CENTRO DE CIENCIAS BÁSICAS

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

OPTIMIZACIÓN INTELIGENTE

5° "A"

PRÁCTICA 1: MÉTODO DE LA SECCIÓN DORADA

Profesor: Aurora Torres Soto

Alumno: Joel Alejandro Espinoza Sánchez

Fecha de Entrega: Aguascalientes, Ags.**, 24** de septiembre de 2020

Práctica 1: Método de la Sección Dorada

Objetivo:

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

Introducción:

La búsqueda de la sección dorada es una técnica de optimización, para funciones de una sola variable sin restricciones, sencilla y de propósito general.

El corazón de este método de reducción de secciones se basa en la sección dorada que corresponde a $R = \frac{\sqrt{5}-1}{2} = 0.61803$ ... y que se refiere a la razón del intervalo de búsqueda que se conserva en cada iteración $d = R(x_U - x_L)$.

En términos generales, el método sigue los pasos:

- Se parte del intervalo inicial x_L y x_U
- Se eligen dos puntos interiores de acuerdo con la razón dorada, de este modo: $[x_1 = x_L + R(x_U - x_L),\ x_2 = x_U - R(x_U - x_L)]$
- (Suponiendo que se desea maximizar) Se evalúa la función en esos puntos y
    - Si $f(x_1) > f(x_2)$, se elimina la sección a la izquierda de x_2 y $x_{2_{siguiente}} = x_1$ y $x_{1_{siguiente}} = x_L + R(x_U - x_L)$
    - Si $f(x_2) > f(x_1)$, se elimina la sección a la derecha del punto x_1 y $x_{1_{siguiente}} = x_2$ y $x_{2_{siguiente}} = x_U + R(x_U - x_L)$
- Se continúa iterando hasta alcanzar un criterio de paro.

En caso de que se requiera minimizar, de debe cambiar el símbolo $>$ por $<$.

Pregunta de Investigación:

¿De qué manera pueden adaptarse las ideas matemáticas del método de la sección dorada en un algoritmo de programación para que una computadora desarrolle el método más rápidamente?

Predicción:

Creo que es importante considerar las condiciones para que el algoritmo actúe de alguna u otra manera en función de como 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 límites del intervalo de búsqueda (xL, xU), 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 construyó de modo que calculara y mostrara tanto el valor óptimo de x "`xopt`" como el valor de la función evaluada en `xopt`, el número de iteraciones ejecutadas y el valor del error `ea`. Este programa termina si se ha alcanzado el número máximo de iteraciones o si $ea < es$. Donde $ea = (1 - R)\frac{|x_U - x_L|}{x_{opt}}(100)$.

4.- El programa se probó con el intervalo $[0,4]$ 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, en el intervalo $[0,4]$ 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	xL	x2	xopt	x1	xU
0	0.000	1.528	---	2.472	4.000
1	0.000	0.944	2.000	1.528	2.472
2	0.000	0.584	1.236	0.944	1.528
3	0.584	0.944	0.764	1.167	1.528
4	0.584	0.807	1.056	0.944	1.167
5	0.807	0.944	0.875	1.029	1.167
6	0.944	1.029	0.987	1.082	1.167
7	0.944	0.997	1.056	1.029	1.082
8	0.997	1.029	1.013	1.050	1.082
9	0.997	1.017	1.039	1.029	1.050
10	1.017	1.029	1.023	1.037	1.050

Figura 1: Primera parte de la tabla generada al probar el método de la Sección Dorada evaluando la función $f(x) = 2sen(x) - \frac{x^2}{2}$ en el intervalo $[0,4]$ con 10 iteraciones y $es = 0.01$.

f(xL)	f(x2)	f(xopt)	f(x1)	f(xU)		ea	es
0.000	0.831	---	-1.815	-9.514		---	0.010
0.000	1.174	-0.181	0.831	-1.815		47.214	0.010
0.000	0.932	1.125	1.174	0.831		47.214	0.010
0.932	1.174	1.092	1.158	0.831		47.214	0.010
0.932	1.119	1.183	1.174	1.158		21.115	0.010
1.119	1.174	1.152	1.184	1.158		15.738	0.010
1.174	1.184	1.182	1.180	1.158		8.628	0.010
1.174	1.183	1.183	1.184	1.180		4.984	0.010
1.183	1.184	1.184	1.184	1.180		3.210	0.010
1.183	1.184	1.184	1.184	1.184		1.934	0.010
1.184	1.184	1.184	1.184	1.184		1.214	0.010

Figura 2: Segunda parte de la tabla generada al probar el método de la Sección Dorada evaluando la función $f(x) = 2sen(x) - \frac{x^2}{2}$ en el intervalo $[0,4]$ con 10 iteraciones y $es = 0.01$.

