

Tarea 1: Aplicación del método de Newton y heurísticas

Sistemas evolutivos
INAOE
Ciro Fabian Bermudez Marquez

3 de febrero de 2021

Encontrar los 3 máximos y 3 mínimos locales de la función $f(x)$ en el rango de $[0, 7]$ la cual tiene la gráfica mostrada en la Figura 1 y esta dada por la siguiente ecuación:

$$f(x) = (x - 2)(x - 5) + \sin(1.5\pi x) \quad (1)$$

para comenzar con el análisis es de gran ayuda conocer el comportamiento de la derivada de la función para poder hallar los puntos críticos, es decir cuando la derivada cruza por cero como se muestra en la Figura 2 y cuya ecuación es Ec. (2).

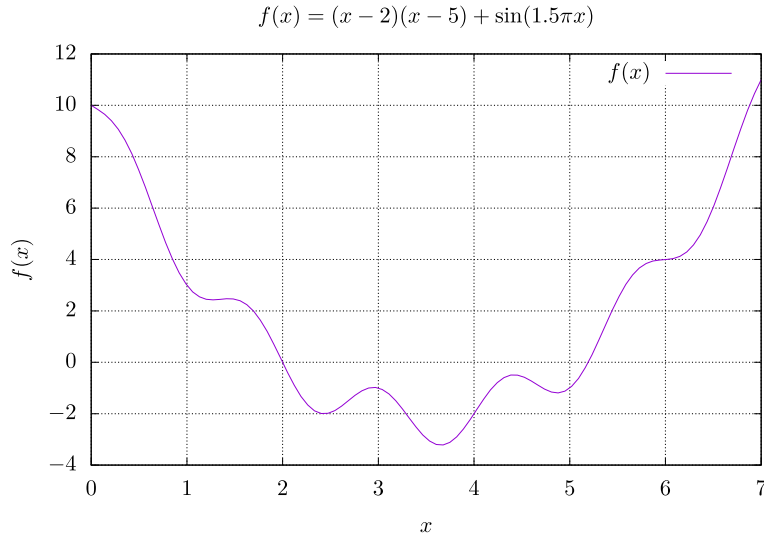


Figura 1: Grafica de la función a analizar.

$$f'(x) = 2x - 7 + 1.5\pi \cos(1.5\pi x) \quad (2)$$

Si analizamos detalladamente la Figura 2 podemos darnos cuenta que la derivada de la función cruza por cero un total de 7 veces en el intervalo de $[0, 7]$, si tratamos de resolver analíticamente este problema nos encontraremos con la problemática que para encontrar los puntos críticos igualando la derivada a cero obtenemos:

$$2x - 7 + 1.5\pi \cos(1.5\pi x) = 0 \quad (3)$$

la cual es una ecuación que no puede resolverse simbólicamente y es necesario utilizar métodos numéricos, en este caso utilizaremos el método de Newton para encontrar máximos y mínimos, la implementación del algoritmo se muestra en el Código 1.

La implementación requiere que calculamos la segunda derivada de la función la cual es:

$$f''(x) = 2 - (1.5\pi)^2 \sin(1.5\pi x) \quad (4)$$

después generamos una solución aleatoria¹ (ver Código 2) de $[0, 7]$ que sirva como entrada para el algoritmo de Newton. Para repetir esto 100 veces utilizamos el script en bash que se muestra en el Código 3, el cual escribe un archivo de texto **datosfloat.txt** que contiene los puntos x donde existe un máximo o un mínimo, sin embargo

¹La solución aleatoria puede ser entera o flotante, para este ejercicio utilizamos flotante.

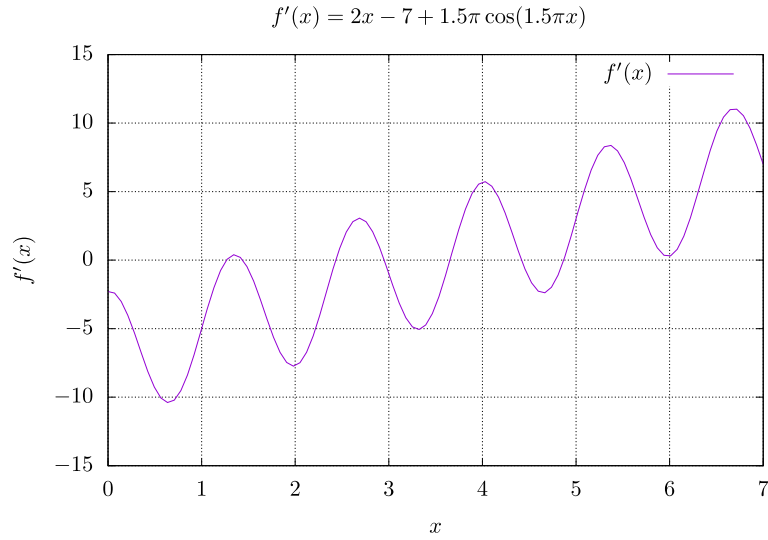


Figura 2: Grafica de la derivada de la función a analizar.

bajo ciertas entradas el método falló o la solución se encuentra fuera de los rangos deseados, y debido a esto es necesario comprobar cuales fueron las soluciones correctas, esto se hace simplemente ingresando los valores del archivo de texto a la derivada de la función y comprobando que sea menor a cierto umbral, en este caso basta con que sea menor a 1×10^{-4} , sin embargo optamos por contar el número de ocurrencias de todos los elementos del array para tener más información.

