Algoritmos para aproximación numérica

Giovanni Ramírez García, PhD

Escuela de Ciencias Físicas y Matemáticas
Universidad de San Carlos de Guatemala

Guatemala, 15 de marzo de 2021

Características generales

REAL(8), ALLOCATABLE DIMENSION(:) datos

- Cada implementación debe leer un archivo de texto donde se encuentran los datos. Cada quien elije el nombre, sugerencia *datos*.
- La primera línea del archivo es el número de datos n. Esto permite hacer asignación dinámica de memoria.

ALLOCATE(datos(n))

- Posteriormente en el archivo estarán los puntos $(x_n, f(x_n))$. La primera columna para x_n y la segunda para $f(x_n)$.
- Para cuando sea necesario, la tercera columna del archivo tendrá la información de la primera derivada $f'(x_n)$.
- El punto que se quiere aproximar $(x, f(x))$ se debe leer de otro archivo de texto. Cada quien elije el nombre, sugerencia *punto*.
- Vamos a tomar siempre puntos ordenados tal que $x_1 < \cdots < x_n$.
- Importante: vamos modificar las definiciones de los polinomios para que los índices inicien vayan de $1 \to n$. Esto no es necesario en el polinomio de diferencias divididas centradas.

Fortran: índices que van de $1 \to n$

Polinomio de Lagrange

- Consideremos los n puntos $\{(x_1, f(x_1)), \ldots, (x_n, f(x_n))\}$.
- Para aproximar $f(x)$ es posible construir un polinomio de Lagrange de orden $n-1$ dado por

$$P_{n-1}(x) = \sum_{k=1}^{n} f(x_k) L_{n-1,k}(x), \quad \text{donde } L_{n,k}(x) = \prod_{\substack{i=1 \\ i\neq k}}^{n+1} \frac{x - x_i}{x_k - x_i},$$

- Para $n = 3$, se tienen los puntos $\{(x_1, f(x_1)), (x_2, f(x_2)), (x_3, f(x_3))\}$. Entonces $P_2(x) = f(x_1)L_{2,1}(x) + f(x_2)L_{2,2}(x) + f(x_3)L_{2,3}(x)$ con

$$L_{2,1}(x) = \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)}, \quad L_{2,2}(x) = \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)},$$

$$L_{2,3}(x) = \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)}.$$

Algoritmo de Lagrange (I)

Entrada: del archivo *datos*, el número de puntos n y los puntos $\{(x_1, f(x_1)), \ldots, (x_n, f(x_n))\}$; del archivo *punto*, el punto $(x, f(x))$. **Salida:** el valor aproximado $f(x) \approx P_{n-1}(x)$ y el error absoluto $|f(x) - P_{n-1}(x)|$. **Requiere:** función `LNK(k,x,Xn)`.

1. Iniciar.
2. Leer el número de puntos n.
3. Reservar memoria para $Xn(n) = x_n$ y $Fn(n) = f(x_n)$.
4. Leer $x_n \rightarrow Xn(n)$ y $f(x_n) \rightarrow Fn(n)$.
5. Leer $x \rightarrow x$ y $f(x) \rightarrow f$.
6. Reservar memoria para $L(n)$.
7. Hacer para $1 \leq k \leq n$:
   7.1 $L(k) = LNK(k, x, Xn)$.
8. Definir $P = 0$.
9. Hacer para $1 \leq k \leq n$:
   9.1 $P = P + Fn(k)L(k)$.
10. $AbsError = ABS(f - P)$.
11. Escribir P y $AbsErr$.
12. Liberar la memoria reservada.
13. Fin.

Programación defensiva: validar los valores de n y de x.

$n \geq 2$

$x \in [x_1, x_n]$

Algoritmo de Lagrange (II)

índice

Función `LNK(k,x,Xn)`. **Entrada:** k es el ~~orden~~ del coeficiente de Lagrange en el punto x, con el conjunto de nodos Xn. **Salida:** el k-ésimo coeficiente de Lagrange $L_{n,k}(x)$.

1. Iniciar.
2. Definir $n = SIZE(Xn, 1)$.
3. Definir $L = 1$.
4. Hacer para $1 \leq i \leq n+1$:

   4.1 Si $i = k$, continuar. Si no,
   $$L = L * \frac{x - Xn(i)}{Xn(k) - Xn(i)}.$$
5. Retornar L.
6. Fin.

Programación defensiva: validar n y que $Xn(k) - Xn(i) \neq 0$.

Programación modular: se debe implementar `LNK(Xi,x,k)` en un archivo aparte.

