CENTRO DE CIENCIAS BÁSICAS

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

OPTIMIZACIÓN INTELIGENTE

5° "A"

PRÁCTICA 2: MÉTODO DE INTERPOLACIÓN CUADRADA

Profesor: Aurora Torres Soto

Alumno: Joel Alejandro Espinoza Sánchez

Fecha de Entrega: Aguascalientes, Ags**., 28** de septiembre de 2020

Práctica 2: Método de Interpolación Cuadrada

Objetivo:

Mediante el desarrollo de esta práctica, implementar el algoritmo de la sección.

Introducción:

Una buena aproximación para la determinación de un óptimo cuando tenemos funciones de una variable es mediante el uso de una función cuadrática. Debido a que hay únicamente una ecuación cuadrática que pasa por tres puntos, si se tiene tres puntos que contienen un punto óptimo (x_0, x_1 y x_2), se puede ajustar una parábola a los puntos y posteriormente derivar e igualar el resultado a cero, para obtener una estimación de la x óptima.

Este método construye un polinomio de interpolación de segundo grado, para el que se determina el óptimo x_3 y posteriormente este punto se utiliza como uno de los puntos a los que se ajustará otro polinomio.

El valor del óptimo x_3 en cada iteración deberá ser agregado a los puntos de partida, lo que produce el desplazamiento de uno de los extremos y la eliminación de un subintervalo.

$$x_3 = \frac{f(x_0)(x_1^2 - x_2^2) + f(x_1)(x_2^2 - x_0^2) + f(x_2)(x_0^2 - x_1^2)}{2f(x_0)(x_1 - x_2) + 2f(x_1)(x_2 - x_0) + 2f(x_2)(x_0 - x_1)}$$

Si x_3 se encuentra entre x_0 y x_1 deberemos eliminar el subintervalo superior.

Si x_3 se encuentra entre x_1 y x_2 deberemos eliminar el subintervalo inferior.

Este proceso se repite hasta que se alcance el óptimo o se cumpla un criterio de paro predeterminado.

El criterio que comúnmente se emplea para detener el proceso iterativo es que el error aproximado ea sea menor que la tolerancia de error preestablecida es.

$$ea = \left| \frac{x_{3_{actual}} - x_{3_{anterior}}}{x_{3_{actual}}} \right| \times 100$$

Pregunta de Investigación:

¿De qué manera pueden adaptarse las ideas matemáticas del método de interpolación cuadrada en un algoritmo de programación para que una computadora desarrolle el método de forma más eficiente?

Predicción:

Creo que es importante considerar las condiciones para que el algoritmo actúe de alguna u otra manera en función de cómo encuentre la función. Si se puede adaptar el algoritmo para este propósito, la computadora podría desarrollar perfectamente el método en cuestión.

Materiales:

Una computadora con compilador de C.
Una calculadora.
Dos hojas de papel
Lápiz o plumas

Método (Variables):

Dependiente: El resultado que arroje el programa según el óptimo hallado.
Independiente: El algoritmo que uno como alumno se focalizará en elaborar.
Controlada: La función matemática que se usará para evaluar.

Seguridad:

Realmente no se trabajó en campo, por lo que no se corren riesgos al elaborar el experimento.

Procedimiento:

1.- Haciendo uso del lenguaje C, se desarrolló un programa que solicite al usuario los puntos en los que se debe basar la ecuación cuadrática (x_0, x_1 y x_2), el número máximo de iteraciones y el valor de la tolerancia de error.

2.- Se calculó y se mostró el máximo para la función $f(x) = 2sen(x) - \frac{x^2}{2}$ que se encontraba comprendido en el intervalo $[0,4]$.

3.- El programa se probó con $x_0 = 0$, $x_1 = 1$, $x_2 = 4$ con 10 iteraciones y $es = 0.01$. Los resultados se procesaron en el presente documento a continuación.

