultados de emplear el algoritmo de la figura 18.7 para obtener una solución se muestran en la figura 18.8. El error estimado, junto con el error verdadero (basándose en el hecho de que $\ln 2 = 0.6931472$), se ilustran en la figura 18.9. Observe que el error estimado y el error verdadero son similares y que su concordancia mejora conforme aumenta el grado. A partir de estos resultados se concluye que la versión de quinto grado da una buena estimación y que los términos de grado superior no mejoran significativamente la predicción.

Este ejercicio también ilustra la importancia de la posición y el orden de los puntos. Por ejemplo, hasta la estimación de tercer grado, la mejoría es lenta debido a que los puntos que se agregaron (en $x = 4$, 6 y 5) están distantes y a un lado del punto de análisis en $x = 2$. La estimación de cuarto grado muestra una mejoría un poco mayor, ya

Observe que, como en el método de Newton, la forma de Lagrange tiene un error estimado de [ecuación (18.17)]

$$R_n = f[x, x_n, x_{n-1}, \ldots, x_0] \prod_{i=0}^{n} (x - x_i)$$

De este modo, si se tiene un punto adicional en $x = x_{n+1}$, se puede obtener un error estimado. Sin embargo, como no se emplean las diferencias divididas finitas como parte del algoritmo de Lagrange, esto se hace rara vez.

Las ecuaciones (18.20) y (18.21) se programan de manera muy simple para implementarse en una computadora. La figura 18.11 muestra el pseudocódigo que sirve para tal propósito.

En resumen, en los casos donde se desconoce el grado del polinomio, el método de Newton tiene ventajas debido a la comprensión que proporciona respecto al comportamiento de las fórmulas de diferente grado. Además, el estimado del error representado por la ecuación (18.18) se agrega usualmente en el cálculo del polinomio de Newton debido a que el estimado emplea una diferencia finita (ejemplo 18.5). De esta manera, para cálculos exploratorios, a menudo se prefiere el método de Newton.

Cuadro 18.1 Obtención del polinomio de lagrange directamente a partir del polinomio de interpolación de Newton

El polinomio de interpolación de Lagrange se obtiene de manera directa a partir de la formulación del polinomio de Newton. Haremos esto únicamente en el caso del polinomio de primer grado [ecuación (18.2)]. Para obtener la forma de Lagrange, reformulamos las diferencias divididas. Por ejemplo, la primera diferencia dividida,

$$f[x_1, x_0] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \tag{C18.1.1}$$

se reformula como

$$f[x_1, x_0] = \frac{f(x_1)}{x_1 - x_0} + \frac{f(x_0)}{x_0 - x_1} \tag{C18.1.2}$$

conocida como la *forma simétrica.* Al sustituir la ecuación (C18.1.2) en la (18.2) se obtiene

$$f_1(x) = f(x_0) + \frac{x - x_0}{x_1 - x_0} f(x_1) + \frac{x - x_0}{x_0 - x_1} f(x_0)$$

Por último, al agrupar términos semejantes y simplificar se obtiene la forma del polinomio de Lagrange,

$$f_1(x) = \frac{x - x_1}{x_0 - x_1} f(x_0) + \frac{x - x_0}{x_1 - x_0} f(x_1)$$

```
FUNCTION Lagrng(x, y, n, xx)
  sum = 0
  DOFOR i = 0, n
    product = y_i
    DOFOR j = 0, n
      IF i ≠ j THEN
        product = product*(xx - x_j)/(x_i - x_j)
      ENDIF
    END DO
    sum = sum + product
  END DO
  Lagrng = sum
END Lagrng
```

Figura 18.11
Pseudocódigo para la interpolación de Lagrange. Este algoritmo se establece para calcular una sola predicción de grado *n*-ésimo, donde $n + 1$ es el número de puntos asociados con datos.

Cuando se va a ejecutar sólo una interpolación, las formulaciones de Lagrange y de Newton requieren un trabajo computacional semejante. No obstante, la versión de Lagrange es un poco más fácil de programar. Debido a que no requiere del cálculo ni del almacenaje de diferencias divididas, la forma de Lagrange a menudo se utiliza cuando el grado del polinomio se conoce *a priori*.