Búsqueda de raíces mediante el método de Newton - Raphson

Función $f(x) = cos(\pi x)$

Alba Crespo Pérez, Raquel Espino Mantas y Robbert Jozef Michiels

Grupo 1

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

Lenguajes y Sistemas Informáticos

Facultad de Matemáticas

Universidad de La Laguna

La Laguna, 10 de mayo de 2014

Índice general

1. Motivación y objetivos **1**

2. Fundamentos teóricos **2**
2.1. Breve introducción histórica 2
2.2. Descripción teórica del método 2
2.3. Análisis de la función de estudio 3

3. Procedimiento experimental **5**
3.1. Descripción de los experimentos 5
3.2. Descripción del material 5
3.3. Resultados obtenidos 6
3.4. Análisis de los resultados 6

4. Conclusiones **7**

A. Algoritmo para el cálculo de raíces **9**

B. Título del Apéndice 2 **11**
B.1. Recuento de iteraciones 11
B.2. Representacion grafica de la evolucion de las iteraciones 12

Bibliografía **13**

Índice de figuras

Índice de cuadros

Capítulo 1

Motivación y objetivos

El objetivo principal del presente trabajo reside en la implementación, mediante el uso del lenguaje de programación Python, de un algoritmo basado en el método de Newton - Raphson que nos permita llevar a cabo el cálculo de las raíces de la función $f(x) = cos(\pi x)$.

Además, se pretende lograr la consecución de los siguientes objetivos específicos:

- Efectuar un análisis numérico y gráfico de la evolución en el número de iteraciones requeridas por este método para la obtención de una raíz, en función del error absoluto de la estimación inicial tomada como punto de partida del método respecto a la solución real determinada tras su aplicación.
- Analizar el costo computacional, en términos de tiempo de uso de CPU, asociado a la ejecución del algoritmo implementado, así como su variabilidad y sensibilidad ante modificaciones en los parámetros iniciales.

Capítulo 2

Fundamentos teóricos

2.1. Breve introducción histórica

Este método constituye una vía algorítmica de análisis numérico destinada a la determinación de los ceros o raíces de una función matemática dada. Su primera descripción relevante fue desarrollada por el multidisciplinar científico inglés Isaac Newton en su obra *De analysi per aequationes numero terminorum infinitas*, escrita en 1669 y publicada en 1711 por William Jones, así como en su tratado *De metodis fluxionum et serierum infinitarum*, editado en 1736 por John Colson bajo el título de *Método de las fluxiones*.

Sin embargo, si bien su reconocimiento no alcanzó en su momento la repercusión otorgada a los trabajos de de Newton, dicho método encuentra mención previa en el libro *Aequationum Universalis* (1690), cuya publicación conllevó para su autor, el matemático inglés Joseph Raphson, la consecución del ingreso en la *Royal Society* de Londres.

2.2. Descripción teórica del método

El método de Newton-Raphson presenta un carácter abierto, lo cual implica que su convergencia global no se encuentra garantinzada en todos sus posibles contextos de aplicación. Como condición indispensable para alcanzar la convergencia, es preciso seleccionar a modo de parámetro de partida un valor inicial suficientemente cercano a la raíz buscada, que progresará acercándose al valor real de esta raíz a medida que avance la ejecución del algoritmo; de esta forma, y según comentaremos en mayor profundidad más adelante, sus probabilidades de divergencia se incrementan para el caso de funciones con múltiples puntos de inflexión o pendientes pronunciadas en el entorno de corte con la abscisa.

Una vez fijada una estimación inicial x_0 apropiada para comenzar la aplicación del método, calcularemos la expresión de la recta tangente a la gráfica de la función dada en el punto $x = x_0$. La coordenada de corte de dicha recta con el eje horizontal será considerada de modo genérico como un valor más cercano a la raíz buscada que la hipótesis original primeramente supuesta, por lo que este paso podrá aplicarse de forma recursiva tomando

como suposición al nuevo valor obtenido, hasta lograr una aproximación aceptable de la raíz dentro de un margen de tolerancia fijado de antemano de acuerdo con la precisión exigida por el contexto.