Para procesar los datos utilizamos el Código 4, el cual comprueba que la solución sea valida, después que no sea negativa, calculamos el número de ocurrencias de cada solución y comprobamos si es un máximo o un mínimo

Del Código 4 obtenemos los siguientes resultados:

x	Número de ocurrencias	Max o Min
1.264980827648	14	mínimo
1.441085745708	15	máximo
2.433055879795	10	mínimo
2.950004356645	4	máximo
3.652887442162	18	mínimo global
4.418288656759	7	máximo
4.868491681705	13	mínimo
fallos	10	ninguno
< 0	9	ninguno

Cuadro 1: Puntos donde hay un máximo o mínimo.

De los resultados se puede notar que el **mínimo global** ocurre en $x = 3.652887442162$ y que este ocurre 18 veces, como se hizo 100 veces el experimento esto equivale a un 18 %.

Algunas notas importantes a tener en cuenta es que para poder ejecutar los archivos .sh hay que cambiar los permisos de ejecución con el comando **chmod +x filename.sh** y que con cambios muy pequeños a los códigos se

```

1 # El metodo de Newton2
2 # Ciro Fabian Bermudez Marquez
3 # 02.02.2021
4 # Librerias
5 import math
6 import sys
7
8 # Funciones para el metodo
9 def df(x) :
10     return 2.0*x - 7 + 1.5*math.pi*math.cos(1.5*math.pi*x)
11
12 def ddf(x) :
13     return 2.0 - 1.5*math.pi*1.5*math.pi*math.sin(1.5*math.pi*x)
14
15 # El valor inicial de a
16 n = len(sys.argv)
17 if n != 2 :
18     print( "Args: valor-inicial" )
19     sys.exit(1)
20
21 a = float( sys.argv[1] )
22
23 # 50 iteraciones maximas
24 i = 1
25 #print("i \txi \t\tea")
26 while i <= 50 :
27     Deltax = -df(a)/ddf(a)
28     a += Deltax
29     #print("%ld \t%3.5f \t%3.5f" %(i,a,Deltax))
30
31     if abs(Deltax) < 1e-10 :
32         break
33
34     i += 1
35
36 #print( "x = ", a , "df(x) = ", df(a) )
37 # Solucion
38 print(a)

```

Código 1: Método de Newton para máximos y mínimos.

```

1 # Autor: Ciro Fabian Bermudez Marquez
2 # 02.02.2021
3
4 # Librerias
5 import sys
6 import random
7
8 n = len(sys.argv)
9 if n != 3 :
10     print( "Args valor inicial" )
11     sys.exit(1)
12
13 inicio = int( sys.argv[1] )
14 fin = int( sys.argv[2] )
15
16 # Para int
17 #print( random.randint( inicio, fin ) )
18 # Para float
19 print( random.uniform( inicio, fin ) )

```

Código 2: Generar número aleatorio flotante.

```

1 #! /bin/bash
2 # Autor: Ciro Fabian Bermudez Marquez
3 # 02.02.2021
4
5 for (( i=1; i<=100; i+=1))
6 do
7     randnum=$(python3 randomnum.py 0 7)
8     #echo $randnum
9     #echo $i
10     python3 newton2.py $randnum
11 done > datosfloat.txt

```

Código 3: Repeticiones.

```

1 # Autor:  Ciro Fabian Bermudez Marquez
2 # 02.02.2021
3
4 # Librerias
5 import math
6 import numpy as np
7 import collections
8
9 def df(x) :
10     return 2.0*x - 7 + 1.5*math.pi*math.cos(1.5*math.pi*x)
11
12
13 def ddf(x) :
14     return 2.0 - 1.5*math.pi*1.5*math.pi*math.sin(1.5*math.pi*x)
15
16
17 data = np.genfromtxt( "datosfloat.txt" )
18 tam = len(data)
19 xp = []
20 min_global = 3.6528874421621778
21
22 #cont = 0 # Apariciones minimo global
23 num_fallas = 0
24 num_neg = 0 # Apariciones de soluciones negativas
25 for i in range( tam ):
26     is_zero = df( data[i] )
27     if is_zero < 1e-4 :    # La solucion es correcta
28         if data[i] < 0 :
29             num_neg += 1
30         else :
31             #print( i ,data[i] )
32             xp.append( data[i] )
33         #if abs( data[i] - min_global ) <= 1e-4 : # Contar minimo
34             # cont += 1
35     else:
36         num_fallas += 1
37
38
39 xp = np.array(xp)          # Convertir a objeto de numpy
40 freq = collections.Counter(xp.round(12))
41
42 for key,value in freq.items() :
43     if ddf( float(key) ) < 0 :
44         s = "maximo"
45     elif ddf( float(key) ) > 0 :
46         s = "minimo"
47     else :
48         s = "ninguno"
49     print(key,value,s)
50
51 freq["fallos"] = num_fallas
52 freq["<0"] = num_neg
53 print(freq)
54 print(num_fallas,num_neg)

```

Código 4: Análisis de los datos.

Referencias

- [1] Apuntes y programas de clase Dr. Luis Gerardo de la Fraga.