Obtención de raíces
Método de bisección

Claudia Ballester Niebla, Carlos Herrera Carballo, Cathaysa Pérez Quintero

Grupo (1 | E)

Técnicas Experimentales. 1^{er} *curso.* 2^{do} *cuatrimestre*

Lenguajes y Sistemas Informáticos

Facultad de Matemáticas

Universidad de La Laguna

La Laguna, 9 de mayo de 2014

Resumen

A lo largo de este informe se mostrarán los experimentos realizados con el código, implementado en Python, con el fin de encontrar las raíces de una función, a través del método de bisección. Así, se podrán encontrar los pasos seguidos para demostrar la eficacia del programa y las conclusiones obtenidas de los mismos.
Será posible acceder al código implementado, y los datos recopilados en cada uno de los experimentos mencionados, además de los antecedentes teóricos, con sus demostraciones correspondientes, que han servido para demostrar el método utilizado para la obtención de raíces. El método de Bisección.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Motivación y objetivos

En la guía docente de la asignatura de Técnicas Experimentales de Primero de Grado en Matemáticas surge como una de las competencias básicas la realización de un experimento del que posteriormente se realizará un informe y una presentación. Para ello, los alumnos deben emplear las herramientas informáticas explicadas durante las clases.

De este modo, el alumno debe analizar, sintetizar, evaluar y describir los datos obtenidos del estudio de la función propuesta. Escoger un intervalo específico, realizar las operaciones correspondientes al método de bisección y obtener un resultado que será, posteriormente, evaluado y analizado cuantitativamente de forma experimental. Representar gráficamente los resultados obtenidos, sintetizándolos y exponiéndolos de forma objetiva. Utilizar herramientas informáticas, programando en un lenguaje relevante para el cálculo científico (python, LaTeX, beamer).

- **Objetivo principal**: Implementación con Python del método de bisección.
- **Objetivo específico**: Cómo se aproximan las raíces de una función, mediante el método de bisección.

1.1. Objetivo Principal

El objetivo principal es implementar el algoritmo del método de bisección en un intervalo [a,b], tal que $f(a) * f(b) < 0$:

1. Se toma $c = \frac{(b-a)}{2}$
2. Si $b - a \leq error$ se acepta c como la raíz y se para.
3. Si $f(b) * f(c) \leq 0$, se toma $a = c$, por el contrario hacer $b = c$.

1.2. Objetivo Específico

Mediante el método citado obtenemos la raíz única ($x = 1$) de la función:

$$f(x) = 5^x - 5$$

Capítulo 2

Fundamentos teóricos

El método de la bisección se basa en dos teoremas, el de Bolzano y el del Valor Intermedio que explicaremos a continuación, y es empleado para aproximar ceros de funciones.

Supongamos que queremos encontrar las raíces de una función $f(x)$ continua. Dados dos puntos a y b, tal que $f(a)$ y $f(b)$ tengan signos distintos, sabemos por el Teorema de Bolzano que $f(x)$ debe tener, al menos, una raíz en el intervalo [a,b]. Este método divide el intervalo en dos utilizando un tercer punto c= $\frac{a+b}{2}$. De esta forma, se darán dos posibilidades: $f(a)$ y $f(c)$, ó $f(c)$ y $f(b)$ tienen distinto signo. Se aplica este método al subintervalo donde ocurre el cambio de signo. Así se realizará tantas vecse como sea necesario para conseguir la máxima precisión.

2.1. Teorema de Bolzano

Sea $f(x)$ una función continua en un intervalo [a,b] tal que $f(a) * f(b) < 0$, entonces existe un punto c perteneciente al intervalo (a,b) tal que $f(c) = 0$

2.1.1. Demostración Teorema de Bolzano

Supongamos que $f(a) < 0$ y $f(b) > 0$. Sea A el conjunto formado por todos los valores x tal que x pertenece al intervalo [a,b] para los que $f(x) < 0$. El conjunto A está acotado superiormente por b, y además, no es vació ya que a pertenece a A. Por ello el conjunto A tiene un extremo superior c. Se cumple que $f(c) = 0$. Veámoslo:

Si $f(c) > 0$, entonces por la propiedad de la conservación del signo de las funciones continuas existiría un intervalo $(c - \alpha, c + \alpha)$ en el que la función sería también positiva. En este caso existirían valores menores que c que servirían de cota superior de A y por ello c no sería el extremo superior de A como hemos supuesto.

Si $f(c) < 0$, entonces existiría un intervalo $(c - \alpha, c + \alpha)$ en el que la función sería negativa y por tanto existían valores de x a la derecha de c para los que la función sería negativa y por tanto c no sería extremo superior de A. De este modo, $f(c)$ tiene que tomar el valor cero: $f(c) = 0$.

