Parcial 1

1.

from math import \*

from sympy import \*

import time

import sys

starttime = time.time()

a=2

b=2

c=7

gx=str(a)+"\*x\*\*2+"+str(b)+"\*x+"+str(c)

g1dx=str(diff(gx))

g2dx=str(diff(g1dx))

x=1.5

i=1

N=50

tolerancia=0.001

r=(b\*\*2-4\*a\*c)

if r<0:

q= -b/(2\*a)

i= (abs(r))\*\*(1/2)/(2\*a)

print("Para el caso de las complejas conjugadas:\n")

print ("La raiz 1 es ",q,"+",i,"i")

print ("La raiz 2 es ",q,"-",i,"i")

endtime = time.time() - starttime

print ("El tiempo de ejecucion fue : %.4f Sg" % endtime)

sys.exit(0)

print('iteracion\tg(f(x))\t\terror\t\tderivada\_1\t\tderivada\_2')

while i<=N:

a=float(eval(gx))

b=float(eval(g1dx))

c=float(eval(g2dx))

d=(abs(b)\*\*2 - a\*c)

x0=x- a\*b/d

er=abs(x0-x)

print("%d\t\t%.5f\t\t%.5f\t\t%.5f\t\t%.5f"%(i,x0,er,b,c))

if er<tolerancia:

endtime = time.time() - starttime

print("Para el caso de multiplicidad dos:\n")

print("La raiz 1 es ",x0)

print("La raiz 2 es -",x0)

print ("El tiempo de ejecucion fue: %.4f Sg" % endtime)

break

i=i+1

x=x0

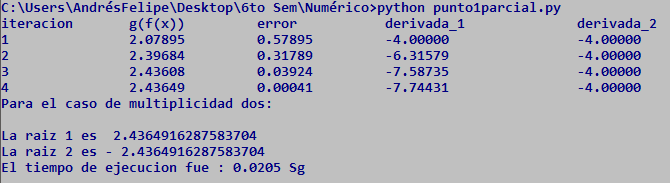
if i>=N:

print("El metodo no converge ")

Para el caso de raíces de multiplicidad dos:

Se toma como ejemplo a=-2, b=2, c=7 🡪 -2x^2+2x+7=0

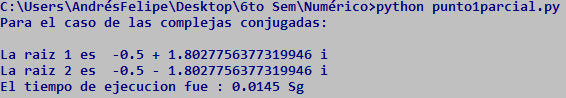
De esto se obtiene:



Para el caso de raíces que son complejas conjugadas:

Se toma como ejemplo a=2, b=2, c=7 🡪 2x^2+2x+7=0

De esto se obtiene:



2.

from math import \*

from sympy import \*

import time

gx="2\*\*x -1.3"

x=-1

tolerancia=0.0001

N=1000

i=1

x0=0

print('iteracion\tg(f(x))\t\terror')

starttime = time.time()

while i<=N:

fx=eval(gx)

aux=x

x=x0

fx0=eval(gx)

x=aux

xi=x-fx\*(x-x0)/(fx-fx0)

er=abs(x0-x)

print("%d\t\t%.4f\t\t%.4f"%(i,x0,er));

if er<tolerancia:

endtime = time.time() - starttime

print("La raiz es %.4f" %x0)

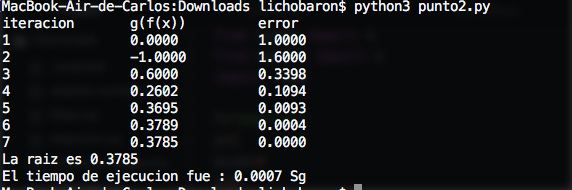
print ("El tiempo de ejecucion fue : %.4f Sg" % endtime)

break

i=i+1

x0=x

x=xi



3 a

from math import \*

from sympy import \*

import time

gx="8\*\*x -25" #funcion inversa transformada

g1x=str(diff(gx))

x=1

tolerancia=0.0001

N=50

i=1

print('iteracion\tg(f(x))\t\terror\t\tderivada')

starttime = time.time()

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:

endtime = time.time() - starttime

print("La raiz es ",x0)

print ("El tiempo de ejecucion fue : %.4f Sg" % endtime)

