Universidad
de La Laguna

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, 11 de mayo de 2014

Índice general

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:

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

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:

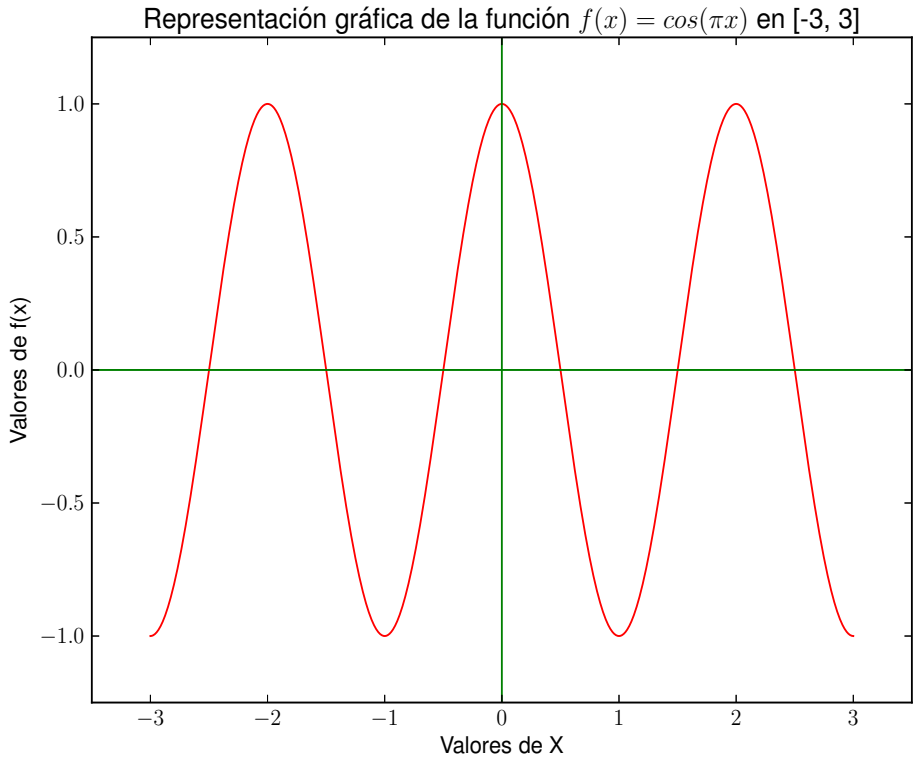

Figura 2.1: $f(x) = cos(\pi x)$

Debido a la naturaleza del método de Newton-Raphson, es preciso señalar que su aplicación no podrá llevarse a cabo en caso de alcanzarse una anulación de la derivada para el punto de partida del algoritmo. Analizaremos por tanto los puntos críticos de la función de estudio, para así conocer qué valores deben ser evitados como suposiciones iniciales en la invocación del método:

$$(cos(\pi x))' = 0 \quad \rightarrow \quad -\pi sen(\pi x) = 0 \quad \rightarrow \quad sen(\pi x) = 0 \quad \rightarrow \quad \pi x = k\pi, \ k \in \mathbb{Z} \quad \rightarrow$$
$$\rightarrow \quad x = k, \ k \in \mathbb{Z} \quad \rightarrow x = ..., -2, -1, 1, 2, ...$$

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 tres programas Python: el primero de ellos (apéndice B1) 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; el segundo (apéndice B2) posibilita la elaboración de una gráfica ilustrativa tomando como base la información contenida en el fichero generado con anterioridad, y el tercero (apéndice B3) cumple la función de generar una representación gráfica del tiempo total de CPU empleado en la ejecución del algoritmo, frente a la cantidad de iteraciones recursivas efectuadas por el mismo.

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

La ejecución del algoritmo implementado (ver apéndice A), basado en el método de Newton-Raphson, para el cálculo de las raíces de la función $f(x) = cos(\pi x)$ ha proporcionado como resultado la siguiente tabla ejemplificativa de raíces, tomando como punto de partida los valores supuestos que en ella se especifican:

INICIO	TOLERANCIA	COTA ITERAC.	ITERAC. REALES	RESULTADO
-7.08	10^{-7}	100	3	-8.5
-7.1	10^{-7}	100	4	-9.5
-7.2	10^{-7}	100	3	-7.5
-5.8	10^{-5}	50	2	-5.49999728877
-5.8	10^{-6}	50	3	-5.5
-0.23	10^{-7}	30	3	-0.5
-0.23	10^{-9}	30	3	-0.5
1.24	10^{-6}	20	2	1.50000001507
1.24	10^{-8}	20	3	1.5
2.17	10^{-12}	10	4	2.5
3.45	10^{-4}	10	1	3.49999999976
3.45	10^{-10}	10	2	3.5