Profundizaremos ahora de forma más explícita en los cálculos matemáticos necesarios para la obtención de una expresión recursiva del algoritmo. Sabemos que la expresión de una recta de pendiente m y que pasa por el punto (a, b) viene dada de la forma:

$$y - b = m(x - a)$$

En consecuencia, y considerando que la pendiente de la recta tangente a una función $f(x)$ en un punto $x = x_0$ corresponde al valor en dicho punto de su primera derivada, obtenemos la siguiente expresión para dicha recta en la denominada *forma punto-pendiente*:

$$y - f(x_0) = f'(x_0)(x - x_0)$$

Tomando $y = 0$, procedemos a determinar el punto de corte de la recta anterior con el eje de abscisas. La notación x_{n+1} indica que el valor resultante de x corresponderá a la estimación tomada como partida para la siguiente ejecución del algoritmo:

$$y = 0 \quad \rightarrow \quad -f(x_0) = f'(x_0)(x_{n+1} - x_0) \quad \rightarrow \quad x_0 f'(x_0) - f(x_0) = x_{n+1} f'(x_0) \quad \rightarrow$$

$$\rightarrow \quad x_{n+1} = x_0 - \frac{f(x_0)}{f'(x_0)}$$

De acuerdo con lo comentado al inicio de la sección, observamos que las funciones con acentuadas variaciones de pendiente en el entorno de la raíz buscada dificultarán la convergencia del algoritmo, por lo que al llevar a cabo su implementación computacional será preciso establecer una cota máxima de iteraciones con el fin de evitar que la ejecución recursiva permanezca atrapada en un patrón de divergencia.

2.3. Análisis de la función de estudio

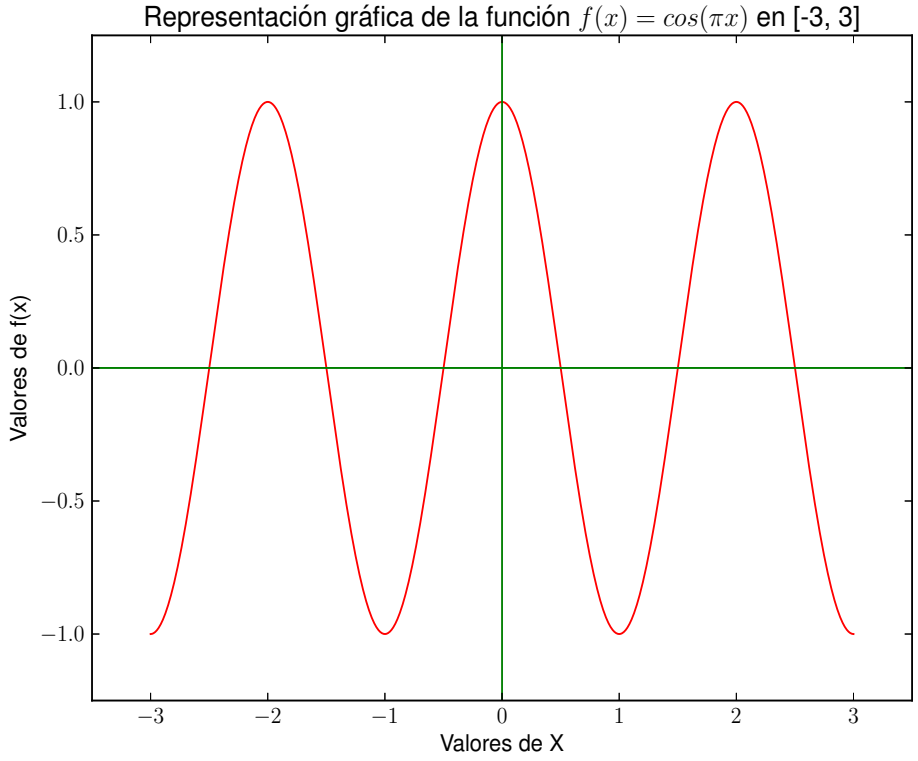
Con el objetivo de lograr una mejor comprensión de los resultados obtenidos en la aproximación de las raíces de la función $f(x) = cos(\pi x)$ mediante el algoritmo de Newton-Raphson, procedemos a calcular dichas raíces de modo teórico, mediante la resolución de una sencilla ecuación trigonométrica:

$$cos(\pi x) = 0 \quad \rightarrow \quad \pi x = \frac{\pi}{2} + k\pi, k \in \mathbb{Z} \quad \rightarrow x = \frac{1}{2} + k\pi, k \in \mathbb{Z}$$

Vemos así que se verifica lo siguiente:

$$cos(\pi x) = 0 \iff x = ..., -2{,}5, -1, 5, -0{,}5, 0{,}5, 1{,}5, 2{,}5, ...$$

Concluimos con ello que se trata de una función periódica con periodo P = 2. Este hecho, así como los puntos de corte con el eje de abscisas, quedan reflejados de modo ilustrativo a través de la siguiente gráfica:



Capítulo 3

Procedimiento experimental

3.1. Descripción de los experimentos

Para llevar a cabo el cálculo de las raíces de la función objeto de estudio, se ha realizado la implementación de un algoritmo recursivo basado en el método de Newton-Raphson (ver apéndice A). Ejecutando en repetidas ocasiones el fichero resultante y utilizando distintos valores como parámetros iniciales del método, ha sido posible observar los efectos producidos por la variabilidad de éstos sobre la solución final generada por el algoritmo.