Obtención y Procesamiento de Datos:

Al terminar de desarrollar el código en C para ejecutar el método en cuestión (véase anexo 1) y calibrarlo con los valores mencionados en el procedimiento, es decir, $x_0 = 0, x_1 = 1, x_2 = 4$ con 10 iteraciones y $es = 0.01$, el programa despliega al usuario la tabla que se observa dividida en las figuras 1 y 2 (véase anexo 2 para observar las capturas de la impresión completa de pantalla).

i	x0	x1	x2	x3
0	0.000	1.000	4.000	---
1	0.000	1.000	4.000	0.998
2	1.000	0.998	4.000	1.033
3	1.000	1.033	0.998	1.030
4	1.033	1.030	0.998	1.030

Figura 1: Primera parte de la tabla generada al probar el método de Interpolación Cuadrada evaluando la función $f(x) = 2sen(x) - \frac{x^2}{2}$ con $x_0 = 0$, $x_1 = 1, x_2 = 4$, con 10 iteraciones y $es = 0.01$.

f(x0)	f(x1)	f(x2)	f(x3)		ea	es
0.000	1.183	-9.514	---		---	0.010
0.000	1.183	-9.514	1.183		100.000	0.010
1.183	1.183	-9.514	1.184		3.389	0.010
1.183	1.184	1.183	1.184		0.307	0.010
1.184	1.184	1.183	1.184		0.003	0.010

Figura 2: Segunda parte de la tabla generada al probar el método de Interpolación Cuadrada evaluando la función $f(x) = 2sen(x) - \frac{x^2}{2}$ con $x_0 = 0$, $x_1 = 1, x_2 = 4$, con 10 iteraciones y $es = 0.01$.

Al final el programa comenta que le tomó 4 iteraciones en encontrar un máximo para la función dada en $x = 1.023$ que equivale a $f(x) = 1.184$ con error de 0.03.

Es posible observar en las tablas que el método va descartando secciones según los resultados de $f(x_0)$, $f(x_1)$ y de $f(x_2)$, y de hecho, puede notarse que este método realiza en menos de la mitad de iteraciones el trabajo que en comparación realiza el Método de la Sección Dorada, pues recordemos que evaluando la misma función, a dicha función le tomó 10 iteraciones conseguir llegar a un error todavía considerable de 1.2 aproximadamente.

Finalmente, comparando el resultado arrojado por el programa, podemos compararlo con los resultados analíticos de una calculadora graficadora, como puede observarse en la figura 3, y realizar el contraste de los resultados numéricos.

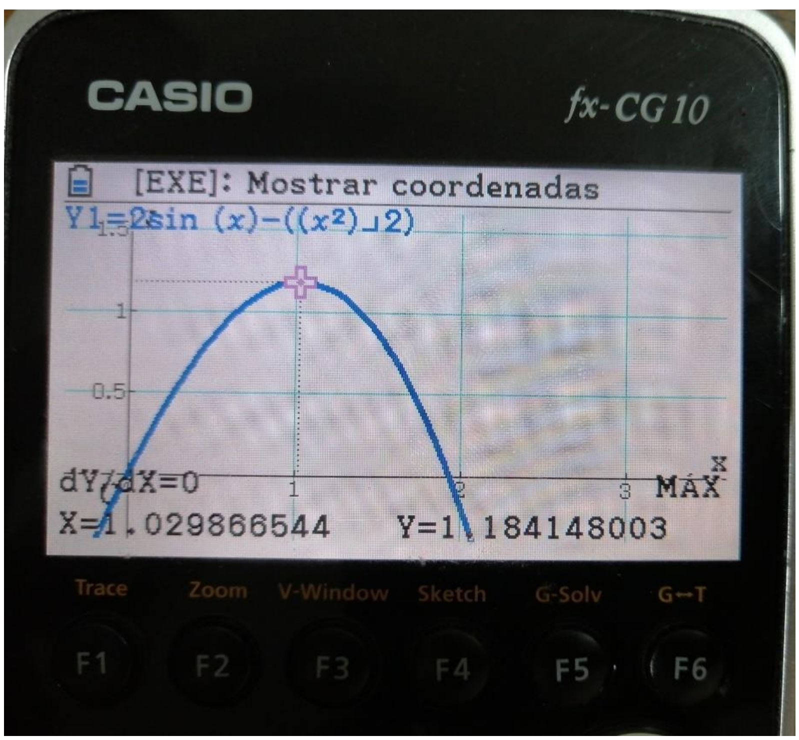

Figura 3: Calculadora Casio fx-CG10 mostrando la gráfica de la función $f(x) = 2sen(x) - \frac{x^2}{2}$ y su máximo.