2.2. Teorema del Valor Intermedio

Sea $f(x)$ una función continua en un intervalo [a,b], tal que $f(a) < f(b)$ entonces, para todo k tal que $f(a) < k < f(b)$ existe x_0 que pertenece al intervalo (a,b) tal que $f(x_0) = k$.

2.2.1. Demostración Teorema del Valor Intermedio

Para la demostración aplicamos el Teorema de Bolzano en la función $g(x) = f(x) - k$, la cual es continua por serlo $f(x)$, $g(a) < 0$ y $g(b) > 0$. El teorema nos permite afirmar que existirá un c perteneciente al intervalo (a,b) tal que $g(c) = 0$ y en consecuencia $f(c) = k$.

2.3. Gráfica

En esta gráfica reflejamos los resultados teóricos esperados tras realizar los experimentos propuestos en el siguiente capítulo. Podemos observar como la función $f(x) = 5^x - 5$ tiene una raíz en el punto $(1, 0)$.

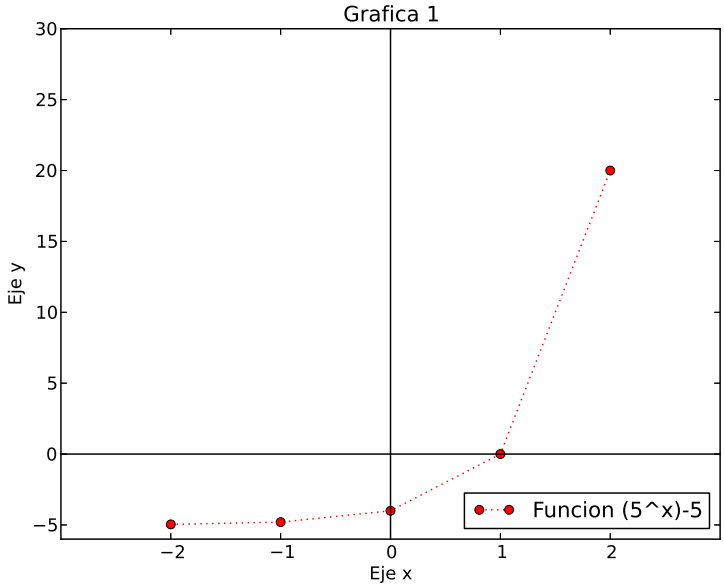

Figura 2.1: Gráfica de la función

Capítulo 3

Procedimiento experimental

Esta investigación consiste en la búsqueda de raíces mediante el método de bisección con un umbral de error determinado. A continuación, se describen los experimentos que se han llevado a cabo con el fin de verificar la hipótesis formulada anteriormente y contrastándola.

3.1. Descripción de los experimentos

3.1.1. Experimento 1

Con el fin de verificar la precisión del algoritmo propuesto, se ejecuta con los mismos valores (intervalo $(-2, 2)$ y umbral 0.1) y la misma función ($f(x) = 5^x - 5$) tres veces. De esta forma, si los resultados coinciden significará que es exacto.

- **Paso 1**: Se introducen en el terminal de la computadora el comando que permite llamar al intérprete de python para ejecutar el algoritmo solucion.py.
- **Paso 2**: A continuación de lo anterior, en el terminal, introducimos los valores -2, 2 y 0.1.
- **Paso 3**: Se ejecuta el programa.

Se repite este procedimiento cuantas veces sea necesario, en este caso han sido 3.

3.1.2. Experimento 2

A continuación, se ejecutará el algoritmo con distintas funciones para así demostrar que es válido para cualquier $f(x)$. Tomando el intervalo $(-2, 2)$ y un umbral de 0.1.

- **Paso 1**: Se cambia dentro de la función f del algoritmo solucion.py en la línea 5 la $f(x)$ que se quiera.
- **Paso 2**: Se introducen en el terminal de la computadora el comando que permite llamar al intérprete de python para ejecutar el algoritmo solucion.py.

- **Paso 3**: A continuación de lo anterior, en el terminal, introducimos los valores -2, 2 y 0.1.
- **Paso 4**: Se ejecuta el programa.

Se repite este procedimiento con las funciones que queramos.

3.1.3. Experimento 3

En este tercer experimento se ejecutará el programa con distintos intervalos para verificar la existencia de raíces en los mismos. Tomamos la función $f(x) = 5^x - 5$ y el umbral 0.1.