$n \geq 2$

Tarea: ¿Qué hace fortran con una división entre cero?

NaN

Polinomio de diferencias divididas (I)

- Consideremos los n puntos $\{(x_1, f(x_1)), \ldots, (x_n, f(x_n))\}$.
- Para aproximar $f(x)$ es posible construir un polinomio de diferencias divididas de Newton de orden $n-1$ dado por

$$P_{n-1}(x) = f[x_1] + \sum_{k=2}^{n} f[x_1, \ldots, x_k]\overbrace{(x - x_1)\cdots(x - x_{k-1})}^{\text{prod Nodos}},$$

dif. adelantadas ⇒ dif. en general

donde $f[x_i] = f(x_i)$ es la diferencia dividida de orden cero y la diferencia dividida de orden k es

$$f[x_i, x_{i+1}, \ldots, x_{i+k-1}, x_{i+k}] = \frac{f[x_{i+1}, x_{i+2}, \ldots, x_{i+k}] - f[x_i, x_{i+1}, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}.$$

Polinomio de diferencias divididas (II)

- Para $n = 3$, se tienen los puntos $\{(x_1, f(x_1)), (x_2, f(x_2)), (x_3, f(x_3))\}$. Entonces $P_2(x) = f[x_1] + f[x_1, x_2](x - x_1) + f[x_1, x_2, x_3](x - x_1)(x - x_2)$.
- Las diferencias de orden cero son $f[x_i] = f(x_i)$.
- Las diferencias divididas de orden uno son $f[x_0, x_1] = \dfrac{f[x_1] - f[x_0]}{x_1 - x_0}$, $f[x_1, x_2] = \dfrac{f[x_2] - f[x_1]}{x_2 - x_1}$, $f[x_2, x_3] = \dfrac{f[x_3] - f[x_2]}{x_3 - x_2}$.
- La diferencia de orden dos es $f[x_1, x_2, x_3] = \dfrac{f[x_2, x_3] - f[x_1, x_2]}{x_3 - x_1}$.

Tabla de diferencias divididas

- Las diferencias divididas se pueden ordenar en una tabla, por ejemplo para $n = 4$

x	$f(x)$	1a. DD	2a DD	3a DD
x_1	$f[x_1]$			
		$f[x_1, x_2]$		
x_2	$f[x_2]$		$f[x_1, x_2, x_3]$	
		$f[x_1, x_3]$		$f[x_1, x_2, x_3, x_4]$
x_3	$f[x_3]$		$f[x_2, x_3, x_4]$	
		$f[x_3, x_4]$		
x_4	$f[x_4]$			

- Sólo se pueden construir diferencias divididas hasta el orden $n - 1$.
- Hay n diferencias divididas de orden cero.
- Hay $n - 1$ diferencias divididas de orden uno.
- Y así, hasta que al final sólo hay una diferencia dividida de orden $n - 1$.

datos
$x_1 \quad f(x_1)$
$\vdots$
$x_4 \quad f(x_4)$

dif. div.
$n\left\{ f[x_1], \ldots, f[x_n] \right.$
$n-1\left\{ f[x_1,x_2], \ldots, f[x_3,x_4] \right.$
$n-2\left\{ f \ldots, f \ldots \right.$
$n-3\left\{ \ldots \right.$

Polinomio de diferencias divididas adelantadas

- Consideremos los n puntos equiespaciados $\{(x_1, f(x_1)), \ldots, (x_n, f(x_n))\}$, tal que $h = x_{i+1} - x_i$, para $i = 1, \ldots, n$.
- Para aproximar $f(x)$, donde $x = x_i + (s - i)h$, es posible construir un polinomio de diferencias divididas adelantadas de Newton de orden $n - 1$ dado por

$$P_{n-1}(x) = f[x_1] + \sum_{k=2}^{n} (s - k + 2) \cdots (s - 1)sh^{k-1} f[x_1, \ldots, x_k].$$

- Para $n = 4$ puntos equiespaciados, $h = x_{i+1} - x_i$, la aproximación para el nodo $x = x_0 + sh$ es

$$P_3(x) = f[x_1] + shf[x_1, x_2] + (s - 1)sh^2 f[x_1, x_2, x_3] + (s - 2)(s - 1)sh^3 f[x_1, x_2, x_3, x_4].$$

Polinomio de diferencias divididas atrasadas

- Consideremos los n puntos equiespaciados $\{(x_1, f(x_1)), \ldots, (x_n, f(x_n))\}$, tal que $h = x_{i+1} - x_i$, para $i = 1, \ldots, n$.
- Para aproximar $f(x)$, donde $x = x_n + sh$, es posible construir un polinomio de diferencias divididas atrasadas de Newton de orden $n - 1$ dado por