Al final el programa comenta que le tomó 10 iteraciones en encontrar un máximo para la función dada en $x = 1.023$ que equivale a $f(x) = 1.184$ con error de 1.214.

Es posible observar en las tablas que el método va descartando secciones que no cree importantes al observar los resultados de $f(x_1)$ y de $f(x_2)$, pues como al comienzo se analizó el algoritmo a mano (véase anexo 3), fue fácil notar que el algoritmo va haciendo los cortes a partir de la proporción dorada.

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.

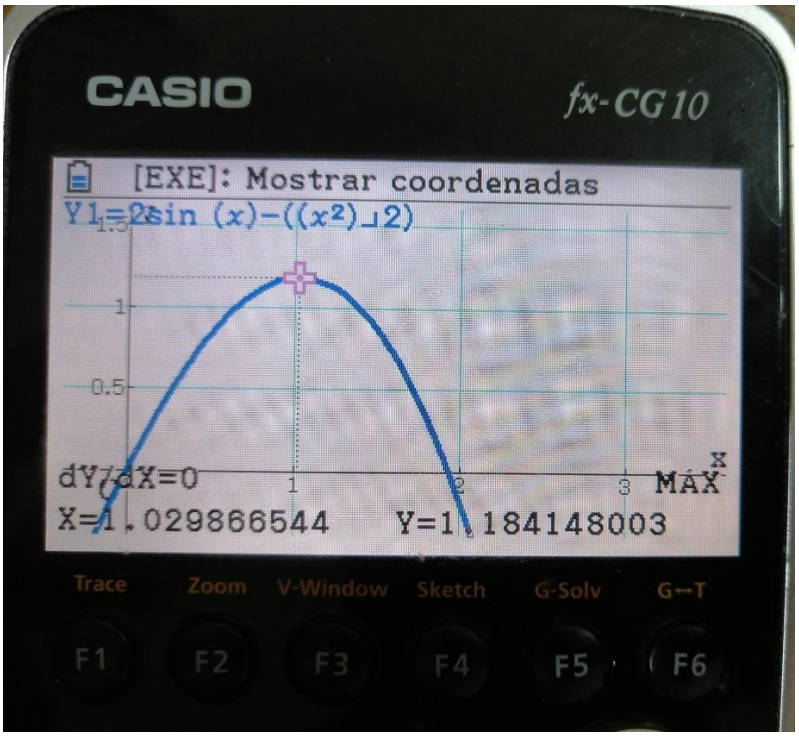


Figura 3: Calculadora Casio fx-CG10 mostrando la gráfica de la función $f(x) = 2sen(x) - \frac{x^2}{2}$ y su máximo.

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.

Conclusiones:

A partir de los resultados obtenidos podemos observar que es un muy buen resultado, incluso leyendo en notas teóricas de Edwin Purcell sobre el método que describe en su libro titulado “Cálculo”, comenta que este método es uno que reduce hasta un 50% las iteraciones, que de hecho podemos observar que gracias a seccionar el intervalo en razón de la proporción dorada, esto puede verse cierto.

Por lo que podemos concluir de la práctica que realmente la implementación del algoritmo de modo que las condiciones estuviera bien claras, permitió desarrollar un programa para ejecutar un método muy eficiente que calcule un máximo o un mínimo de una función y que es muy útil para abordar los temas de optimización en la computación.