Es posible observar que la solución analítica proporcionada por una calculadora graficadora fija el máximo de la función en $x = 1.02986$ mientras que la solución numérica obtenida es $x = 1.023$ por lo que observamos que fue una buena aproximación al máximo por lo que la realización de esta práctica bajo la interpolación cuadrática fue correcta. Aún podemos concluir contrastando este método con el desarrollado en la práctica anterior.

Conclusiones:

Podemos observar que ambos métodos están pensados para el mismo objetivo, sin embargo están elaborados por vías distintas, pues el primer método trataba de encontrar máximos y mínimos a partir del sesgo de la función en el intervalo y eliminar esas secciones en las que se sabe, por teoremas de continuidad, que el extremos no estaría en ese sitio.

Por otro lado, la interpolación cuadrática se basa en el conocimiento del comportamiento de las funciones cuadráticas, sabiendo perfectamente que dos funciones cuadráticas no son iguales refiriéndonos a que dos curvas cuadráticas no pasan por los mismos tres puntos.

Si tomamos en cuenta este hecho matemático, se puede elaborar el algoritmo elaborado en esta práctica, el cual hemos observado que es sumamente eficiente, puesto que en cuatro iteraciones se encontraba perfectamente acotado en un error muy pequeño y una aproximación muy confiable.

Podemos concluir que ambos métodos son muy importantes para el desarrollo del conocimiento en esta área, ya sea de forma introductoria como el primer método o para observar a mayor profundidad los métodos que existen para los trabajos de optimización que deseamos realizar.

Referencias:

Purcell, E. (2007). *Cálculo.* Londres: Pearson Education.

Talbi, E. (2009). *Metaheuristics from design to implementation.* New Jersey: John Wiley & Sons Publication.

Torres, A. (2020). *Apuntes: Optimización Inteligente.* 5° ICI. México: Universidad Autónoma de Aguascalientes.

Anexos:

