Universidad Autónoma de Querétaro

Facultad de Ingeniería

Tarea 2: Métodos cerrados para ecuaciones de una variable

Análisis numérico

Autor:
J.A. Salinas Sánchez
Febrero 2022

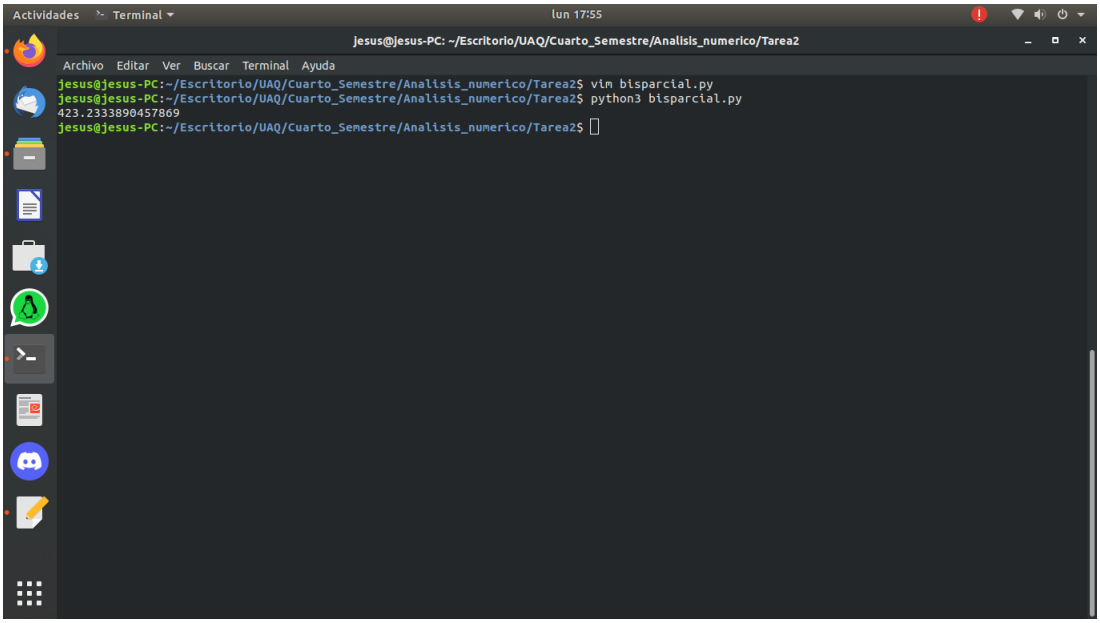
Índice general

0.1. Problema 1

0.1.1. Cerrado

```
import math
import matplotlib.pyplot as plt


def main():

    x=[]
    y=[]
    i=0
    print(bis())


def bis():

    xr=1
    xl=400
    xu=500
    i=0
    e=100
    temp=1
    flag=False
    error=[]
    iteraciones=[]


    while(e>=0.5 or flag!=True):

        temp=xr
        xr=(xu+xl)/2

        e=abs((xr-temp)/(temp))*100
        error.append(e)
        iteraciones.append(i)
        i+=1

        if ((((1/(4*3.141592*8.885e-12))*((2e-5)*xl)/(xl**2+0.9**2)
    **(3/2))-1)*(((1/(4*3.141592*8.885e-12))*((2e-5)*xr)/(xr
    **2+0.9**2)**(3/2))-1) < 0):
            xu=xr
        elif ((((1/(4*3.141592*8.885e-12))*((2e-5)*xl)/(xl
    **2+0.9**2)**(3/2))-1)*(((1/(4*3.141592*8.885e-12))*((2e-5)*xr)
    /(xr**2+0.9**2)**(3/2))-1) > 0):
            xl=xr
        else:
            flag=True