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 1: Método de la Sección Dorada

            Doctora Aurora Torres Soto

   Alumno: Joel Alejandro Espinoza Sánchez

  Fecha de Entrega: 24 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);
void Maximizar(float xL, float xU, int i, float es);
void Minimizar(float xL, float xU, int i, float es);

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

     /*
     Declaramos las variables que usaremos
          i: El número máximo de iteraciones
          es: Valor de la tolerancia de error
          xL: El límite inferior del intervalo
```

```
          xU: El límite superior del intervalo
          u: Uso distinto para pedir maximización o minimización
     */
     int i,u;
     float es,xL,xU;

     printf("========================== MÉTODO DE LA SECCIÓN
DORADA ==========================\n");
     do
     {
          printf("\n");
          do
          {
               //Pedimos el límite inferior, el límite superior,
el número máximo de iteraciones y la tolerancia de error
               printf("--------------------------------\n");
               printf("Otorgue el límite inferior: ");
               scanf("%f",&xL);
               printf("--------------------------------\n");
               printf("Otorgue el límite superior: ");
               scanf("%f",&xU);
               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");

               //Para darle mayor generalidad al programa, él
mismo tratará de buscar la concavidad de la curva (de no
conseguirlo, pregunta al usuario)
               if((f(xL) < f(xL + ((xU - xL)/100))) && (f(xU) <
f(xU - ((xU - xL)/100)))) //Si la función es identificable como
cóncava hacia abajo
               {
                    Maximizar(xL,xU,i,es); //El programa ha
identificado que en el intervalo hay un máximo, así que se
maximizará
                    break;
               }
```

```
                if((f(xL + ((xU - xL)/100)) < f(xL)) && (f(xU -
((xU - xL)/100)) < f(xU))) //Si la función es identificable como
cóncava hacia arriba
                {
                     Minimizar(xL,xU,i,es); //El programa ha
identificado que en el intervalo hay un mínimo, así que se
minimizará
                     break;
                }
                //El programa llega hasta aquí si no consiguió
clasificar la función en cóncava o convexa, por lo que se le
pedirá al usuario definir
                printf("!  ------------------------------------
--------  !\n");
                printf("No se ha logrado determinar la concavidad
de la curva\n");
                printf("!  ------------------------------------
--------  !\n");
                printf("Defina si se va a maximizar o
minimizar:\n");
                printf("1. Maximizar\n");
                printf("2. Minimizar\n");
                scanf("%d",&u);
                printf("\n");
                if(u == 1)
                {
                     Maximizar(xL,xU,i,es); //Se maximizará
                     break;
                }
                if(u == 2)
                {
                     Minimizar(xL,xU,i,es); //Se minimizará
                     break;
                }
           }
           while(i == i);
           printf("¿Desea repetir el procedimiento?\n");
           printf("1. Sí\n");
           printf("2. No\n");
           scanf("%d",&u);
     }
     while(u == 1);
```

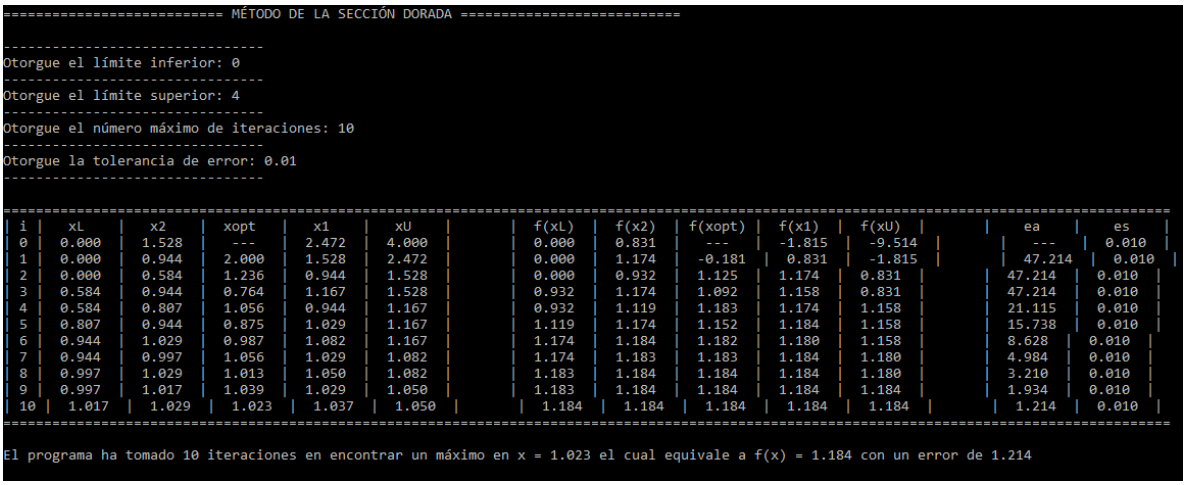
```
	getchar();
}

