Parcial 1 Analisis Numérico

Cristian Camilo Contreras Borja

Punto1

1. En cada uno de los siguientes ejercicios implemente en R o Python el algoritmo necesario que permita calular el número mínimo de operaciones requeridas para resolver el problema, una gráfica de n versus numero de operaciones y evaluar el error relativo, en cada caso

c) Algoritmo que le permita sumar los primeros números naturales al cuadrado. Imprima varias pruebas, para diferentes valores de n y evalue el error relativo porcentual para cuando $n = 4, 5, 10$ y el error en cada valor es de 0.1

```python
#@author: cristian

def funcion(n):

   iter=0
   error=0.1
   result=0
   suma=0
   while(iter<=n):
        num=iter*iter
        suma=suma+num

        iter=iter+1
   print("N(equivale a la cantidad de naturales a sumar):",n)
   print("Solucion",suma)
   errorrela=(error/suma)*100
   print("Error Relativo",errorrela)
   print("iteraciones",iter)

funcion(10)
```

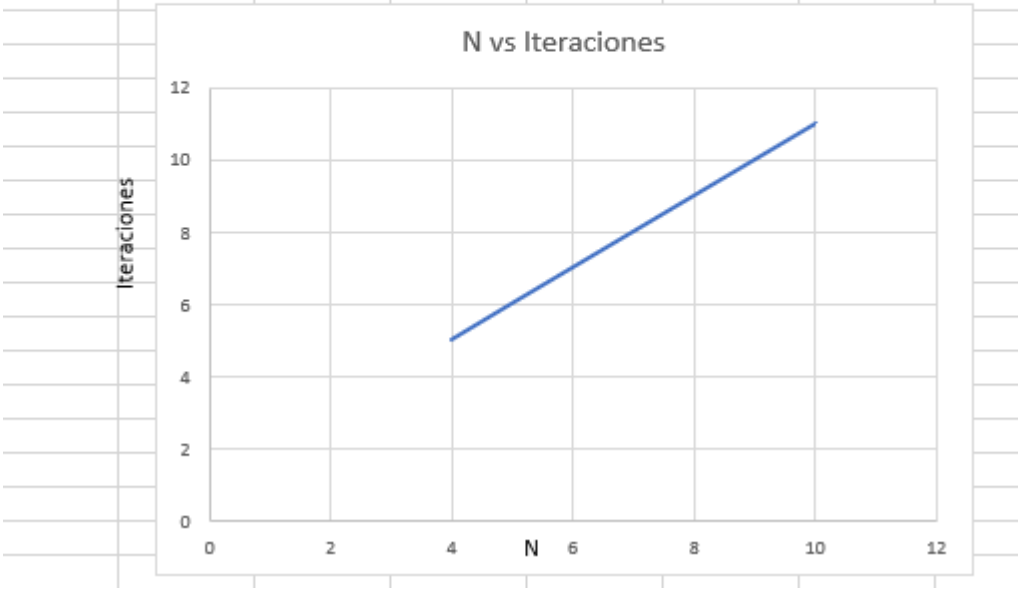
```
N(equivale a la cantidad de naturales a sumar): 4
Solucion 30
Error Relativo 0.33333333333333337
iteraciones 5
```

```
N(equivale a la cantidad de naturales a sumar): 5
Solucion 55
Error Relativo 0.18181818181818182
iteraciones 6
```

```
N(equivale a la cantidad de naturales a sumar): 10
Solucion 385
Error Relativo 0.025974025974025976
iteraciones 11
```

- Grafica n vs iteraciones

N	Iteraciones
4	5
5	6
10	11



Punto 2

2. Para cada uno de los siguientes ejercicios: utilice el algoritmo señalado para encontrar la intersección entre $f(x) = x^2$ y $g(x) = 1 + cosx$, en el intervalo $[1, 2]$ con $E < 10^{-9}$, determinar el número de iteraciones realizadas,una grafica que evidencie el tipo de convergencia del método,debe expresarla en notación $O()$

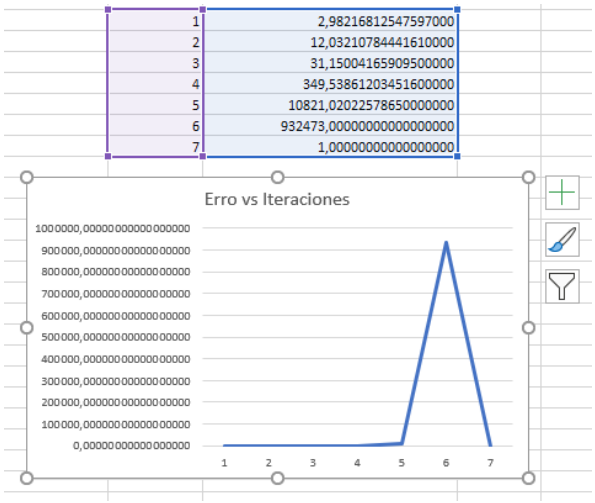
- Cogido(Python)

```python
#Cristian Contreras
import math
import decimal
import matplotlib.pyplot as plot
x = []
y = []
def funcion(fun,xn,xn1,tol):

    it = 0
    error = 0.1
    x0=xn
    x1=xn1
    while(it<100 and error>tol):
        if((fun(x1)-fun(x0))==0):
            break
        x2= x1 - fun(x1)*((x1-x0)/(fun(x1)-fun(x0)))
        error = abs((fun(x2)-fun(x1)/fun(x2)))
        y.append(error)
        it= it+1
        x.append(it)
        x0=x1
        x1=x2
        print(x2,error)

    print("Respuesta Final:",fun(x2),error)

def fun(x):
    vcos = math.cos(x)
    return x**2 - vcos

funcion(fun,2.0,1.0,10**-9)
```

```
0.8838105387976174 2.9821681254759715
0.8292379144270658 12.03210784441618
0.8242966689403904 31.150041659095052
0.8241327825561032 349.53861203451606
0.8241323123459798 10821.020225786595
0.8241323123025224 932473.0
0.8241323123025224 1.0
Respuesta Final: -1.1102230246251565e-16 1.0
```

Se realizaron 7 iteraciones.

- Grafica error vs iteraciones



La convergencia del método a infinito en caso de no encontrar el resultado pero al momento de encontrarlo vuelve a converger a 0.