- **Paso 1**: Se introducen en el terminal de la computadora el comando que permite llamar al intérprete de python para ejecutar el algoritmo solucion.py.
- **Paso 2**: A continuación de lo anterior, en el terminal, introducimos los intervalos que se desee y el umbral 0.1.
- **Paso 3**: Se ejecuta el programa.

3.1.4. Experimento 4

Se observa cómo varían las soluciones obtenidas con distintos valores de tolerancia de error realizamos los siguientes pasos:

- **Paso 1**: Se introducen en el terminal de la computadora el comando que permite llamar al intérprete de python para ejecutar el algoritmo solucion.py.
- **Paso 2**: A continuación de lo anterior, en el terminal, introducimos el intervalo (-2,2) y los distintos umbrales que deseemos.
- **Paso 3**: Se ejecuta el programa.

3.1.5. Experimento 5

Para determinar el rendimiento del algoritmo, se introduce un cronómetro en el mismo de forma que imprimiera el tiempo que tarda la CPU de la computadora en ejecutar el programa.

3.2. Descripción del material

Para realizar los distintos experimentos propuestos se han utilizado como materiales:

- **Sistema Operativo**: Linux-3.2.0-61-generic-i686-with-Ubuntu-12.04-precise.
- **Procesador**: Genuine Intel(R) 2160.
- **Velocidad del procesador**: 1.80GHz.

- **Memoria del ordenador**: 1024 KB.
- **Lenguaje de programación**: Python.
- **Versión del compilador**: 2.7.3.

3.3. Resultados obtenidos

Intento	**Resultado**
1	0.969
2	0.969
3	0.969

Cuadro 3.1: Tabla experimento 1

Funcion	**Resultado**
5^x	Error
$5x - 10$	1.969
$3x^2 - 1$	Error

Cuadro 3.2: Tabla experimento 2

Intervalo	**Resultado**
(-3,2)	1.023
(-2,3)	1.008
(2,5)	Error

Cuadro 3.3: Tabla experimento 3

Tolerancia	**Resultado**
0.01	0.996
0.02	0.992
0.001	1.000

Cuadro 3.4: Tabla experimento 4

Intento	Tiempo (segundos)
1	$2{,}8133 * 10^{-}5$
2	$2{,}4080 * 10^{-}5$
3	$3{,}1948 * 10^{-}5$

Cuadro 3.5: Tabla experimento 5

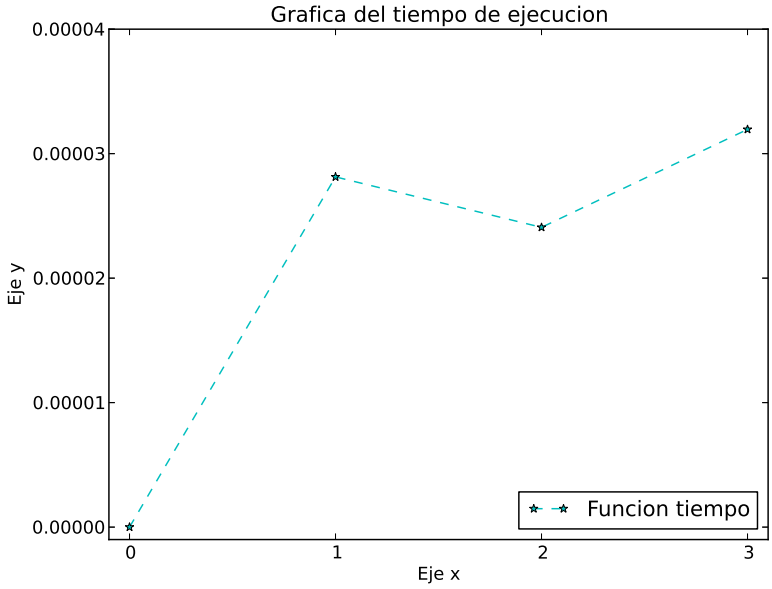

Figura 3.1: Gráfica del tiempo

3.4. Análisis de los resultados

3.4.1. Experimento 1

En este caso, se obtuvo el mismo resultado al ejecutar tres veces el algoritmo con los mismos parámetros, de esta forma se verifica que el programa implementado en Python es exacto. Los valores escogidos fueron: el intervalo (-2,2), el umbral 0.1 y la función $f(x) = 5^x - 5$, al aplicarlos se obtiene siempre que el valor aproximado de la raíz de dicha función es: 0.969.

