Análisis Numérico
Taller 2

Carlos Barón
Santiago Chaustre
Andrés Cocunubo

TRISECCIÓN f(x)=x**3 +4x +10x -20

```python
from math import *
import time

def main():
    fx = input('Ecuacion f(x) = ')
    xi = float(input('Limite inferior : '))
    xf = float(input('Limite superior : '))
    err = float(input('Ingresa la tolerancia : '))
    nIte = ceil((log(xf - xi) - log(err)) / log(2))
    i = 1

    fi = eval(fx,{'x' : xi})
    ff = eval(fx,{'x' : xf})
    ft = 1E10
    p3 = 1E10
    p4 = 1E10
    res = 0
    res2 = 0
    prom = 1E10
    signo = ''
    starttime = time.time()
    if fi * ff <0 :
        print('___________________________________________
        print('| {:^15} | {:^15} | {:^15} | {:^15} | {:^15} | {:^15} |
        #print('___________________________________________
        while i <= nIte and (abs(p3) > err and abs(p4) > err   ):

            xt = (xi + xf)/2
            p1 = (xi + xt)/2
            p2 = (xt + xf)/2
            p3 = eval (fx,{'x' : p1})
            p4 = eval (fx,{'x' : p2})
            p5 = ff * p4
            p6 = fi * p3
            p7 = p3 * p4

            print('| {:^15} | {:^15.4f} | {:^15.4f} | {:^15.4f} | {:^1



            if p6 < 0 :
                xf = p1
                signo = 'INFERIOR'
            if p7 < 0 :
                xi = p1
                xf = p2
                signo = 'MITAD'
            if p5 < 0 :
                xi = p2
                signo = 'SUPERIOR'

            res = p1
            res2 = p2
            prom = abs(p3) + abs(p4) / 2
            i+=1
    else:
        print ("\n Ingreso algun dato erroneo ")
        exit(1)
    endtime = time.time() - starttime ;
    print('| {:^15} | {:^15.4f} | {:^15.4f} | {:^15.4f} | {:^15.4f} |
    print('___________________________________________


    print ( "El resultado aproximado es : %.4f" % ((res+res2)/2 ))
    print ( "El tiempo de ejecucion fue : %.4f Sg" % endtime )
    exit(0)
main()
```

Resultados:

```
Ecuacion f(x) = x**3 +2*x*2+10*x-20
Limite inferior : 0
Limite superior : 2
Ingresa la tolerancia : 0.001
```

i	a	b	ft1	ft2	cambio	xi	xf
1	0.5000	1.5000	-12.8750	4.3750		0.0000	2.0000
2	0.7500	1.2500	-9.0781	-0.5469	MITAD	0.5000	1.5000
3	1.3125	1.4375	0.6360	3.0955	SUPERIOR	1.2500	1.5000
4	1.2656	1.2969	-0.2540	0.3374	INFERIOR	1.2500	1.3125
5	1.2734	1.2891	-0.1068	0.1889	MITAD	1.2656	1.2969
6	1.2773	1.2852	-0.0331	0.1148	MITAD	1.2734	1.2891
7	1.2793	1.2832	0.0039	0.0778	MITAD	1.2773	1.2852
8	1.2778	1.2788	-0.0238	-0.0054	INFERIOR	1.2773	1.2793
9	1.2789	1.2792	-0.0031	0.0015	SUPERIOR	1.2788	1.2793
10	1.2790	1.2791	-0.0019	0.0004	MITAD	1.2789	1.2792
11	1.2790	1.2791	-0.0019	0.0004	MITAD	1.2790	1.2791

```
El resultado aproximado es : 1.2791
El tiempo de ejecucion fue : 0.0232 Sg
```

CUATRISECCIÓN f(x)=x**3 +4x +10x -20

```python
from math import *

import time

def main():
    fx = input('Ecuacion f(x) = ')
    xi = float(input('Limite inferior : '))
    xf = float(input('Limite superior : '))
    err = float(input('Ingresa la tolerancia : '))
    nIte = ceil((log(xf - xi) - log(err)) / log(2))
    i = 1

    fi = eval(fx,{'x' : xi})
    ff = eval(fx,{'x' : xf})
    ft = 1E10
    p3 = 1E10
    p4 = 1E10
    res = 0
    res2 = 0
    signo = ''
    starttime = time.time()
    if fi * ff <0 :
        print('___________________________________________
        print('| {:^15} | {:^15} | {:^15} | {:^15} | {:^15} | {:^15} | {:^20} | {

        while i <= nIte and (abs(p3) > err and abs(p4) > err and abs(ft) > err):

            xt = (xi + xf)/2
            p1 = (xi + xt)/2
            p2 = (xt + xf)/2
            ft = eval (fx,{'x' : xt})
            p3 = eval (fx,{'x' : p1})
            p4 = eval (fx,{'x' : p2})

            p5 = ff * p4
            p6 = fi * p3
            p7 = p3 * ft
            p8 = ft * p4



            if p6 < 0 :
                xf = p1
            elif p7 < 0 :
                xi = p1
                xf = xt
            elif p8 < 0 :
                xi = float (xt)
                xf = p2
            elif p5 < 0 :
                xi = p2

            res = p1
            res2 = p2
            res3 = xt
            i+=1
    else:
        print ("\n Ingreso algun dato erroneo ")
        exit(1)
    endtime = time.time() - starttime
    print('| {:^15} | {:^15.4f} | {:^15.4f} | {:^15.4f} | {:^15.4f} | {:^
    print('___________________________________________
    print ( "El resultado aproximado es : %.4f" % ((res+res2+res3)/3 ))
    print ( "El tiempo de ejecucion fue : %.4f Sg" % endtime )
    exit(0)
main()
```