Cuadro 3.1: Resultados experimentales del algoritmo de Newton-Raphson

En lo referente al análisis de la evolución en la cantidad de iteraciones para distintos errores absolutos o márgenes de desviación en la estimación inicial respecto a la raíz buscada, los resultados obtenidos quedan reflejados a través del siguiente gráfico, basado

en la tabla que a seguidamente se presenta y correspondientes a un margen de tolerancia de 10^{-9}:

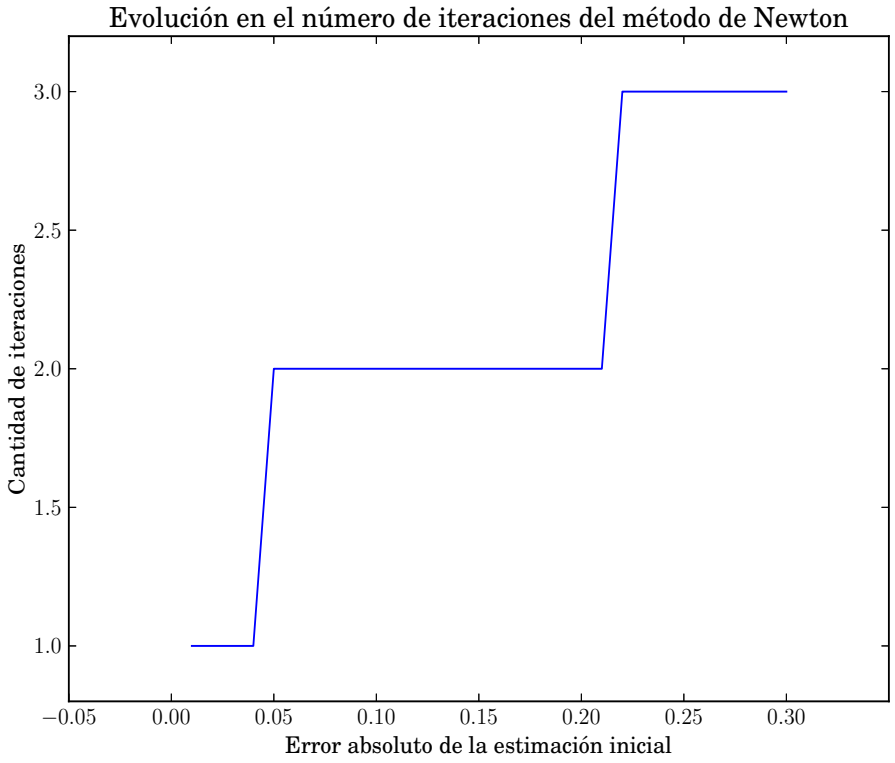

Figura 3.1: Análisis de la tendencia al incremento en las iteraciones.

VALOR INICIAL	ERROR ABSOLUTO	ITERACIONES
Desde 0.2 hasta 0.28	$e \in [0{,}22, 0{,}3]$	3
Desde 0.29 hasta 0.45	$e \in [0{,}05, 0{,}21]$	2
Desde 0.46 hasta 0.49	$e \in [0{,}01, 0{,}04]$	1

Cuadro 3.2: Iteraciones requeridas para determinar $x_0 = 0{,}5$

Por último, y siguiendo el esquema anterior, la información relativa al tiempo de CPU empleado para distintas cantidades de iteraciones puede consultarse a través del gráfico que a continuación se muestra, así como a partir de su tabla correspondiente:

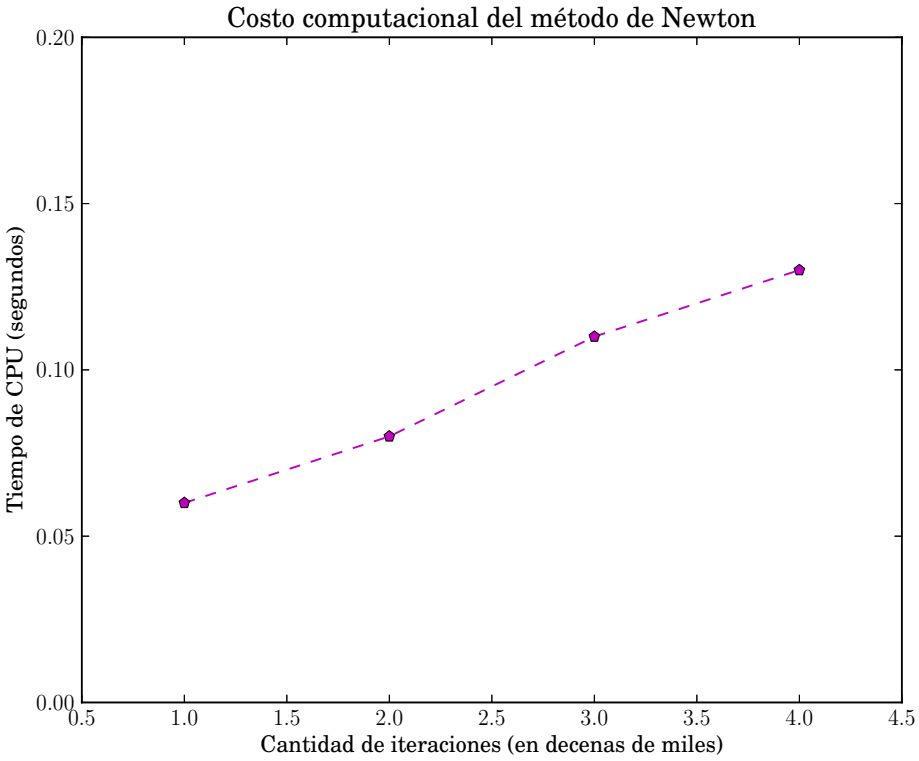

Figura 3.2: Avance del tiempo de CPU por cantidad de iteraciones.

ITERACIONES	TIEMPO DE CPU
10^4	0.06 s
$2 \cdot 10^4$	0.078 s
$3 \cdot 10^4$	0.11 s
$4 \cdot 10^4$	0.13 s

Cuadro 3.3: Tiempo de CPU frente al volumen de iteraciones

3.4. Análisis de los resultados

De acuerdo con información recogida en la tabla 3.1, es posible comentar diversos aspectos relevantes. Por una parte, sus dos primeras filas reflejan un proceso de ejecución cuya iteración inicial diverge respecto a la raíz más cercana al valor supuesto, debido a la proximidad del mismo a un extremo relativo de la función, lo cual origina una recta

tangente de pendiente reducida cuyo punto de corte con los ejes se encuentra notablemente alejado del valor de partida. Sin embargo, observamos que a partir de este momento la ejecución recursiva se torna convergente, debido a que dicho punto de corte constituye un valor supuesto próximo a otra de las raíces de la función estudiada; en este sentido, podemos concluir que el método de Newton-Raphson logrará la convergencia para la práctica totalidad de los posibles valores iniciales, debido a las infinitas raíces que la función presenta y a su característica de periodicidad, si bien la solución devuelta por el método podría no resultar la más próxima a la suposición inicial en caso de encontrarse ésta cercana a un extremo relativo de la función.

Por otra parte cabe destacar que, de modo general, la cantidad de iteraciones requerida por este método para la obtención de una solución válida resulta considerablemente reducida, no superando las cuatro iteraciones aún en el caso de seleccionar estrechos márgenes de tolerancia, del orden de 10^{-9}; de acuerdo con ello, podemos afirmar que el método de Newton-Raphson presenta en este caso una elevada velocidad de convergencia. Debido a este hecho, el valor fijado para la cota máxima de iteraciones reviste escasa importancia, siendo factible reducirlo hasta cantidades incluso inferiores a las diez iteraciones.

En lo referente al incremento en la cantidad de iteraciones en función del aumento en el error absoluto de la estimación inicial respecto a la raíz buscada, los datos recogidos en el gráfico 3.1 y la tabla 3.2 corroboran lo ya observado con anterioridad: la rápida convergencia del algoritmo permite obtener un suave ascenso en el número de ejecuciones recursivas, que tal y como ha sido expuesto, no llegan en ningún caso a sobrepasar el margen de cuatro iteraciones. De cualquier modo, la separación constante de una unidad existente entre raíces imposibilita el análisis del comportamiento del algoritmo para errores absolutos superiores a 0.3 en la estimación inicial, a riesgo de comprometer la convergencia, producir una anulación de la derivada u obtener raíces alejadas del entorno de partida.