Anexo 1: Código del programa en lenguaje C:

```
/*
    Universidad Autónoma de Aguascalientes

             Centro de Ciencias Básicas
  Departamento de Ciencias de la Computación
              Optimización Inteligente

                           5° "A"

   Práctica 2: Método de Interpolación Cuadrática

             Doctora Aurora Torres Soto

   Alumno: Joel Alejandro Espinoza Sánchez

  Fecha de Entrega: 28 de septiembre del 2020

Descripción:
*/
//Cargamos las librerías
#include <stdio.h>
#include <locale.h>
#include <math.h>

//Declaramos las funciones del programa
float f(float x);

//Declaramos la función principal
main()
{
	setlocale(LC_ALL,"");

	//Declaramos las variables que usaremos
	int i,j,u;
	float ea,es,x0,x1,x2,x3ant=0,x3sig=0;

	printf("========================== MÉTODO DE INTERPOLACIÓN
CUADRADA ==========================\n");
	printf("\n");
```

```
	do
	{
		//Pedimos x0, x1, x2, el número máximo de iteraciones y
la tolerancia de error
		printf("--------------------------------\n");
		printf("Otorgue x0: ");
		scanf("%f",&x0);
		printf("--------------------------------\n");
		printf("Otorgue x1: ");
		scanf("%f",&x1);
		printf("--------------------------------\n");
		printf("Otorgue x2: ");
		scanf("%f",&x2);
		printf("--------------------------------\n");
		printf("Otorgue el número máximo de iteraciones: ");
		scanf("%d",&i);
		printf("--------------------------------\n");
		printf("Otorgue la tolerancia de error: ");
		scanf("%f",&es);
		printf("--------------------------------\n\n");

		//Comienza la tabla de valores

	printf("=============================================================
==============================================================
========================\n");
		printf("| i |   x0    |   x1    |  x2   |   x3    |
|  f(x0)  |  f(x1)  | f(x2) |  f(x3)  |       |    ea    |    es
|\n");
		printf("| 0 |  %.3f  |  %.3f  |  %.3f  |   ---   |
|  %.3f  |  %.3f  |  %.3f  |   ---   |       |   ---   |  %.3f
|\n",x0,x1,x2,f(x0),f(x1),f(x2),es);

		j = 0;
		//Comienzan las iteraciones
		do
		{
			x3ant = x3sig;
			x3sig = (((f(x0))*((pow(x1,2)) - (pow(x2,2)))) +
((f(x1))*((pow(x2,2)) - (pow(x0,2)))) + ((f(x2)) * ((pow(x0,2)) -
(pow(x1,2)))))/((2*(f(x0))*(x1 - x2)) + (2*(f(x1))*(x2 - x0)) +
(2*(f(x2))*(x0 - x1)));
```

```
			ea = 100*fabs((x3sig - x3ant)/(x3sig));

			printf("| %d |  %.3f  |  %.3f  |  %.3f  |  %.3f  |
|  %.3f  |  %.3f  |  %.3f  |  %.3f  |       |  %.3f  |  %.3f
|\n",j+1,x0,x1,x2,x3sig,f(x0),f(x1),f(x2),f(x3sig),ea,es);

			//Se tomarán caminos según se haya registrado el
valor del actual x3
			if(x1 < x3sig && x3sig < x2)
			{
				x2 = x1;
				x1 = x3sig;
			}
			if(x0 < x3sig && x3sig < x1)
			{
				x0 = x1;
				x1 = x3sig;
			}

			j++;
		}
		while(j < i && es < ea);

		//Resultado final

	printf("==========================================================
=============================================================
========================\n\n");
		printf("El programa ha tomado %d iteraciones en
encontrar un extremo en x = %.3f el cual equivale a f(x) = %.3f
con un error de %0.3f\n\n\n",j,x3sig,f(x3sig),ea);

		//Opción para repetir el procedimiento
		printf("¿Desea repetir el procedimiento?\n");
		printf("1. Sí\n");
		printf("2. No\n");
		scanf("%d",&u);
	}
	while(u == 1);

	getchar();
```

```
}

float f(float x)
{
	return (2*(sin(x))) - (x*x/2);
}
```

Anexo 2: Impresión de pantalla completa al ejecutar el programa

```
========================== MÉTODO DE INTERPOLACIÓN CUADRADA ==========================

--------------------------------
Otorgue x0: 0
--------------------------------
Otorgue x1: 1
--------------------------------
Otorgue x2: 4
--------------------------------
Otorgue el número máximo de iteraciones: 10
--------------------------------
Otorgue la tolerancia de error: 0.01
--------------------------------

=====================================================================================================================
| i |   x0    |   x1    |  x2   |   x3    |       |  f(x0)  |  f(x1)  | f(x2) |  f(x3)  |       |    ea    |    es    |
| 0 |  0.000  |  1.000  | 4.000 |   ---   |       |  0.000  |  1.183  | -9.514 |   ---   |       |   ---   |  0.010  |
| 1 |  0.000  |  1.000  | 4.000 |  0.998  |       |  0.000  |  1.183  | -9.514 |  1.183  |       |  100.000  |  0.010  |
| 2 |  1.000  |  0.998  | 4.000 |  1.033  |       |  1.183  |  1.183  | -9.514 |  1.184  |       |  3.389  |  0.010  |
| 3 |  1.000  |  1.033  | 0.998 |  1.030  |       |  1.183  |  1.184  | 1.183 |  1.184  |       |  0.307  |  0.010  |
| 4 |  1.033  |  1.030  | 0.998 |  1.030  |       |  1.184  |  1.184  | 1.183 |  1.184  |       |  0.003  |  0.010  |
=====================================================================================================================

El programa ha tomado 4 iteraciones en encontrar un extremo en x = 1.030 el cual equivale a f(x) = 1.184 con un error de 0.003


¿Desea repetir el procedimiento?
1. Sí
2. No
```