Resultados:

```
Ecuacion f(x) = x**3 +2*x*2+10*x-20
Limite inferior : 0
Limite superior : 4
Ingresa la tolerancia : 0.001
```

i	xi	xf	a	xt	b	p5	p6	p7	p8
1	0.0000	4.0000	1.0000	2.0000	3.0000	4900.0000	100.0000	-80.0000	784.0000
2	1.0000	2.0000	1.2500	1.5000	1.7500	985.9375	10.9375	-2.3926	43.1348
3	1.2500	1.5000	1.3125	1.3750	1.4375	309.5459	-12.7197	1.1763	5.7254
4	1.2500	1.3125	1.2656	1.2812	1.2969	33.7444	5.0793	-0.0104	0.0138
5	1.2656	1.2812	1.2695	1.2734	1.2773	-3.3064	3.6089	0.0193	0.0035
6	1.2773	1.2812	1.2783	1.2793	1.2803	2.2324	0.2922	-0.0001	0.0001
7	1.2783	1.2793	1.2786	1.2788	1.2791	-0.0762	0.1999	0.0001	0.0000
8	1.2791	1.2793	1.2786	1.2788	1.2791	-0.0762	0.1999	0.0001	0.0000

```
El resultado aproximado es : 1.2788
El tiempo de ejecucion fue : 0.0180 Sg
```

MÉTODO DE PUNTOFIJO f(x)=eˆ(-x)

```python
def puntofijo():
    x=0
    tolerancia=0.0001
    N=50
    gx="math.exp(-x)"
    er=100;
    i=0;
    print('iteracion\tg(f(x))\t\terror')
    while(i<=N and er>=tolerancia):
        temp=x
        x=eval(gx);
        er=abs((x-temp));
        print("%d\t\t%.4f\t\t%.4f"%(i,x,er));
        i+=1;

    print("La raiz es:",x);

puntofijo();
```

Resultados:

iteracion	g(f(x))	error
0	1.0000	1.0000
1	0.3679	0.6321
2	0.6922	0.3243
3	0.5005	0.1917
4	0.6062	0.1058
5	0.5454	0.0608
6	0.5796	0.0342
7	0.5601	0.0195
8	0.5711	0.0110
9	0.5649	0.0063
10	0.5684	0.0035
11	0.5664	0.0020
12	0.5676	0.0011
13	0.5669	0.0006
14	0.5673	0.0004
15	0.5671	0.0002
16	0.5672	0.0001
17	0.5671	0.0001

```
La raiz es: 0.5671190400572149
El tiempo de ejecucion fue : 0.0018 Sg
```

MÉTODO DE PUNTOFIJO CON CRITERIO DE CONVERGENCIA f(x)=x**3 +4x +10x -20

```python
from math import *
from sympy import *

fx= "x**3 +2x*2 +10*x -20" #se tiene f= "x**3 +2x*2 +10*x -20"
#sus formas son "20 / (x**2 + 2*x +10)" y "x**3 + 2*x**2 + 10*x - 20"
g1x= "20 / (x**2 + 2*x +10)"
g2x= "x**3 + 2*x**2 + 10*x - 20"
g1dx=str(diff(g1x))
g2dx=str(diff(g2x))
x=1
a= abs(eval(g1dx))
b= abs(eval(g2dx))
if a<1:
    gx=g1x
    gdx=g1dx
if b<1:
    gx=g2x
    gdx=g2dx

tolerancia=0.0001
N=50
i=1

print('iteracion\tg(f(x))\t\terror\t\tderivada')
while i<=N:
    d= abs(float(eval(gdx)))
    if d == 0:
        print ("El metodo no aplica")
    x0=eval(gx)
    er=abs(x0-x)
    print("%d\t\t%.5f\t\t%.5f\t\t%.5f"%(i,x0,er,d));
    if er<tolerancia:
        print("La raiz es ",x0)
        break
    i=i+1
    x=x0
if i>=N:
    print("El metodo no converge ")
```

Resultados:

iteracion	g(f(x))	error	derivada
1	1.53846	0.53846	0.47337
2	1.29502	0.24344	0.42572
3	1.40183	0.10681	0.45100
4	1.35421	0.04762	0.44047
5	1.37530	0.02109	0.44529
6	1.36593	0.00937	0.44317
7	1.37009	0.00416	0.44412
8	1.36824	0.00184	0.44370
9	1.36906	0.00082	0.44389
10	1.36870	0.00036	0.44380
11	1.36886	0.00016	0.44384
12	1.36879	0.00007	0.44382

```
La raiz es  1.3687861025779886
El tiempo de ejecucion fue : 0.0018 Sg
```

