Título

Pedro Bonilla Nadal

14 de marzo de 2018

1. Ejemplos.

Busca tres ecuaciones no lineales diferentes $f(x) = 0$, verificando $|f'(x^*)| \simeq 1$, $|f'(x^*)| >> 1$, $|f'(x^*)| << 1$, respectivamente, de las que puedas obtener una solución exacta y un intervalo para cada una de ellas que contenga una única solución. Aproxima la misma en cada caso con el método de bisección fijando un error máximo y analiza la diferencia entre el número de iteraciones realizadas con cada una de las variantes *a), b), c)* y *d)* del método.

Solución.

- $f_1(x) = e^x - 1 \implies f_1(0) = 0,\ f_1'(0) = 1 \simeq 1$

```
Introduce el primer valor del intervalo: -1
Introduce el segundo valor del intervalo: 2
Introduzca la funcion de la que quiere hallar una raiz en
el intervalo anterior:
f(x) := np.exp(x)-1

Criterio 1:
0.5
-0.25
0.125
-0.0625
0.03125
-0.015625
0.0078125
-0.00390625
0.001953125
-0.0009765625
0.00048828125
-0.000244140625
0.0001220703125
-6.103515625e-05
3.0517578125e-05
-1.52587890625e-05
7.62939453125e-06
-3.814697265625e-06
1.9073486328125e-06
1.9073486328125e-06

Criterio 2:
```

```
0.5
-0.25
0.125
-0.0625
0.03125
-0.015625
0.0078125
-0.00390625
0.001953125
-0.0009765625
0.00048828125
-0.000244140625
0.0001220703125
-6.103515625e-05
3.0517578125e-05
-1.52587890625e-05
7.62939453125e-06
7.62939453125e-06

Criterio 3:
0.5
-0.25
0.125
-0.0625
0.03125
-0.015625
0.0078125
-0.00390625
0.001953125
-0.0009765625
0.00048828125
-0.000244140625
0.0001220703125
-6.103515625e-05
3.0517578125e-05
-1.52587890625e-05
7.62939453125e-06
7.62939453125e-06

Criterio 4:
17
0.5
-0.25
0.125
-0.0625
0.03125
-0.015625
0.0078125
-0.00390625
0.001953125
-0.0009765625
```

```
0.00048828125
-0.000244140625
0.0001220703125
-6.103515625e-05
3.0517578125e-05
-1.52587890625e-05
7.62939453125e-06
7.62939453125e-06
```

- $f_2(x) = e^x - e \implies f_2(1) = 0,\ f_2'(1) = e\ >> 1$

```
Introduce el primer valor del intervalo: 0
Introduce el segundo valor del intervalo: 3
Introduzca la funcion de la que quiere hallar una raiz en el intervalo anterior
f(x) := np.exp(x) - np.exp(1)
Criterio 1:
1.5
0.75
1.125
0.9375
1.03125
0.984375
1.0078125
0.99609375
1.001953125
0.9990234375
1.00048828125
0.999755859375
1.0001220703125
0.99993896484375
1.000030517578125
0.9999847412109375
1.0000076293945312
0.9999961853027344
1.0000019073486328
1.0000019073486328
Criterio 2:
1.5
0.75
1.125
0.9375
1.03125
0.984375
1.0078125
0.99609375
0.99609375
1.001953125
0.9990234375
1.00048828125
0.999755859375
```

```
1.0001220703125
0.99993896484375
1.000030517578125
0.9999847412109375
1.0000076293945312
0.9999961853027344
1.0000019073486328
1.0000019073486328
Introduce una raiz: 1
Criterio 3:
1.5
0.75
1.125
0.9375
1.03125
0.984375
1.0078125
0.99609375
1.001953125
0.9990234375
1.00048828125
0.999755859375
1.0001220703125
0.99993896484375
1.000030517578125
0.9999847412109375
1.0000076293945312
1.0000076293945312
Criterio 4:
17
1.5
0.75
1.125
0.9375
1.03125
0.984375
1.0078125
0.99609375
1.001953125
0.9990234375
1.00048828125
0.999755859375
1.0001220703125
0.99993896484375
1.000030517578125
0.9999847412109375
1.0000076293945312
1.0000076293945312
```

