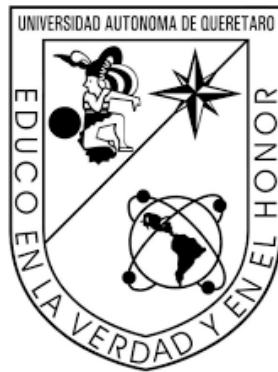


Universidad Autónoma de Querétaro

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

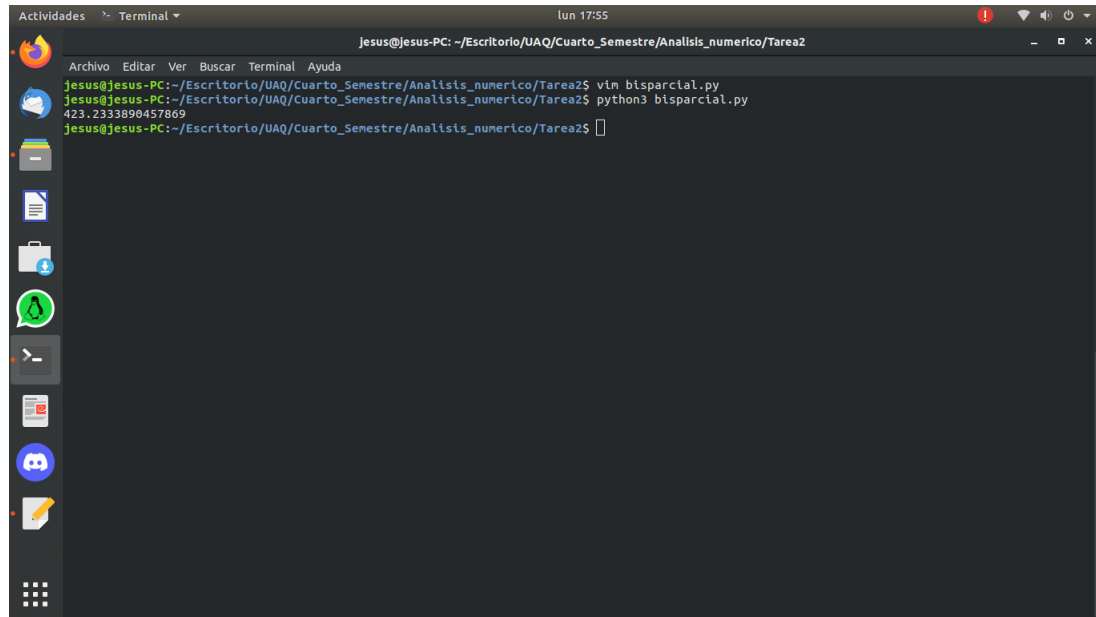
0.1.1. Cerrado

```
1 import math
2 import matplotlib.pyplot as plt
3
4
5 def main():
6
7     x=[]
8     y=[]
9     i=0
10    print(bis())
11
12
13 def bis():
14
15     xr=1
16     xl=400
17     xu=500
18     i=0
19     e=100
20     temp=1
21     flag=False
22     error=[]
23     iteraciones=[]
24
25
26     while(e>=0.5 or flag!=True):
27
28         temp=xr
29         xr=(xu+xl)/2
30
31         e=abs((xr-temp)/(temp))*100
32         error.append(e)
33         iteraciones.append(i)
34         i+=1
35
36         if (((1/(4*3.141592*8.885e-12))*((2e-5)*xl)/(xl**2+0.9**2)
37         *(3/2))-1)*(((1/(4*3.141592*8.885e-12))*((2e-5)*xr)/(xr
38         **2+0.9**2)*(3/2))-1) < 0):
39             xu=xr
40         elif (((1/(4*3.141592*8.885e-12))*((2e-5)*xl)/(xl
41         **2+0.9**2)*(3/2))-1)*(((1/(4*3.141592*8.885e-12))*((2e-5)*xr)
42         /(xr**2+0.9**2)*(3/2))-1) > 0):
43             xl=xr
44         else:
45             flag=True
```

```

43     return(xr)
44
45 if __name__=='__main__':
46     main()

```



0.1.2. Abierto

```

1 import math
2 import numpy
3 import pandas
4 import sympy
5
6 x=sympy.symbols('x')
7 xi=sympy.symbols('xi')
8 xl=sympy.symbols('xl')
9 xu=sympy.symbols('xu')
10
11 def main():
12
13     func='((1/(4*3.141592*8.885e-12))*((2e-5)*x)/(x**2+0.9**2)
14     *(3/2))-1'
15
16     x0=500
17     emax=10
18     print('Resultado NR: \n')
19     print(f'{NR(func,x0,emax)} m')
20
21 def getfunction(f):
22     global x
23     global xi
24     global xl
25     global xu
26     return(sympy.sympify(f))
27
28 def NR(eq,x0,emax):
29     global x
30     ecuacion=getfunction(eq)
31     derivada= sympy.diff(ecuacion)
32     f_NR=x-(ecuacion/derivada)
33     e=100

