Tarea 1: Aplicación del método de Newton y heurísticas

Sistemas evolutivos
INAOE
Ciro Fabian Bermudez Marquez
cirofabian.bermudez@gmail.com
https://github.com/cirofabianbermudez/sistemas_evolutivos/tree/main/Tareas/Tarea1

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:

$$f(x) = (x-2)(x-5) + \sin(1.5\pi x) \tag{1}$$

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).

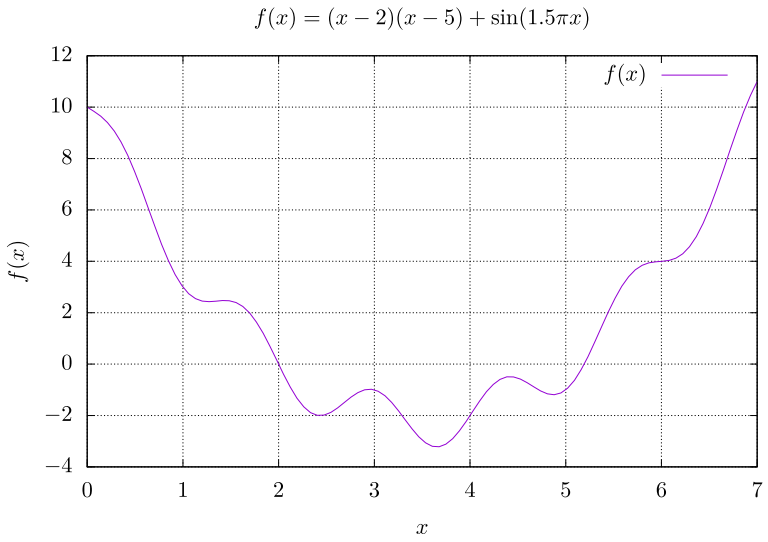

Figura 1: Grafica de la función a analizar.

$$f'(x) = 2x - 7 + 1.5\pi\cos(1.5\pi x) \tag{2}$$

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:

$$2x - 7 + 1.5\pi\cos(1.5\pi x) = 0 \tag{3}$$

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 algoritmos se muestra en el Código 1.

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

$$f''(x) = 2 - (1.5\pi)^2\sin(1.5\pi) \tag{4}$$

después generamos una solución aleatoria[1] (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

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

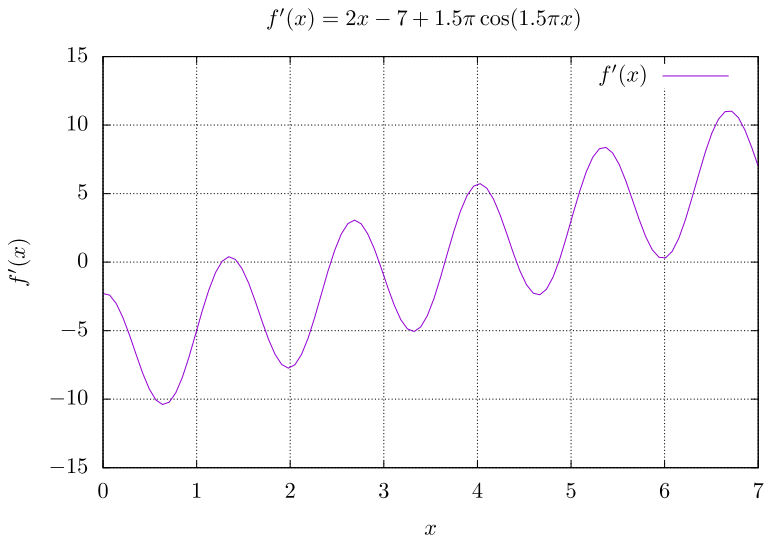

Figura 2: Grafica de la derivada de la función a analizar.

archivo de texto **datosfloat.txt** que contiene los puntos x donde existe un máximo o un mínimo, sin embargo 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 pueden hacer 1000 repeticiones y analizar los resultados fácilmente.