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

void Maximizar(float xL, float xU, int i, float es)
{
	//Declaramos más variables
	int j;
	float d,ea,R,x1,x2,xopt=(xU + xL)/2;

	//Damos valor a R y a d
	R =(sqrt(5) - 1)/2;
	d = R * (xU - xL);

	//Se eligen dos puntos interiores de acuerdo a la proporción
áurea
	x1 = xL + d;
	x2 = xU - d;

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

	//Comenzamos a iterar
	for(j = 0; j<i; j++)
	{
		//Evaluamos la función en esos puntos y comparamos
		xopt = (xU + xL)/2;
		if(f(x1) > f(x2)) //Si f(x1) es mayor a f(x2)
		{
```

```
                //Se elimina la sección a la izquierda de x2 por
lo tanto actualizamos como sigue los valores
                xL = x2;
           }
           else //Si f(x2) es mayor a f(x1)
           {
                //Se elimina la sección a la derecha de x1 por lo
tanto actualizamos como sigue los valores
                xU = x1;
           }

           d = R * (xU - xL);
           x1 = xL + d;
           x2 = xU - d;
           ea = 100 * (1 - R) * ((xU - xL)/(xopt));

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

           if(ea<=es) //Terminará el ciclo en caso de que se
alcance la tolerancia de error
           {
                break;
           }
     }
     printf("=================================================
================================================================
=======================\n\n");
     printf("El programa ha tomado %d iteraciones en encontrar un
máximo en x = %.3f el cual equivale a f(x) = %.3f con un error de
%0.3f\n\n\n",j,xopt,f(xopt),ea);
     return;
}
void Minimizar(float xL, float xU, int i, float es)
{
     //Declaramos más variables
     int j;
     float d,ea,R,x1,x2,xopt=(xU + xL)/2;

     //Damos valor a R y a d
```

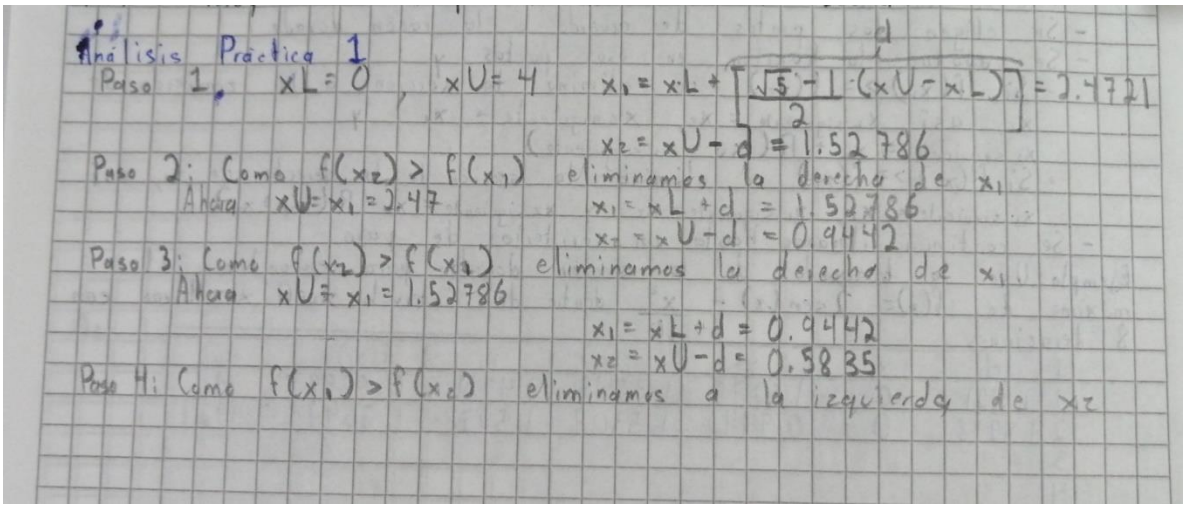
```
      R =(sqrt(5) - 1)/2;
      d = R * (xU - xL);