* revisar

$$P_{n-1}(x) = f[x_n] + \sum_{k=2}^{n} (s + k - 2) \cdots (s + 1) s h^{k-1} f[x_n, \ldots, x_{n-k+1}]$$

- Para $n = 4$ puntos equiespaciados, $h = x_{i+1} - x_i$, la aproximación para el nodo $x = x_3 + sh$ es

$$P_3(x) = f[x_4] + shf[x_4, x_3] + (s + 1)sh^2 f[x_4, x_3, x_2] + (s + 2)(s + 1)sh^3 f[x_4, x_3, x_2, x_1].$$

Diferencias divididas centradas (I)

- Consideremos que x_0 está cerca del centro del intervalo.
- Los nodos menores que x_0 se etiquetan de modo que $\cdots < x_{-2} < x_{-1} < x_0$.
- Los nodos mayores que x_0 se etiquetan de modo que $x_0 < x_1 < x_2 < \cdots$.
- Entonces si n es par, la fórmula de Stirling es $P_n(x) = P_{2m}(x)$, donde

$$
\begin{aligned}
P_{2m}(x) = f[x_0] + sh\left(\frac{f[x_{-1},x_0] + f[x_0,x_1]}{2}\right) + s^2h^2 f[x_{-1},x_0,x_1] + \\
s(s^2-1)h^3\left(\frac{f[x_{-2},x_{-1},x_0,x_1] + f[x_{-1},x_0,x_1,x_2]}{2}\right) + \cdots + \\
s^2(s^2-1)(s^2-4)\cdots(s^2-(m-1)^2)h^{2m} f[x_m,\ldots,x_m].
\end{aligned}
$$

Diferencias divididas centradas (II)

- En el caso de que n sea impar, la fórmula de Stirling es $P_n(x) = P_{2m+1}(x)$, donde

$$
\begin{aligned}
P_{2m+1}(x) = {} & f[x_0] + sh\left(\frac{f[x_{-1},x_0]+f[x_0,x_1]}{2}\right) + s^2h^2 f[x_{-1},x_0,x_1] + \\
& s(s^2-1)h^3\left(\frac{f[x_{-2},x_{-1},x_0,x_1]+f[x_{-1},x_0,x_1,x_2]}{2}\right) + \cdots + \\
& s^2(s^2-1)(s^2-4)\cdots(s^2-(m-1)^2)h^{2m} f[x_m,\ldots,x_m] + \\
s(s^2-1)\cdots(s^2-m^2)h^{2m+1} & \left(\frac{f[x_{-m-1},\ldots,x_m]+f[x_{-m},\ldots,x_{m+1}]}{2}\right).
\end{aligned}
$$

- Recodar que los índices en Fortran inician en 1.

Algoritmo de Diferencias divididas de Newton

Entrada: del archivo *datos*, el número de puntos n y los puntos $\{(x_1, f(x_1)), \ldots, (x_n, f(x_n))\}$; del archivo *punto*, el punto $(x, f(x))$. **Salida:** el valor aproximado $f(x) \approx P_{n-1}(x)$ y el error absoluto $|f(x) - P_{n-1}(x)|$. **Requiere:** funciones `divDiff(Xn,Fn)` y `prodNodes(x,Xn)`.

1. Iniciar.
2. Leer el número de puntos n.
3. Reservar memoria para $Xn(n) = x_n$ y $Fn(n) = f(x_n)$.
4. Leer $x_n \to Xn(n)$ y $f(x_n) \to Fn(n)$.
5. Leer $x \to x$ y $f(x) \to f$.
6. Reservar memoria para $Tabla(n-1, n-1)$.
7. $Tabla = divDiff(Xn, Fn)$
8. Reservar memoria para $N(n-1)$.
9. $N = prodNodes(x, X_n)$.
10. Definir $P = Fn(1)$.
11. Hacer para $2 \leq k \leq n$:
    11.1 $P = P + Tabla(1, k)N(k)$.
12. $AbsError = ABS(f - P)$.
13. Escribir P y $AbsErr$.
14. Liberar la memoria reservada.
15. Fin.