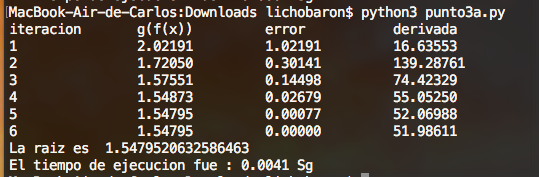
break

i=i+1

x=x0

if i>=N:

print("El metodo no converge ")



3b

from sympy import \*

from mpmath import \*

import math

import time

fx1= "math.log(x + 2,10)" #se tiene f= "x\*\*3 +2x\*2 +10\*x -20"

fx2= "-sin(x)"

#sus formas son "20 / (x\*\*2 + 2\*x +10)" y "x\*\*3 + 2\*x\*\*2 + 10\*x - 20"

fd1="1/(x+2)"#str(diff(fx1)) #se toma debido a que es la corta el eje en la interpretacion grafica

fd2="-cos(x)"#str(diff(fx2))

x=1

x1=x

x2=x

tolerancia=0.1

N=50

i=1

print('iteracion1\tg(f(x))1\t\terror1\t\tderivada1\t\t\titeracion2\tg(f(x)2)\t\terror2\t\tderivada2')

starttime = time.time()

while i<=N:

aux=x

x=x1

d1=float(eval(fd1))

a1=float(eval(fx1))

x01=x1-a1/d1

x=aux

er1=abs(x01-x1)

aux=x

x=x2

x=aux

d2= float(eval(fd2))

a2= float(eval(fx2))

x02=x2-a2/d2

x=aux

er2=abs(x02-x2)

print("%d\t\t%.6f\t\t%.6f\t\t%.6f\t\t\t%d\t\t%.6f\t\t%.6f\t\t%.6f"%(i,x01,er1,d1,i,x02,er2,d2));

if abs(1-x01)<= abs(1-x02):

endtime = time.time() - starttime

print("Se cruzan en %.6f" %x01)

print ("El tiempo de ejecucion fue : %.4f Sg" % endtime)

break

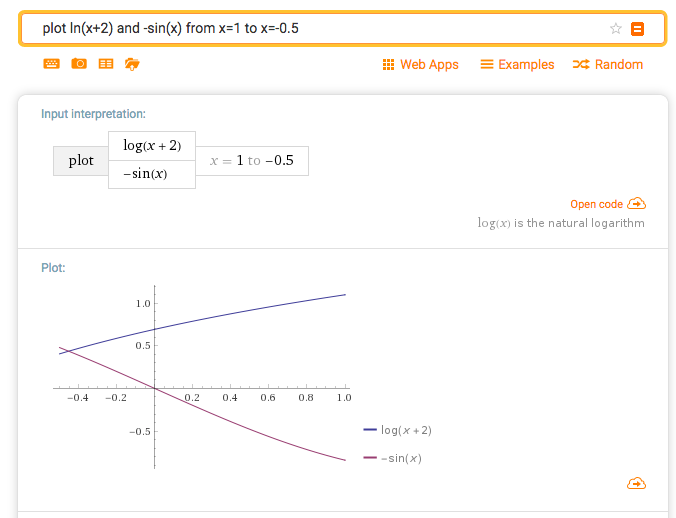
i=i+1

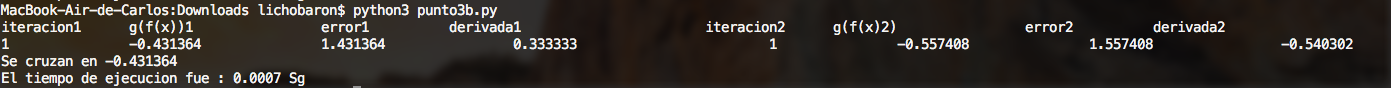
x1=x01

x2=x02

if i>=N:

print("El metodo no converge ")





4

from sympy import \*

from mpmath import \*

import time

f="exp(-x)"

y=5

acum=0

x=0

i=1

while i<9:

if i%2==0:

d="-exp(-x)"

else:

d="exp(-x)"

r= (eval(d)\*y\*\*i)/(fac(i))

i=i+1

acum= acum+r

print ("la aproximacion 1 es: ",acum+eval(f))

p = taylor(exp, -x, 9)

print ("la aproximacio 2 es: ", polyval(p[::-1], 2.5 - 2.0))

print ("la aproximacion es mas exacta porque presenta un reultado mas cercano al real")