Por último, la información relativa al costo computacional del algoritmo en términos de tiempo de CPU requerido para efectuar su ejecución (véase figura 3.2 y tabla 3.3) nos permite corroborar la elevada eficiencia de la implementación recursiva del método de Newton-Raphson, puesto que la reducida cantidad de iteraciones que requiere hace preciso reiterar decenas de miles de veces la invocación al algoritmo hasta obtener valores significativos para los tiempos de uso de CPU. Además, observamos que la variabilidad en los parámetros iniciales tan sólo afecta a la utilización de recursos computacionales cuando su modificación altera el volumen total de iteraciones requerido para la ejecución del algoritmo.

Capítulo 4

Conclusiones

En esta sección se facilita un resumen de las principales conclusiones alcanzadas tras la finalización y análisis de los experimentos llevados a cabo durante la realización del presente trabajo:

1. Las raíces de la función $f(x) = cos(\pi x)$ vienen dadas de la siguiente forma:

$$x = \frac{1}{2} + k,\ k \in \mathbb{Z} \iff x = ..., -2{,}5, -1, 5, -0{,}5, 0{,}5, 1{,}5, 2{,}5, ...$$

2. Para el caso de la función de estudio, la convergencia del algoritmo de Newton-Raphson se encuentra prácticamente garantizada debido a la reducida separación entre raÃces y a su carÃ¡cter de periodicidad, salvo en el caso de seleccionar un valor inicial coincidente con un extremo relativo de la función.

3. La eficiencia del algoritmo implementado resulta ser considerablemente elevada, al requerir ínfimos tiempos de CPU para la determinación de una raíz y no superar en ningún caso el margen de cuatro iteraciones hasta obtener la solución exacta.

4. El método de Newton-Raphson demuestra caracterizarse por una notable velocidad de convergencia para la función considerada, llegando a proporcionar hasta cinco cifras decimales correctas a lo largo de de una única iteración.

5. Las alteraciones en los parámetros iniciales tan sólo afectan al tiempo de uso de CPU a través de la modificación de la cantidad total de invocaciones recursivas requeridas para alcanzar una raíz válida, si bien el reducido valor de dicho tiempo origina que las diferencias globales resulten despreciables tanto desde el punto de vista del usuario como en un sentido computacional.

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

Algoritmos para la evaluación de la eficiencia

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. Creando figura con evolución 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()
```

B.3. Estudio del tiempo de CPU

```
##############################################################################################
# Fichero CPUtime.py
##############################################################################################
#
#  AUTORES: Alba Crespo Perez, Raquel Espino Mantas y Robbert Jozef Michiels
#
#  FECHA: 5 de mayo de 2014
#
#  DESCRIPCION: Este programa cumple la funcion de representar graficamente el tiempo
#  de CPU requerido en la ejecucion de 10^4 invocaciones al metodo de Newton-Raphson
#  tomando para ello cuatro posibles combinaciones de parametros iniciales que
#  precisan, respectivamente, de 1, 2, 3 y 4 llamadas recursivas al algoritmo.
#
```

```
##############################################################################################

#!encoding: UTF-8

import matplotlib.pylab as pl
import numpy as np
import newton
import time

pl.rc('text', usetex=True)
pl.rc('font', family='Bookman')

g = (3.45,-5.8,-7.2,2.17)
tol = (10e-4,10e-5, 10e-7,10e-12)
nmax = (10,50,100,10)

X = []
Y = []

for i in range (0, 4):
    t0 = time.clock()
    for j in range (0, 10000):
        sol = newton.newton (g[i], tol[i], nmax[i], 0)
    tf = time.clock() - t0
    X = X + [i+1]
    Y = Y + [tf]

print Y

pl.plot(X,Y,'mp--')
pl.title(r'Costo computacional del m\'etodo de Newton')
pl.xlabel(r'Cantidad de iteraciones (en decenas de miles)')
pl.ylabel(r'Tiempo de CPU (segundos)')

pl.xlim(0.5, 4.5)
pl.ylim(0, 0.2)

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

Bibliografía

[1] Steven C. Chapra. *Métodos numéricos para ingenieros*. McGraw Hill - Interamericana de México, México, 2007.

[2] Evolución histórica del método de Newton y calculadora virtual de raíces. http://www.uam.es/personal_pdi/ciencias/barcelo/cnumerico/recursos/newton.html.

[3] Alex Martelli. *Python: guía de referencia*. Anaya, Madrid, España, 2003.

[4] A. A. Samarski. *Introducción a los métodos numéricos*. MIR Editorial, Moscú, Rusia, 1986.