Por otra parte, con el fin de cumplir los objetivos específicos de esta investigación (ver capítulo 1), hemos recurrido a la elaboración de dos programas Python: el primero de ellos (apéndice 2A) está destinado a efectuar un recuento de las iteraciones requeridas para el cálculo de una raíz en función del error absoluto del valor proporcionado al método como estimación inicial, almacenando las cantidades obtenidas en un fichero para su posterior lectura, mientras que el segundo (apéndice 2B) posibilita la elaboración de una gráfica ilustrativa tomando como base la información contenida en el fichero generado con anterioridad.

3.2. Descripción del material

Como soporte físico requerido para la puesta en práctica de los experimentos, ha sido utilizada una computadora con las especificaciones de hardware que se detallan a continuación:

- **Tipo de CPU:** Pentium(R) Dual-Core CPU, GenuineIntel, T4500.
- **Velocidad de la CPU:** 1200 Hz.
- **Tamaño del caché:** 1024 KB.

- **Memoria RAM:** 8 GB.

En lo referente a las características y versiones del software empleado, cabe señalar la siguente información:

- **Versión de Python:** 2.7.3.
- **Compilador Python:** GCC 4.7.2.
- **Sistema operativo:** Linux-3.5.0-17-genérico con Ubuntu-12.10-quantal.
- **Fecha de creación de la versión de Python:** Sep 26 2012 21:53:58.

3.3. Resultados obtenidos

Subsección 3

3.4. Análisis de los resultados

Subsección 4

Capítulo 4

Conclusiones

Contenido del capÃtulo 4

Apéndice A

Algoritmo para el cálculo de raíces

```
##############################################################################################
# Fichero newton.py
##############################################################################################
#
#  AUTORES: Alba Crespo Perez, Raquel Espino Mantas y Robbert Jozef Michiels
#
#  FECHA: 5 de mayo de 2014
#
#  DESCRIPCION: Este codigo Python nos permite calcular las raices de la funcion
#  f(x) = cos (pi * x), mediante la aplicacion del metodo de Newton-Raphson. Como
#  parametros de inicio se solicita al usuario una estimacion de la raiz, el margen
#  de tolerancia permitido y una cota maxima de iteraciones.
#
##############################################################################################

#!encoding: UTF-8

from math import cos
from math import sin
PI = 3.141592653589793116

def f(x):
    return cos (PI * x)

def df(x):
    return - PI * sin (PI * x)

def newton (g, tol, nmax, it):
    if (it < nmax):
        if (df(g) != 0):
            g = g - (f(g)/df(g))
        else:
            return 1e7
        if (abs(f(g)) > tol):
            g = newton (g, tol, nmax, it)
            it = it + 1
```

```
        if (abs(f(g)) < tol):
            return g
    else:
        return 1e6

g = float(raw_input("\nProporcione una estimacion para iniciar el calculo: "))
tol = float(raw_input("Introduzca el margen de tolerancia: "))
nmax = int(raw_input("Indique la cantidad maxima de iteraciones: "))
it = 0
sol = newton (g, tol, nmax, it)
if (sol == 1e6):
    print "\n\tLo sentimos, no hemos localizado ninguna raiz \n\ttras alcanzar el maximo
            de iteraciones permitidas."
    print "\nIntentelo de nuevo proporcionando una mejor estimacion como inicio del mÃ©todo,\n
            o bien incrementando la cota de iteraciones.\n"
elif (sol == 1e7):
    print "\n\tHemos alcanzado una anulacion de la derivada durante la ejecucion, \n\tpor lo
           que el metodo no es aplicable para los valores aportados.\n"
    print "Intentelo de nuevo modificando los parametros iniciales.\n"
else:
    print "\nRaiz encontrada para la funcion:", sol, "\n"
```