Anexo 3: Algunas capturas del código en el IDE

```c
/*
    Universidad Autónoma de Aguascalientes

          Centro de Ciencias Básicas
  Departamento de Ciencias de la Computación
           Optimización Inteligente

                    5° "A"

   Práctica 2: Método de Interpolación Cuadrática

          Doctora Aurora Torres Soto

   Alumno: Joel Alejandro Espinoza Sánchez

  Fecha de Entrega: 28 de septiembre del 2020

Descripción:
*/
//Cargamos las librerías
#include <stdio.h>
#include <locale.h>
#include <math.h>

//Declaramos las funciones del programa
float f(float x);

//Declaramos la función principal
main()
{
    setlocale(LC_ALL,"");

    //Declaramos las variables que usaremos
    int i,j,u;
    float ea,es,x0,x1,x2,x3ant=0,x3sig=0;
```

```c
printf("========================= MÉTODO DE INTERPOLACIÓN CUADRADA =========================\n");
printf("\n");
do
{
    //Pedimos x0, x1, x2, el número máximo de iteraciones y la tolerancia de error
    printf("--------------------------------\n");
    printf("Otorgue x0: ");
    scanf("%f",&x0);
    printf("--------------------------------\n");
    printf("Otorgue x1: ");
    scanf("%f",&x1);
    printf("--------------------------------\n");
    printf("Otorgue x2: ");
    scanf("%f",&x2);
    printf("--------------------------------\n");
    printf("Otorgue el número máximo de iteraciones: ");
    scanf("%d",&i);
    printf("--------------------------------\n");
    printf("Otorgue la tolerancia de error: ");
    scanf("%f",&es);
    printf("--------------------------------\n\n");

    //Comienza la tabla de valores
    printf("==============================================================================================================================
    printf("| i |   x0    |   x1    |  x2   |   x3    |       |  f(x0)  |  f(x1)  | f(x2) |  f(x3)  |       |    ea    |    es    |\n");
    printf("| 0 |  %.3f  |  %.3f  | %.3f  |   ---   |       | %.3f  | %.3f  | %.3f  |   ---   |       |   ---   |  %.3f  |\n",x0,x1,x2,

    j = 0;
    //Comienzan las iteraciones
    do
    {
        x3ant = x3sig;
        x3sig = (((f(x0))*((pow(x1,2)) - (pow(x2,2)))) + ((f(x1))*((pow(x2,2)) - (pow(x0,2)))) + ((f(x2)) * ((pow(x0,2)) - (pow(x1,2)))))/(

        ea = 100*fabs((x3sig - x3ant)/(x3sig));
```

```c
j = 0;
//Comienzan las iteraciones
do
{
    x3ant = x3sig;
    x3sig = (((f(x0))*((pow(x1,2)) - (pow(x2,2)))) + ((f(x1))*((pow(x2,2)) - (pow(x0,2)))) + ((f(x2)) * ((pow(x0,2)) - (pow(x1,2)))))/(

    ea = 100*fabs((x3sig - x3ant)/(x3sig));

    printf("| %d |  %.3f  |  %.3f  |  %.3f  |  %.3f  |       |  %.3f  |  %.3f  |  %.3f  |  %.3f  |       |  %.3f  |  %.3f  |\n",j+1,x0,

    //Se tomarán caminos según se haya registrado el valor del actual x3
    if(x1 < x3sig && x3sig < x2)
    {
        x2 = x1;
        x1 = x3sig;
    }
    if(x0 < x3sig && x3sig < x1)
    {
        x0 = x1;
        x1 = x3sig;
    }

    j++;
}
while(j < i && es < ea);

//Resultado final
printf("==============================================================================================================================
printf("El programa ha tomado %d iteraciones en encontrar un extremo en x = %.3f el cual equivale a f(x) = %.3f con un error de %0.3f\n

//Opción para repetir el procedimiento
printf("¿Desea repetir el procedimiento?\n");
printf("1. Sí\n");
printf("2. No\n");
scanf("%d",&u);
```