- $f_3(x) = ln(x) - 1 \implies f_3(e) = 0,\ f_3'(e) = \frac{1}{e} << 1$

```
  Introduce el primer valor del intervalo: 0
Introduce el segundo valor del intervalo: 4
Introduzca la funcion de la que quiere hallar una raiz en el intervalo anterior
f(x) := np.log(x)-1
source/Ejercicio-pruebas.py:1: RuntimeWarning: divide by zero encountered in log
  #Algoritmo de biseccion
Criterio 1:
2.0
3.0
2.5
2.75
2.625
2.6875
2.71875
2.703125
2.7109375
2.71484375
2.716796875
2.7177734375
2.71826171875
2.718505859375
2.7183837890625
2.71832275390625
2.718292236328125
2.7182769775390625
2.7182846069335938
2.7182846069335938
Criterio 2:
2.0
3.0
2.5
2.75
2.625
2.6875
2.71875
2.703125
2.7109375
2.71484375
2.716796875
2.7177734375
2.71826171875
2.71826171875
Introduce una raiz: 2.718281828459
Criterio 3:
2.0
3.0
2.5
2.75
2.625
2.6875
2.71875
```

```
2.703125
2.7109375
2.71484375
2.716796875
2.7177734375
2.71826171875
2.718505859375
2.7183837890625
2.71832275390625
2.718292236328125
2.7182769775390625
2.7182769775390625
Criterio 4:
17
2.0
3.0
2.5
2.75
2.625
2.6875
2.71875
2.703125
2.7109375
2.71484375
2.716796875
2.7177734375
2.71826171875
2.718505859375
2.7183837890625
2.71832275390625
2.718292236328125
2.718292236328125
```

2. Anexo-Código.

```
#Algoritmo de biseccion
#Realizadores:
#Pedro Bonilla, Johanna Capote y Guillermo Galindo
#Ejecucion:
#$ python3.5 Ejercicio1.py

#IMPORTANTE: para ejecutar este programa se requiere tener instalado "numpy"
#en su defecto, se puede sustituir por funciones de la libreria math

import numpy as np

Threshold = pow(10,-5)
Root = 0
counter = 0

#Lectura del intervalo
a=float(input('Introduce el primer valor del intervalo: '))
b=float(input('Introduce el segundo valor del intervalo: '))

#Los valores asignados son irrelevantes, pero necesitamos tener las variables
    iniciadas
x1 = a
x2 = b

#Lectura de la funcion
funcion = input('Introduzca la funcion de la que quiere hallar una raiz en el \
intervalo anterior\nf(x) := ')
exec('f = lambda x:' + funcion)

#Signo de los extremos del intervalo
sgnI = np.sign(f(a))
sgnD = np.sign(f(b))

n = 0

#Criterio de parada 1

print ("Criterio 1:")
def halt1(x,y):
    return True if abs(x-y) > Threshold else False

while(halt1(x1,x2)) :
    x2=x1
    x1 = (a+b)/2

    #Reducimos el intervalo a la mitad
    if(np.sign(f(x1)) == sgnI):
        a = x1
    else:
        b = x1
    #Descomentar la linea de abajo para mostrarlo en cada iteracion
    #print(x1)

#Imprimimos la aproximacion final
print(x1)

#Criterio de parada 2
print ("Criterio 2:")
def halt2(x):
    return True if abs(f(x)) > Threshold else False
```

```
while(halt2(x1)) :
    x1 = (a+b)/2

    #Reducimos el intervalo a la mitad
    if(np.sign(f(x1)) == sgnI):
        a = x1
    else:
        b = x1
    #Descomentar la linea de abajo para mostrarlo en cada iteracion
    print(x1)

#Imprimimos la aproximacion final
print(x1)

#Criterio de parada 3
Root=float(input('Introduce una raiz: '))

print ("Criterio 3:")
def halt3(x):
     return True if abs(x-Root) > Threshold else False

while(halt3(x1)) :
    x1 = (a+b)/2

    #Reducimos el intervalo a la mitad
    if(np.sign(f(x1)) == sgnI):
        a = x1
    else:
        b = x1
    #Descomentar la linea de abajo para mostrarlo en cada iteracion
    print(x1)

#Imprimimos la aproximacion final
print(x1)

#Criterio de parada 4
print ("Criterio 4:")
n = int(np.log2((b-a)/Threshold)-1)
print(n)
for i in range(n):
    x1 = (a+b)/2

    #Reducimos el intervalo a la mitad
    if(np.sign(f(x1)) == sgnI):
        a = x1
    else:
        b = x1
    #Descomentar la linea de abajo para mostrarlo en cada iteracion
    print(x1)

#Imprimimos la aproximacion final
print(x1)
```

../source/Ejercicio1.py