Algoritmo de Diferencias divididas adelantadas

Entrada: del archivo *datos*, el número de puntos n y los puntos $\{(x_1, f(x_1)), \ldots, (x_n, f(x_n))\}$; del archivo *punto*, el punto $(x, f(x))$. **Salida:** el valor aproximado $f(x) \approx P_{n-1}(x)$ y el error absoluto $|f(x) - P_{n-1}(x)|$. **Requiere:** funciones `divDiff(Xn,Fn)` y `prodFwd(s,h,n)`

1. Iniciar.
2. Leer el número de puntos n.
3. Reservar memoria para $Xn(n) = x_n$ y $Fn(n) = f(x_n)$.
4. Leer $x_n \to Xn(n)$ y $f(x_n) \to Fn(n)$.
5. Leer $x \to x$ y $f(x) \to f$.
6. Reservar memoria para $Tabla(n-1, n-1)$.
7. $Tabla = divDiff(Xn, Fn)$
8. Calcular s y h.
9. Reservar memoria para $N(n-1)$.
10. $N = prodFwd(s, h, n)$.
11. Definir $P = Fn(1)$.
12. Hacer para $2 \leq k \leq n$:
    12.1 $P = P + Tabla(1,k)N(k)$.
13. $AbsError = ABS(f - P)$.
14. Escribir P y $AbsErr$.
15. Liberar la memoria reservada.
16. Fin.

Algoritmo de Diferencias divididas atrasadas

Entrada: del archivo *datos*, el número de puntos n y los puntos $\{(x_1, f(x_1)), \ldots, (x_n, f(x_n))\}$; del archivo *punto*, el punto $(x, f(x))$. **Salida:** el valor aproximado $f(x) \approx P_{n-1}(x)$ y el error absoluto $|f(x) - P_{n-1}(x)|$. **Requiere:** funciones `divDiff(Xn,Fn)` y `prodBwd(s,h,n)`.

1. Iniciar.
2. Leer el número de puntos n.
3. Reservar memoria para $Xn(n) = x_n$ y $Fn(n) = f(x_n)$.
4. Leer $x_n \to Xn(n)$ y $f(x_n) \to Fn(n)$.
5. Leer $x \to x$ y $f(x) \to f$.
6. Reservar memoria para $Tabla(n-1, n-1)$.
7. $Tabla = divDiff(Xn, Fn)$
8. Calcular s y h.
9. Reservar memoria para $N(n-1)$.
10. $N = prodBwd(s, h, n)$.
11. Definir $P = Fn(1)$. ← revisar Fn(n)
12. Hacer para $2 \leq k \leq n$:
    12.1 Tarea.
13. $AbsError = ABS(f - P)$.
14. Escribir P y $AbsErr$.
15. Liberar la memoria reservada.
16. Fin.

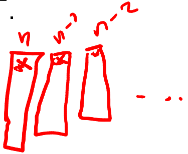
Algoritmo de Diferencias divididas centradas

Tarea

Algoritmo de la Tabla de Diferencias divididas de Newton

Función `divDiff(Xn,Fn)`. **Entrada:** el conjunto de puntos $\{(x_1, f(x_1)), \ldots, (x_n, f(x_n))\}$. **Salida:** un arreglo bidimensional de dimensión $(n-1) \times (n-1)$ con el valor de las diferencias divididas.

1. Iniciar.
2. Definir $n = SIZE(Xn, 1)$
3. Reservar memoria para $T(n-1, n-1)$.
4. Hacer para $2 \leq k \leq n-1$:
   - 4.1 Tarea: ¿cómo calcular las diferencias divididas?
5. Retornar T.
6. Fin.



Vector de punteros

Programación defensiva: validar los valores de n y de x.
Optimizable: el arreglo *Tabla* tiene $n-1$ columnas. La i-ésima columna tiene las diferencias de orden i y tiene $n-i$ filas. Si se deja la primera diferencia de orden i en la primera fila de la i-ésima columna, la última diferencia estará en la fila $n-i$.
Reutilizable: se puede usar la función `divDiff(Xn,Fn)` para las otras versiones del polinomio de diferencias divididas.

Algoritmo del Producto de nodos

Función `prodNodes(x,Xn)`. **Entrada:** el conjunto de nodos $\{x_n\}$ y el nodo de interés x. **Salida:** un arreglo unidimensional de dimensión $(n-1)$ con el valor de los productos $(x-x_1)\cdots(x-x_{k-1})$

Programación defensiva: validar los valores de n y de x.

1. Iniciar.
2. Definir $n = SIZE(Xn, 1)$
3. Reservar memoria para $Q(n-1)$.
4. Hacer para $2 \le k \le n$:
   4.1 $Q(k) = (\cancel{x - x_1}) \cdots (x - x_{k-1})$
5. Retornar Q.
6. Fin.