Apéndice B

Título del Apéndice 2

B.1. Recuento de iteraciones

```
##############################################################################################
# Fichero countiter.py
##############################################################################################
#
#  AUTORES: Alba Crespo Perez, Raquel Espino Mantas y Robbert Jozef Michiels
#
#  FECHA: 5 de mayo de 2014
#
#  DESCRIPCION: El codigo Python que a continuacion se presenta cumple la funcion
#  de determinar la cantidad de iteraciones requeridas para la deteccion de la raiz
#  x = 0.5, tomando como estimacion inicial a los distintos valores comprendidos en
#  el intervalo [0.2, 0.5), con una diferencia de una centesima entre dos valores
#  consecutivos. Los resultados finales seran almacenados en un fichero.
#
##############################################################################################

#!encoding: UTF-8

from math import cos
from math import sin
PI = 3.141592653589793116

def f(x):
    return cos (PI * x)

def df(x):
    return - PI * sin (PI * x)

def newton (g, tol, nmax, it, name):
    if (it < nmax):
        if (df(g) != 0):
            g = g - (f(g)/df(g))
        else:
```

```
            return 1e7
        if (abs(f(g)) > tol):
            g = newton (g, tol, nmax, it, name)
            it = it + 1
        if (abs(f(g)) < tol):
            fich = open(name, "a")
            fich.write(str(it) + "\n")
            fich.close()
            return g
    else:
        return 1e6

tol = float(raw_input("\nIntroduzca el margen de tolerancia: "))
nmax = int(raw_input("Indique la cantidad maxima de iteraciones: "))
name = raw_input("Especifique un nombre para el fichero de salida: ")

for i in range (20, 50):
    x = i/100.0
    sol = newton (x, tol, nmax, 0, name)
    if (sol != 1e6) and (sol != 1e7):
        fich = open(name, "a")
        fich.write (str(x) + "\n")
        fich.close()

print "\nOperacion realizada con exito.\n"
```

B.2. Representacion grafica de la evolucion de las iteraciones

```
##############################################################################################
# Fichero graphiter.py
##############################################################################################
#
#  AUTORES: Alba Crespo Perez, Raquel Espino Mantas y Robbert Jozef Michiels
#
#  FECHA: 5 de mayo de 2014
#
#  DESCRIPCION: A partir de la lectura e interpretacion de la informacion contenida
#  en el fichero generado por el programa anterior, este codigo nos permite elaborar
#  una representacion grafica de los datos experimentales obtenidos para asi facilitar
#  su comprension y visualizacion.
#
##############################################################################################

#!encoding: UTF-8

import matplotlib.pyplot as pl
pl.rc('text', usetex=True)
pl.rc('font', family='Bookman')
```

```
name = raw_input("Introduzca el nombre del fichero para lectura: ")
f = open(name, "r")

flag = 0
X = []
Y = []

while (True):
    l = f.readline().rstrip()
    if (l == ""):
        break
    elif (l != "0") and (l != "1"):
        Y = Y + [flag-1]
        flag = 0
    else:
        flag += 1

f.close()

for i in range (20, 50):
    x0 = i/100.0
    dif = 0.5 - x0
    X = X + [dif]

pl.plot(X,Y,"b")
pl.title(r'Evolucion en el numero de iteraciones del metodo de Newton')
pl.xlabel(r'Error absoluto de la estimacion inicial')
pl.ylabel('Cantidad de iteraciones')

pl.xlim(-0.05, 0.35)
pl.ylim(0.8, 3.2)

pl.savefig("iterevol.eps", dpi=72)
pl.show()
```

Bibliografía

[1] Anita de Waard. A pragmatic structure for research articles. In *Proceedings of the 2nd international conference on Pragmatic web*, ICPW '07, pages 83–89, New York, NY, USA, 2007. ACM.

[2] J. Gibaldi and Modern Language Association of America. *MLA handbook for writers of research papers*. Writing guides. Reference. Modern Language Association of America, 2009.

[3] G.D. Gopen and J.A. Swan. The Science of Scientific Writing. *American Scientist*, 78(6):550–558, 1990.

[4] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison–Wesley Pub. Co., Reading, MA, 1986.

[5] Coromoto León. *Diseño e implementación de lenguajes orientados al modelo PRAM*. PhD thesis, 1996.

[6] Guido Rossum. Python library reference. Technical report, Amsterdam, The Netherlands, The Netherlands, 1995.

[7] Guido Rossum. Python reference manual. Technical report, Amsterdam, The Netherlands, The Netherlands, 1995.

[8] Guido Rossum. Python tutorial. Technical report, Amsterdam, The Netherlands, The Netherlands, 1995.

[9] ACM LaTeX Style. http://www.acm.org/publications/latex_style/.