```

```

33     xr=x0
34     while(e>emax):
35         temp=xr
36         xr=f_NR.evalf(subs={x:temp})
37         if(xr!=0):
38             e=abs((xr-temp)/xr)*100
39     return(xr)
40
41 def PF(eq,x0,emax):
42     global x
43     ecuacion=getfunction(eq)
44     xr=x0
45     e=100
46     while(e>emax):
47         temp=xr
48         xr= ecuacion.evalf(subs={x:temp})
49         if(xr!=0):
50             e=abs((xr-temp)/xr)*100
51     return(xr)
52
53 def Sec(eq,x0,emax):
54     global x
55     global xi
56     global xl
57     global xu
58     ecuacion=getfunction(eq)
59     a=eq.replace("x","xl")
60     ecuacion2=getfunction(a)
61     f_Sec=x-((ecuacion*(xl-x))/(a-ecuacion))
62     temp1=x0
63     temp2=1
64     temp3=0
65     xr=x0
66     e=100
67     while(e>emax):
68         xr=f_Sec.evalf(subs={x:temp1,xl:temp2})
69         temp3=ecuacion.evalf(subs={x:xr})
70         if(xr!=0):
71             e=abs((xr-temp1)/(temp1))*100
72             if(ecuacion.evalf(subs={x:temp1})*temp3 < 0):
73                 temp1=temp1
74                 temp2=temp3
75             elif(ecuacion.evalf(subs={x:temp2})*temp3 < 0):
76                 temp1=temp3
77                 temp2=temp2
78             elif(temp3==0):
79                 return(xr)
80             else:
81                 return(None)
82     return(xr)
83
84
85
86
87 if __name__ == '__main__':
88     main()

```

0.2. Problema 2

```

1 #coding:utf8
2 import numpy as np
3

```

```
Actividades Terminal lun 17:47
Jesus@Jesus-PC: ~/Escritorio/UAQ/Cuarto_Semestre/Analisis_numerico/Tarea3
Archivo Editar Ver Buscar Terminal Ayuda
Jesus@Jesus-PC:~/Escritorio/UAQ/Cuarto_Semestre/Analisis_numerico/Tarea3$ python3 Tarea3Parcial.py
Resultado NR:
179127.716604556*x/(x**2 + 0.81)**(3/2) - 1
421.525112119270
Jesus@Jesus-PC:~/Escritorio/UAQ/Cuarto_Semestre/Analisis_numerico/Tarea3$
```

```
Actividades Terminal lun 17:23
Jesus@Jesus-PC: ~/Escritorio/UAQ/Cuarto_Semestre/Analisis_numerico/Tarea5
Archivo Editar Ver Buscar Terminal Ayuda
Jesus@Jesus-PC:~/Escritorio/UAQ/Cuarto_Semestre/Analisis_numerico/Tarea5$ vim matricesparcial.py
Jesus@Jesus-PC:~/Escritorio/UAQ/Cuarto_Semestre/Analisis_numerico/Tarea5$ python3 matricesparcial.py
Respuestas GJ:
[-21.42857142857143, -38.392857142857146, -57.142857142857146, -47.32142857142858, -75.0, -90.17857142857144, -92.85714285714288, -124.10714285714288, -128.57142857142858]
Respuestas determinantes:
[-21.428571428571455, -38.39285714285719, -57.14285714285707, -47.32142857142859, -75.00000000000001, -90.17857142857152, -92.85714285714283, -124.1071428571428, -128.57142857142867]
Error relativo aproximado:
[1.1695531350749382e-13, 1.110429577746017e-13, 1.3677947663381926e-13, 3.0030485435898574e-14, 1.3263464400855204e-13, 7.879285782686258e-14, 4.591199215680646e-14, 5.725236432023829e-14, 6.631732200427601e-14]
Jesus@Jesus-PC:~/Escritorio/UAQ/Cuarto_Semestre/Analisis_numerico/Tarea5$
```

```
4
5 def main():
6     matriz1
7     = [[-4,1,0,1,0,0,0,0,0],[1,-4,1,0,1,0,0,0,0],[0,1,-4,0,0,1,0,0,0],
8         [0,0,0,0,0,0,0,0,0],
9         [0,0,0,0,0,0,0,0,0],
10        [0,0,0,0,0,0,0,0,0],
11        [0,0,0,0,0,0,0,0,0],
12        [0,0,0,0,0,0,0,0,0]]
13
14     tempmatrix=[0,0,0,0,0,0,0,0,0]
15     uk=9
16
17     a=gaussjordan(matriz1,tempmatrix,uk)
18     b=krammer(matriz1,tempmatrix,uk)
19
20     e=[]
21     for i in range(len(a)):
22         ea=float((abs((a[i]-b[i])/a[i])))*100
23         e.append(ea)
24     print('Error relativo aproximado: \n')
25     print(e)
26
27 def gauss(m1,tm,uk):
28     i=0
29     j=0
30     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()
```