      //Se eligen dos puntos interiores de acuerdo a la proporción
áurea
      x1 = xL + d;
      x2 = xU - d;

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

      //Comenzamos a iterar
      for(j = 0; j<i; j++)
      {
            //Evaluamos la función en esos puntos y comparamos
            xopt = (xU + xL)/2;
            if(f(x1) < f(x2)) //Si f(x1) es menor a f(x2)
            {
                  //Se elimina la sección a la izquierda de x2 por
lo tanto actualizamos como sigue los valores
                  xL = x2;
            }
            else //Si f(x2) es menor a f(x1)
            {
                  //Se elimina la sección a la derecha de x1 por lo
tanto actualizamos como sigue los valores
                  xU = x1;
            }

            d = R * (xU - xL);
            x1 = xL + d;
            x2 = xU - d;
            ea = 100 * (1 - R) * ((xU - xL)/(xopt));
```

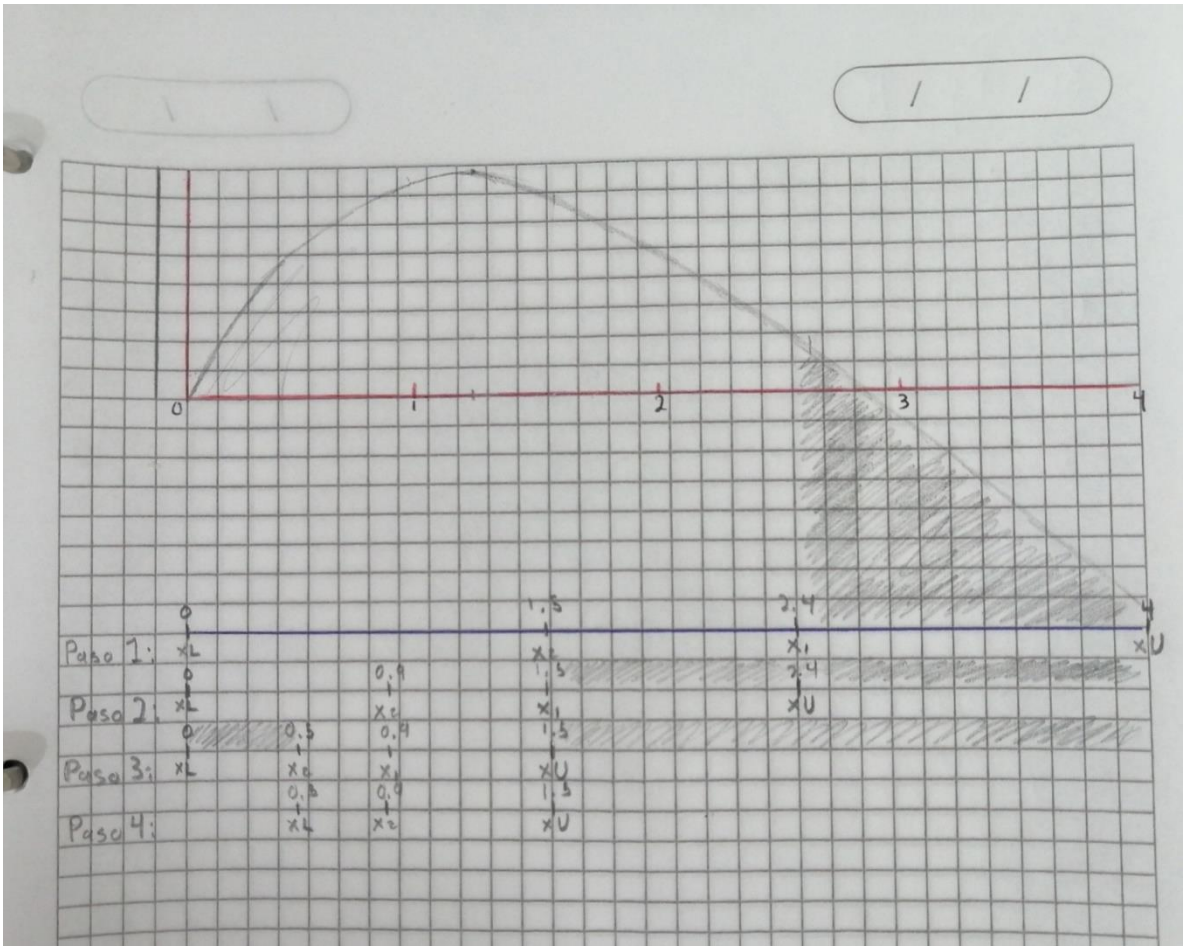
```
            printf("| %d |  %.3f  |  %.3f  |  %.3f  |  %.3f  |  %.3f
|       |  %.3f  |  %.3f  |  %.3f  |  %.3f  |  %.3f  |       |
%.3f  |  %.3f
|\n",j+1,xL,x2,xopt,x1,xU,f(xL),f(x2),f(xopt),f(x1),f(xU),ea,es);