$Q(1) = (x - x_1)$

$Q(k) = Q(1) * (x - x_{k-1})$

Algoritmo del Producto de nodos para las diferencias adelantadas

Función `prodFwd(s,h,n)`. **Entrada:** el incremento s, el paso h y el número de nodos n. **Salida:** un arreglo unidimensional de dimensión $(n-1)$ con el valor de los productos $(s-k+2)\cdots(s-1)sh^{k-1}$.

1. Iniciar.
2. Reservar memoria para $Q(n-1)$.
3. Hacer para $2 \leq k \leq n$:
   3.1 $Q(k) = sh^{k-1}(s-1)\cdots(s-k+2)$
4. Retornar Q.
5. Fin.

Programación defensiva: validar el valor de n.

Tarea: ¿cómo se hace el producto del punto 3.1?

Productos de matrices y vectores

Algoritmo del Producto de nodos para las diferencias atrasadas

revisar si es s+1

Función `prodBwd(s,h,n)`. **Entrada:** el incremento s, el paso h y el número de nodos n. **Salida:** un arreglo unidimensional de dimensión $(n-1)$ con el valor de los productos $(s+k-2)\cdots(s-1)sh^{k-1}$.

1. Iniciar.
2. Reservar memoria para $Q(n-1)$.
3. Hacer para $2 \leq k \leq n$:
   3.1 $Q(k) = sh^{k-1}(s-1)\cdots(s+k-2)$
4. Retornar Q.
5. Fin.

Programación defensiva: validar el valor de n.

Tarea: ¿cómo se hace el producto del punto 3.1?

Polinomio de Hermite usando el polinomio de Newton

- Consideremos los n puntos $\{(x_1, f(x_1)), \ldots, (x_n, f(x_n))\}$.
- Hay que definir una nueva secuencia de nodos $z_0, \ldots, z_{2n+1}$, con $z_{2i} = z_{2i+1} = x_i$.
- Como $z_{2i} = z_{2i+1} = x_i$, no se puede definir $f[z_{2i}, z_{2i+1}]$ de la forma tradicional, así que se usa $f[z_{2i}, z_{2i+1}] = f'(z_{2i}) = f'(x_i)$.
- Entonces, se puede definir el polinomio de Hermite con

$$H_{2n-1}(x) = f[z_1] + \sum_{k=2}^{2n} f[z_1, \ldots, z_k](x - z_1)(x - z_2)\cdots(x - z_{k-1}).$$

- Para $n = 3$, se tendrán $2n = 6$ valores $\{z_i\}$ y

$$\begin{aligned} H_5(x) = & f[z_1] + f[z_1, z_2](x - z_1) + f[z_1, z_2, z_3](x - z_1)(x - z_2) \\ & + f[z_1, z_2, z_3, z_4](x - z_1)(x - z_2)(x - z_3) + f[z_1, z_2, z_3, z_4, z_5]\cdot \\ & (x - z_1)(x - z_2)(x - z_3)(x - z_4) + f[z_1, z_2, z_3, z_4, z_5, z_6]\cdot \\ & (x - z_1)(x - z_2)(x - z_3)(x - z_4)(x - z_5). \end{aligned}$$

Algoritmo de Hermite

Entrada: del archivo *datos*, el número de puntos n y los puntos $\{(x_1, f(x_1)), \ldots, (x_n, f(x_n))\}$; del archivo *punto*, el punto $(x, f(x))$. **Salida:** el valor aproximado $f(x) \approx H_{2n-1}(x)$ y el error absoluto $|f(x) - H_{2n-1}(x)|$. **Requiere:** implementación del polinomio de diferencias divididas de Newton.

1. Iniciar.
2. Leer el número de puntos n.
3. Reservar memoria para $Xn(n) = x_n$ y $Fn(n) = f(x_n)$.
4. Leer $x_n \rightarrow Xn(n)$ y $f(x_n) \rightarrow Fn(n)$.
5. Leer $x \rightarrow x$ y $f(x) \rightarrow f$.
6. Reservar memoria para $Zn(2n)$ y $FZn(2n)$.
7. Hacer para $1 \leq i \leq n$:
   - 8.1 Definir $Zn(2i) = Xn(i)$
   - 8.2 Definir $Zn(2i+1) = Xn(i)$
   - 8.3 Definir $FZn(2i) = Fn(i)$
   - 8.4 Definir $FZn(2i+1) = Fn(i)$
8. Encontrar el polinomio de diferencias divididas de Newton usando como datos los $2n$ puntos $\{(Zn(i), FZn(i))\}$.
9. Fin.

¡Muchas gracias!

Contacto:

Giovanni Ramírez García, PhD
ramirez@ecfm.usac.edu.gt
http://ecfm.usac.edu.gt/ramirez