3.4.2. Experimento 2

En este experimento, se ha realizado la prueba con tres funciones distintas. En primer lugar, con la función $f(x) = 5^x$, y el programa ha dado como salida un error, debido a que dicha función no posee ninguna raíz posible. En segundo lugar, se empleó la función

$f(x) = 5x - 10$, en este caso sí que se obtuvo una aproximación de la raíz: 1.969, con esto se verifica que el programa es válido para esta función. No obstante, en la última prueba realizada con la función $f(x) = 3x^2 - 1$, el programa vuelve a devolver un error, ya que dicha función poseé dos raíces posibles. Así, si se escoge un intervalo en el que sólo exista una raíz de f(x) sí la aproximará, de lo contrario, si están las dos en el mismo intervalo no funciona el algoritmo.

3.4.3. Experimento 3

En el tercer experimento se ejecutó el programa con distintos intervalos para ver la fiabilidad del algoritmo a la hora de aproximar la raíz de la función $f(x) = 5^x - 5$. Así, al poner el intervalo (-3,2) se obtuvo el resultado 1.023 y con el intervalo (-2,3) el resultado 1.008, de esta forma vemos como los valores de las raíces no varían excesivamente. Sin embargo, al introducir el intervalo (2,5) se produce un error ya que la función empleada no tiene una raíz en dicho intervalo.

3.4.4. Experimento 4

En esta ocasión, el experimento consistía en modificar el valor del umbral y así comprobar la precisión del programa. Asimismo, al introducir 0.01,0.02,0.001 se obtienen como resultados: 0.996,0.992 y 1.00 respectivamente. Como conclusión, se puede observar como el umbral más preciso es el 0.001 seguido de 0.01 y finalmente 0.02, por ello, cuanto menor sea el valor de la tolerancia del error mejor será la aproximación de la raíz de la función.

3.4.5. Experimento 5

Finalmente, este último experimento trataba de mostrar el rendimiento del programa. Para ello, primero ejecutamos el programa haciendo que nos pidiera los valores con los que se iba a ejecutar, en este caso el tiempo empleado fue 2,8133$*10^-5$ segundos. Seguidamente, introduciendo los valores mediante la consola sin que el programa nos los pidiera, el tiempo fue de $2{,}4080 * 10^-5$ segundos. Por último, introduciendo nuevamente los valores mediante la consola, lo que se llevó a cabo fue la sustitución del intervalo (-2,2) por (-10,10) en el algoritmo, con vistas de observar si aumentaba el tiempo de ejecución al probarlo con un intervalo mayor, efectivamente lo hizo, obteniéndose un tiempo de $3{,}1948 * 10^-5$.

Capítulo 4

Conclusiones

Tras haber realizado una serie de experimentos con el programa implementado, podemos afirmar que éste es eficaz para la función trabajada, pues, al utilizarse distintos intervalos, repitiendo los mismos, y eligiendo diferentes funciones, siempre podemos obtener una solución. Ésta consistirá en devolvernos si existe, o no, una raíz real. Es decir, si existe un punto de dicha función que corta al eje de abscisas.

Así, hemos conseguido un programa que nos trabaja con cualquier tipo de función y hemos probado, experimentalmente, cómo se aproximan las raíces y cómo funciona el método de bisección.

Sin embargo, para la función escogida, quizás podemos considerar como método más rápido, resolver la ecuación obtenida al igualar a cero nuestra función. Además de tener que considerar el hecho de que, si trabajamos con una función con más de una raíz, y elegimos un intervalo en el que se encuentran, también, más de una, nuestro programa dará error, puesto que solo funcionará para intervalos en los que encontremos un único punto de corte.

En definitiva, los experimentos llevados a cabo nos muestran una buena ejecución del programa para funciones similares a la escogida, o para polinomios de grado mayor o igual a dos, siempre que conozcamos intervalos en los que podamos encontrar cada una de las raíces; siempre recordardo que no han de ser intervalos diferentes, para que así nuestro programa no cause problema alguno, al ser ejecutado.

Apéndice A

Algoritmos de bisección y hardware y software

A.1. Algoritmo Metodo de Biseccion

```
#! /usr/bin/python
#!encoding: UTF-8
import time
import timeit

def f(x):
  return (5**x)-5
def biseccion(a,b,tol):
  c=(a+b)/2.0
  while((f(c)!=0.000001) and (abs(b-a)>tol)):
    if f(a)*f(c)<0.000001:
      b=c
    else:
      a=c
    c=(a+b)/2.0
  return c

import sys
if (len(sys.argv)==4):
  A=float(sys.argv[1])
  B=float(sys.argv[2])
  TOL=float(sys.argv[3])
else:
  A=float(raw_input("Introduzca el extremo a del intervalo: "))
  B=float(raw_input("Introduzca el extremo b del intervalo: "))
  TOL=float(raw_input("Introduzca la tolerancia del error que desee: "))
if f(A)*f(B)<0.000001:
  start=time.time()
  raiz=biseccion(A,B,TOL)
  finish=time.time()-start
  print "El tiempo de ejecucion es:"
  print finish
```

```
  print "La raiz aproximada de la funcion escogida es: %4.3f" %raiz
else:
  print "En ese intervalo no existe raÃz, por favor vuelva a ejecutar el programa con otros valores"
```