            if(ea<=es) //Terminará el ciclo en caso de que se
alcance la tolerancia de error
            {
                  break;
            }
      }
      printf("=================================================
================================================================
========================\n\n");
      printf("El programa ha tomado %d iteraciones en encontrar un
mínimo en x = %.3f el cual equivale a f(x) = %.3f con un error de
%0.3f\n\n\n",j,xopt,f(xopt),ea);
      return;
}
```

Anexo 2: Impresión de pantalla completa al ejecutar el programa



Anexo 3: Análisis en papel y lápiz del algoritmo





Anexo 4: 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 1: Método de la Sección Dorada

            Doctora Aurora Torres Soto

   Alumno: Joel Alejandro Espinoza Sánchez

  Fecha de Entrega: 24 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);
void Maximizar(float xL, float xU, int i, float es);
void Minimizar(float xL, float xU, int i, float es);
```

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

    /*
    Declaramos las variables que usaremos
        i: El número máximo de iteraciones
        es: Valor de la tolerancia de error
        xL: El límite inferior del intervalo
        xU: El límite superior del intervalo
        u: Uso distinto para pedir maximización o minimización
    */
    int i,u;
    float es,xL,xU;

    printf("========================== MÉTODO DE LA SECCIÓN DORADA ==========================\n");
    do
    {
        printf("\n");
        do
        {
            //Pedimos el límite inferior, el límite superior, el número máximo de iteraciones y la tolerancia de error
            printf("--------------------------------\n");
            printf("Otorgue el límite inferior: ");
            scanf("%f",&xL);
            printf("--------------------------------\n");
            printf("Otorgue el límite superior: ");
            scanf("%f",&xU);
            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");
```

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

```c
//Damos valor a R y a d
R =(sqrt(5) - 1)/2;
d = R * (xU - xL);

//Se eligen dos puntos interiores de acuerdo a la proporción áurea
x1 = xL + d;
x2 = xU - d;

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

//Comenzamos a iterar
for(j = 0; j<i; j++)
{
    //Evaluamos la función en esos puntos y comparamos
    xopt = (xU + xL)/2;
    if(f(x1) > f(x2)) //Si f(x1) es mayor a f(x2)
    {
        //Se elimina la sección a la izquierda de x2 por lo tanto actualizamos como sigue los valores
        xL = x2;
    }
    else //Si f(x2) es mayor a f(x1)
    {
        //Se elimina la sección a la derecha de x1 por lo tanto actualizamos como sigue los valores
        xU = x1;
    }

    d = R * (xU - xL);
    x1 = xL + d;
    x2 = xU - d;
    ea = 100 * (1 - R) * ((xU - xL)/(xopt));

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

    if(ea<=es) //Terminará el ciclo en caso de que se alcance la tolerancia de error
    {
        break;
    }
}
printf("=============================================================================================================================================\n\n");
printf("El programa ha tomado %d iteraciones en encontrar un máximo en x = %.3f el cual equivale a f(x) = %.3f con un error de %0.3f\n\n\n\n",j,xopt,f(xopt),ea);
return;
```