MÉTODO DE NEWTON-RHAPSON f(x)=x**3 +4x +10x -20

```python
from math import *
from sympy import *

fx= "x**3 +2x*2 +10*x -20" #se tiene f= "x**3 +2x*2 +10*x -20"
#sus formas son "20 / (x**2 + 2*x +10)" y "x**3 + 2*x**2 + 10*x - 20"
gx="x**3 + 2*x**2 + 10*x - 20" #se toma debido a que es la corta el eje en
g1x=str(diff(gx))
x=1
tolerancia=0.0001
N=50
i=1
print('iteracion\tg(f(x))\t\terror\t\tderivada')
while i<=N:
    d= float(eval(g1x))
    if d == 0:
        print ("El metodo no aplica")
    x0=x-float(eval(gx))/d
    er=abs(x0-x)
    print("%d\t\t%.5f\t\t%.5f\t\t%.5f"%(i,x0,er,d));
    if er<tolerancia:
        print("La raiz es ",x0)
        break
    i=i+1
    x=x0
if i>=N:
    print("El metodo no converge ")
```

Resultados:

iteracion	g(f(x))	error	derivada
1	1.41176	0.41176	17.00000
2	1.36934	0.04243	21.62630
3	1.36881	0.00053	21.10259
4	1.36881	0.00000	21.09614

```
La raiz es  1.3688081078213745
El tiempo de ejecucion fue : 0.0007 Sg
```

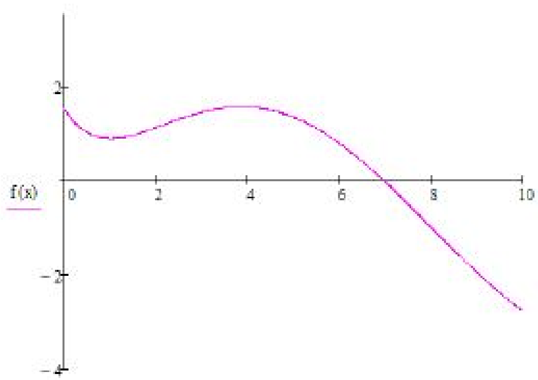
¿POR QUÉ USAN EL MÉTODO DE NEWTON-RAPHSON EN EL ARTÍCULO "APPLICATION OF NUMERICAL METHODS TO SOLVE NON-LINEAR EQUATIONS FOR SEA WAVE MODELING"?

Tomando como ecuación y valores:

$$h = h_0\left[sen\left(\frac{2\pi x}{\lambda}\right)\cos\left(\frac{2\pi tv}{\lambda}\right) + e^{-x}\right]$$

Para $\lambda = 16$, t=12, v=48, h=0.4h_o, (40% de la altura inicial)

A través del método numérico de newton-raphson pretenden encontrar una distancia x, donde la ola se rompe.Toman un valor inicial $x0$ igual al punto medio entre un intervalo [5,10] donde se encuentra la solución más pequeña, por lo tanto el $x0$ es iguala 7.5.



Se toma una cantidad de iteraciones arbitraria, en este caso 10 y se obtienen los siguientes resultados:

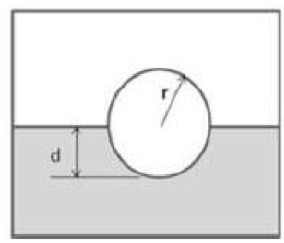
x =

	0
0	6.5
1	6.978852791803652
2	6.954779208670931
3	6.954731290074158
4	6.954731289881505
5	6.954731289881505
6	6.954731289881505
7	6.954731289881505
8	6.954731289881505
9	6.954731289881505
10	6.954731289881505

Adicionalmente utilizan el método de la secante con el fin de corroborar los resultados obtenidos en el método anterior, dando como resultado los mismos números.

¿POR QUÉ USAN EL MÉTODO DE NEWTON-RAPHSON EN EL ARTÍCULO "Aplicación del Método de Newton Raphson en el Principio de la "Esfera Sumergida en Agua""?

En esté artículo usan el método de newton-raphson para encontrar un valor apróximado de cuanto se hundirá la esfera en un sistema hidráulico.La esfera tiene un radio de 5.5cm y una gravedad de 0.6.



Ayudándose del principio de Arquímides, Signos de Descartes y regla de Laguerre llegan a la siguiente funcion:

$$f(d) = d^3 - 0.165d^2 + 3.993\ x\ 10^{-4}$$

Según el análisis el polinomio tiene tres raíces, de las cuales dos de ellas son positivas las cuales son las que interesan para la resolución del problema.
Toman un valor inicial $d0$ de 0.05m, ya que el diametro del circulo son 0.11m. Realizan tres iteraciones puesto que el error relativo absoluto es de cero en esta última.Por lo tanto, el valor apróximado para la variable d es igual a 0.06m.