A.2. Algoritmo Hardware y Software

```
import os
import platform

def SOFTinfo():
  softinfo={}
  softinfo={'Several': platform.uname() , 'S.O':platform.platform(), 'Pythons Version': platform.python_vers
  return softinfo

def CPUinfo():
# infofile on Linux machines:
  infofile = '/proc/cpuinfo'
  cpuinfo = {}
  if os.path.isfile(infofile):
    f = open(infofile, 'r')
    for line in f:
      try:
name, value = [w.strip() for w in line.split(':')]
      except:
continue
      if name == 'model name':
cpuinfo['CPU type'] = value
      elif name == 'cache size':
cpuinfo['cache size'] = value
      elif name == 'cpu MHz':
cpuinfo['CPU speed'] = value + 'Hz'
      elif name == 'vendor_id':
cpuinfo['vendor ID'] = value
    f.close()
  return cpuinfo

if __name__ == '__main__':

  softinfo=SOFTinfo()
  for keys in softinfo.keys():
    print softinfo[keys]

  cpuinfo=CPUinfo()
  for keys in cpuinfo.keys():
    print CPUinfo()

  print "Introduzca el nombre del fichero para almacenar los resultados:"
  nombre_fichero= raw_input();
  f=open(nombre_fichero, "w")

  for keys in softinfo.keys():
    if type (softinfo[keys]) is list:
```

```
      f.write('\n'.join(softinfo[keys]))
    else:
      f.write(str(softinfo[keys]))
      f.write('\n')

for keys in cpuinfo.keys():
  if type (cpuinfo[keys]) is list:
    f.write('\n'.join(cpuinfo[keys]))
  else:
    f.write(str(cpuinfo[keys]))
    f.write('\n')
f.close()
```

Apéndice B

Algoritmos de gráficas

B.1. Algoritmo Grafica de la funcion

```
#! encoding: UTF-8
import matplotlib.pyplot as pl
x=[-2.0,-1.0,0.0,1.0,2.0]
y=[-4.96,-4.8,-4,0,20]
x1=[-3,3]
y1=[0,0]
x2=[0,0]
y2=[-6,30]
pl.title('Grafica 1')
pl.xlabel('Eje x')
pl.ylabel('Eje y')
pl.plot (x,y, marker='o',linestyle=':',color='r', label='Funcion (5^x)-5')
pl.plot (x1,y1,color='k')
pl.plot (x2,y2,color='k')
pl.legend(loc=4)
pl.xlim(-3.0, 3.0)
pl.ylim(-6.0, 30.0)
pl.xticks(x)
pl.savefig("grafcap2.eps", dpi=100)
pl.show()
```

B.2. Algoritmo Grafica del tiempo

```
#! encoding: UTF-8
import matplotlib.pyplot as pl
x=[0,1,2,3]
y=[0.0,0.000028133,0.000024080,0.000031948]
pl.title('Grafica del tiempo de ejecucion')
pl.xlabel('Eje x')
pl.ylabel('Eje y')
pl.plot (x,y, marker='*',linestyle='--',color='c', label='Funcion tiempo')
pl.legend(loc=4)
pl.xlim(-0.1,3.1)
pl.ylim(-0.000001,0.00004)
```

```
pl.xticks(x)
pl.savefig("graftime.eps", dpi=100)
pl.show()
```

Bibliografía

[Guía Docente, 2013] Guía docente de la asignatura: Técnicas Experimentales. (2013)

$http://eguia.ull.es/matematicas/query.php?codigo = 299341201$

[Cálculo Infinitesimal] Spivak, M.-Calculus, Ed. Cambridge, 2006; Ed. Reverté, 1981 [BULL]

[Método de Bisección] Demostración del Método de Bisección. (2008)

$http://www.ma3.upc.edu/users/carmona/teaching/clases/08-09/trabajos/$

[Tutorial de Python] Python para todos. -Raúl González Duque.

$http://campusvirtual.ull.es/1314/pluginfile.php/197675/mod_resource/content/4/Primera_Parte/General20$

[Tutorial de LaTeX] The beamer class. User Guide for version 3.26.

$http://campusvirtual.ull.es/1314/pluginfile.php/197674/mod_resource/content/1/beameruserguide.pdf$