Finalmente podemos concluir que con los códigos que se muestran a continuación se encontraron los los 4 máximos y los 3 mínimos de la función $f(x)$ de los cuales solo uno fue el mínimo global. El método no es muy eficiente y se requieren muchas iteraciones para encontrar la solución correcta sin embargo su complejidad es poca y es en lo que cabe sencillo de implementar.

```python
# El metodo de Newton2
# Ciro Fabian Bermudez Marquez
# 02.02.2021
# Librerias
import math
import sys

# Funciones para el metodo
def df(x) :
  return 2.0*x - 7 + 1.5*math.pi*math.cos(1.5*math.pi*x)

def ddf(x) :
  return 2.0 - 1.5*math.pi*1.5*math.pi*math.sin(1.5*math.pi*x)

# El valor inicial de a
n = len(sys.argv)
if n != 2 :
  print( "Args: valor_inicial" )
  sys.exit(1)

a = float( sys.argv[1] )

# 50 iteraciones maximas
i = 1
#print("i \txi \t\tea")
while i <= 50 :
  Deltax = -df(a)/ddf(a)
  a += Deltax
  #print("%1d \t%3.5f \t%3.5f" %(i,a,Deltax))

  if abs(Deltax) < 1e-10 :
    break

  i += 1

#print( "x = ", a ,"df(x) = ", df(a) )
# Solucion
print(a)
```

Código 1: Método de Newton para máximos y mínimos.

```python
# Autor: Ciro Fabian Bermudez Marquez
# 02.02.2021

# Librerias
import sys
import random

n = len(sys.argv)
if n != 3 :
  print( "Args valor inicial" )
  sys.exit(1)

inicio = int( sys.argv[1] )
fin  = int( sys.argv[2] )

# Para int
#print( random.randint( inicio, fin ) )
# Para float
print( random.uniform( inicio, fin) )
```

Código 2: Generar número aleatorio flotante.

```bash
#! /bin/bash
# Autor: Ciro Fabian Bermudez Marquez
# 02.02.2021

for (( i=1; i<=100; i+=1))
do
  randnum=$(python3 randomnum.py 0 7)
  #echo $randnum
  #echo $i
  python3 newton2.py $randnum
done > datosfloat.txt
```

Código 3: Repeticiones.

```python
# Autor: Ciro Fabian Bermudez Marquez
# 02.02.2021

# Librerias
import math
import numpy as np
import collections

def df(x) :
  return 2.0*x - 7 + 1.5*math.pi*math.cos(1.5*math.pi*x)


def ddf(x) :
  return 2.0 - 1.5*math.pi*1.5*math.pi*math.sin(1.5*math.pi*x)


data = np.genfromtxt( "datosfloat.txt" )
tam = len(data)
xp = []
min_global = 3.6528874421621778

#cont = 0 # Apariciones minimo global
num_fallas = 0
num_neg = 0 # Apariciones de soluciones negativas
for i in range( tam ):
  is_zero =  df( data[i] )
  if  is_zero  < 1e-4 :   # La solucion es correcta
    if data[i] < 0 :
      num_neg += 1
    else :
      #print( i ,data[i] )
      xp.append( data[i] )
    #if abs( data[i] - min_global ) <= 1e-4 : # Contar minimo
    # cont += 1
  else:
    num_fallas += 1


xp = np.array(xp)        # Convertir a objeto de numpy
freq = collections.Counter(xp.round(12))

for key,value in freq.items() :
  if ddf( float(key) )  < 0 :
    s = "maximo"
  elif ddf( float(key) )  > 0 :
    s = "minimo"
  else :
    s = "ninguno"
  print(key,value,s)

freq["fallos"] = num_fallas
freq["<0"] = num_neg
print(freq)
print(num_fallas,num_neg)
```

Código 4: Análisis de los datos.

Referencias

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