```

```
    return(xr)

if __name__=='__main__':
    main()
```



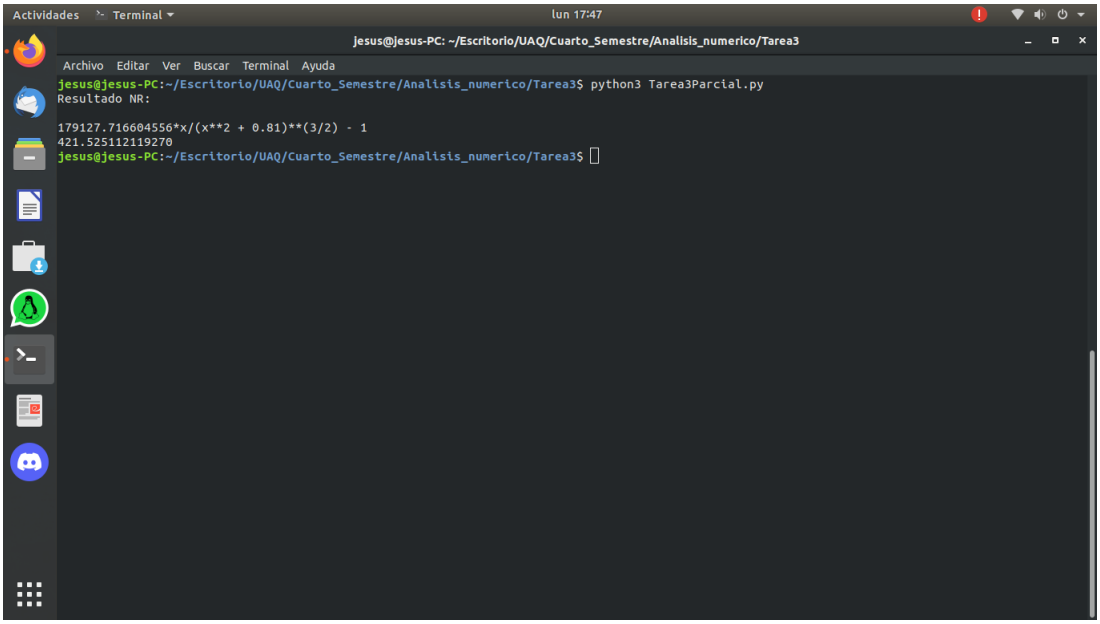
0.1.2. Abierto

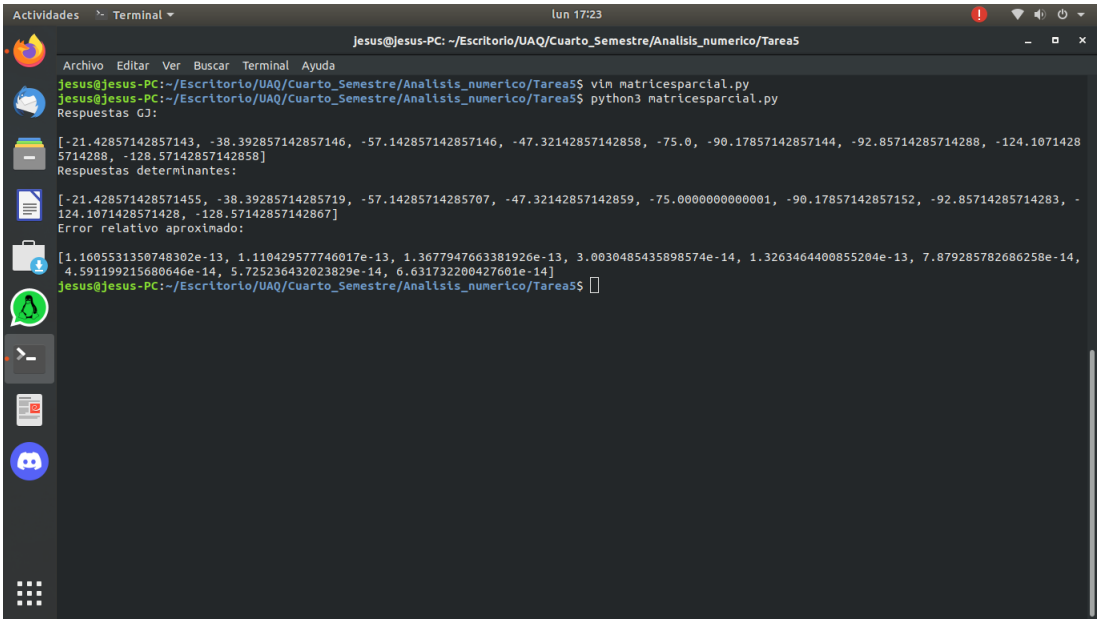
```
import math
import numpy
import pandas
import sympy

x=sympy.symbols('x')
xi=sympy.symbols('xi')
xl=sympy.symbols('xl')
xu=sympy.symbols('xu')

def main():

    func='((1/(4*3.141592*8.885e-12))*((2e-5)*x)/(x**2+0.9**2)
   **(3/2))-1'

    x0=500
    emax=10
    print('Resultado NR: \n')
    print(f'{NR(func,x0,emax)} m')

def getfunction(f):
    global x
    global xi
    global xl
    global xu
    return(sympy.sympify(f))

def NR(eq,x0,emax):
    global x
    ecuacion=getfunction(eq)
    derivada= sympy.diff(ecuacion)
    f_NR=x-(ecuacion/derivada)
    e=100
```

```
    xr=x0
    while(e>emax):
        temp=xr
        xr=f_NR.evalf(subs={x:temp})
        if(xr!=0):
            e=abs((xr-temp)/xr)*100
    return(xr)

def PF(eq,x0,emax):
    global x
    ecuacion=getfunction(eq)
    xr=x0
    e=100
    while(e>emax):
        temp=xr
        xr= ecuacion.evalf(subs={x:temp})
        if(xr!=0):
            e=abs((xr-temp)/xr)*100
    return(xr)

def Sec(eq,x0,emax):
    global x
    global xi
    global xl
    global xu
    ecuacion=getfunction(eq)
    a=eq.replace("x","xl")
    ecuacion2=getfunction(a)
    f_Sec=x-((ecuacion*(xl-x))/(a-ecuacion))
    temp1=x0
    temp2=1
    temp3=0
    xr=x0
    e=100
    while(e>emax):
        xr=f_Sec.evalf(subs={x:temp1,xl:temp2})
        temp3=ecuacion.evalf(subs={x:xr})
        if(xr!=0):
            e=abs((xr-temp1)/(temp1))*100
        if(ecuacion.evalf(subs={x:temp1})*temp3 < 0):
                temp1=temp1
                temp2=temp3
        elif(ecuacion.evalf(subs={x:temp2})*temp3 < 0):
            temp1=temp3
            temp2=temp2
        elif(temp3==0):
            return(xr)
        else:
            return(None)
    return(xr)




if __name__ =='__main__':
    main()
```

