Tarea 2
Algoritmos y Complejidad

Interpolación cuadrática inversa

Benjamín Camus
202173072-9

2023-2

Concepto	**Tiempo [min]**
Revisión	330
Desarrollo	310
Informe	210

Podemos considerar el método de la secante como interpolando para $y = 0$ entre los puntos $(y_n, x_n), (y_{n+1}, x_{n+1})$ para dar x_{n+2}. Una extensión obvia es interpolar para $y = 0$ entre los puntos $(y_n, x_n), (y_{n+1}, x_{n+1}), (y_{n+2}, x_{n+2})$ para dar x_{n+3}. Derive la iteración para este método. Note que la fórmula resultante de usar interpolación de Lagrange es numéricamente inestable, plantéela en la forma:

$$x_{n+3} = x_{n+2} + \delta$$

Para esto conviene usar la forma de Newton.

Escriba una función con prototipo:

```
double zero(double f(double),
            double x0, double x1, double x2,
            double err);
```

y úsela para hallar el valor de w tal que $we^w = 6$ con 5 cifras. (La función $W(x)$ definida mediante $W(x)e^{W(x)} = x$ se conoce como W *de Lambert*.)

Solución

Para encontrar la solución de w tal que $we^w = 6$, tenemos que despejar la función W de Lambert, denotada como $W(x)$, que es la función inversa de $f(x) = x \cdot e^x$. Al aplicar la función de Lambert a los dos lados de la ecuación nos queda w = W(6), para ver el valor al que debemos llegar introducimos la función a WolframAlpha, el cual nos da un valor para w de $1,4324047758...$, debemos llegar a una aproximación

con 5 cifras, es decir 10^{-5}.
Para ello realizaremos una interpolación inversa mediante la interpolación de Newton usando el método de las diferencias divididas. El método de Newton consiste en armar un polinomio de interpolación y como queremos realizar el método para los puntos $(y_n, x_n), (y_{n+1}, x_{n+1}), (y_{n+2}, x_{n+2})$, primero notar que los valores de las coordendas están cambiados de orden de como deberían ser x e y, esto es porque queremos calcular la inversa, pero inicialmente se explicara el método para un (x_n, y_n) lo importante es tener en cuenta que tenemos que calcular la inversa invirtiendo los valores de x e y. El polinomio resultante estará formado por unos valores a_0, a_1, a_2 donde el polinomio será $P(x) = a_0 + a_1(x - x_1) + a_2(x - x_1)(x - x_2)$. Para saber estos valores es útil mirar una tabla como la siguiente, en donde nuestro $f(x)$ surge al despejar la función original y nos queda como $f(w) = we^w - 6$.
DD1, DD2 corresponden a los grados de los términos de diferencias divididas.

x	$f(x)$	$DD1$	$DD2$
x_0	y_0		
		a_1	
x_1	y_1		a_2
		a_{med}	
x_2	y_2		

Tenemos que a_0 es igual al valor de y_0, el valor de $a_1 = \frac{y_1 - y_0}{x_1 - x_0}$, a_{med} corresponde a un a intermedio utilizado para calcular a_2, el cual vale $a_{med} = \frac{y_2 - y_1}{x_2 - x_1}$, y el valor de $a_2 = \frac{a_{med} - a_1}{x_2 - x_0}$, a partir de esto y observando la tabla podemos ver que existe como un patron visual de la tabla para calcular los a.
La idea del código a realizar es primeramente declarar la función que queremos encontrar, es decir $f(w) = we^w - 6$, la cual recibe como parámetro los valores de x_n, luego la funcion principal a programar, que corresponde al prototipo de:

```
double zero(double f(double),
            double x0, double x1, double x2,
            double err);
```

Tenemos que encontrar un método iterativo para aproximar el valor de $W(6)$, para ello empezamos un bucle infinito y obtenemos los valores de y_0, y_1, y_2 llamando a la función f(double), mediante estos resultados calculamos los valores de a_0, a_1 y de a_2, luego a una variable llamada xn le asignamos el valor obtenido de evaluar $x = 0$ en el polinomio de interpolación de Newton, y aquí viene la parte importante, en la cual tenemos que ir rotando los valores de las x, como ya no nos interesa el valor de la x_2 vamos rotando los valores de la siguiente manera:
$x_2 = x_1$;
$x_1 = x_0$;
$x_0 = xn$;
Luego la interacción vuelve a empezar, pero debemos agregar una condición de termino para frenar el bucle, la cual debe estar luego de calcular el valor del polinomio xn pero antes de rotar los valores de las x, esto es para hacer el calculo correctamente, la condición de término será que $|xn - x_0| \leq$ err.

Ahora como debemos calcular la inversa tenemos 2 opciones, o podemos invertir los valores de las x e y antes y dentro del while mediante algunas condiciones por posibles errores, o podemos invertir las columnas x y $f(x)$ de la tabla de tal forma que nos queda de la siguiente forma, esto es para invertir los valores pero en la formula para el cálculo del polinomio:

$f(x)$	x	$DD1$	$DD2$
y_0	x_0		
		a_1	
y_1	x_1		a_2
		a_{med}	
y_2	x_2		

Al cambiar el orden de esta manera estamos cambiando los valores de los a_0, a_1, a_{med}, a_2 quedando como $a_0 = x_0$, el valor de $a_1 = \frac{x_1 - x_0}{y_1 - y_0}$, $a_{med} = \frac{x_2 - x_1}{y_2 - y_1}$, y el valor de $a_2 = \frac{a_{med} - a_1}{y_2 - y_0}$, de esta forma invertimos los valores pero indirectamente y el polinomio nos queda como $xn = a_0 + a_1(0 - f(x_0)) + a_2(0 - f(x_0))(0 - f(x_1))$.

A la función como puntos iniciales debemos entregarle valores cercanos al valor buscado, en el código de la tarea los puntos iniciales son 1.0 , 2.0 y 3.0, con estos valores se puede encontrar perfectamente el valor buscado.

Para acompañar la explicación del código se muestra la función realizada:

```

    do {
        //Evaluar valores de x en la función
        y0 = f(x0);
        y1 =  f(x1);
        y2 = f(x2);

        //con los yn calcular los coeficientes del polinomio
        a0 = x0;
        a1 = (x1-x0)/(y1-y0);
        a_intermedio = (x2-x1)/(y2-y1);
        a2 = (a_intermedio- a1)/(y2-y0);

        //Usar estos valores para sacar el polinomio y calcular su valor donde y es 0
        xn = a0 + a1*(0-y0) + a2*(0-y0)*(0-y1);


        diferencia = fabs(xn - x0); // Calcula la diferencia entre iteraciones
        if (diferencia <= err) {
            break; // Detiene el bucle si la diferencia es menor que la tolerancia o error
        }

        //Rotar los valores
        x2 = x1;
        x1 = x0;
        x0 = xn;

        //para ir observando la precisión del método en cada iteración
        //printf("Valor: %.12lf\n", xn);
        //sleep(1);
    } while (1);
```