0.2. Problema 2

```
#coding:utf8
import numpy as np

```






```
4
5  def main():
6      matriz1
       =[[-4,1,0,1,0,0,0,0,0,0],[1,-4,1,0,1,0,0,0,0,0],[0,1,-4,0,0,1,0,0
7      tempmatrix=[0,0,0,0,0,0,0,0,0]
8      uk=9
9
10     a=gaussjordan(matriz1,tempmatrix,uk)
11     b=krammer(matriz1,tempmatrix,uk)
12
13     e=[]
14     for i in range(len(a)):
15         ea=float((abs((a[i]-b[i])/a[i])))*100
16         e.append(ea)
17     print('Error relativo aproximado: \n')
18     print(e)
19
20 def gauss(m1,tm,uk):
21     i=0
22     j=0
23     k=0
```

```
24
25      for i in range(uk):
26          for j in range(i+1, uk):
27              ratio = m1[j][i]/m1[i][i]
28              for k in range(uk+1):
29                  m1[j][k] = m1[j][k]-ratio*m1[i][k]
30
31      tm[uk-1] = m1[uk-1][uk]/m1[uk-1][uk-1]
32
33      for i in range(uk-2,-1,-1):
34          tm[i] = m1[i][uk]
35          for j in range(i+1,uk):
36              tm[i] = tm[i] - m1[i][j]*tm[j]
37          tm[i] = tm[i]/m1[i][i]
38
39      print(tm)
40      return(0)
41
42  def gaussjordan(m1,tm,uk):
43      i=0
44      j=0
45      k=0
46
47      for i in range(uk):
48          for j in range(uk):
49              if i!=j:
50                  ratio = m1[j][i]/m1[i][i]
51                  for k in range(uk+1):
52                      m1[j][k] = m1[j][k]-ratio*m1[i][k]
53
54      for i in range(uk):
55          tm[i]=m1[i][uk]/m1[i][i]
56
57      print('Respuestas GJ: \n')
58      print(tm)
59      return(tm)
60
61  def gausspivote(m1,tm,uk):
62      i=0
63      j=0
64      k=0
65
66      for i in range(uk):
67          for j in range(i+1, uk):
68
69              if(m1[i][i]==0):
70                  m1 = pivoteo(m1,uk,i)
71
72              ratio = m1[j][i]/m1[i][i]
73              for k in range(uk+1):
74                  m1[j][k] = m1[j][k]-ratio*m1[i][k]
75
76      tm[uk-1] = m1[uk-1][uk]/m1[uk-1][uk-1]
77
78      for i in range(uk-2,-1,-1):
79          tm[i] = m1[i][uk]
80          for j in range(i+1,uk):
81              tm[i] = tm[i] - m1[i][j]*tm[j]
82
83          tm[i] = tm[i]/m1[i][i]
84
85      print(tm)
86      return(0)
```

```
87
88  def krammer(m1,tm,uk):
89
90      dets=[]
91      respuestas=[]
92
93      for k in range(uk+1):
94          m2=changecols(m1,uk,k)
95          dets.append(np.linalg.det(m2))
96
97      for k in range(uk):
98          respuestas.append(dets[k]/dets[uk])
99
100     print('Respuestas determinantes: \n')
101     print(respuestas)
102     return(respuestas)
103
104
105 def changecols(m,uk,col):
106     temp1=np.zeros((uk,uk+1))
107     temp2=np.zeros((uk,uk))
108
109     i=0
110     index=uk
111
112     while(i!=col):
113         i+=1
114
115     if(i!=uk):
116         for j in range(uk):
117             temp1[j][col]=m[j][index]
118             temp1[j][index]=m[j][col]
119
120         for i in range(uk):
121             for j in range(uk+1):
122                 if (j!=col and j!=index):
123                     temp1[i][j]=m[i][j]
124                     temp1[i][j]=m[i][j]
125
126     else:
127         temp1=m.copy()
128     for i in range(uk):
129         for j in range(uk):
130             temp2[i][j]=temp1[i][j]
131
132     return(temp2)
133
134
135 def pivoteo(m,uk,row):
136     temp=np.zeros((uk,uk+1))
137
138     if(row==uk):
139         index=row-1
140     else:
141         index=row+1
142
143     for j in range(uk+1):
144         temp[row][j]=m[index][j]
145         temp[index][j]=m[row][j]
146
147     for i in range(uk):
148         if (i!=row and i!=index):
149             for j in range(uk+1):
```

```
150                 temp[i][j]=m[i][j]
151                 temp[i][j]=m[i][j]
152 
153     return (temp)
154 
155 if __name__=='__main__':
156     main()
```