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

Capítulo 1

Introducción

Siguiendo con el hecho de que el análisis numérico surge de que existen expresiones matemáticas que son muy difíciles de resolver exactamente, o que aún se desconoce una manera de hacerlo; es fácil deducir que debe existir alguna ecuación de una sola variable que cumpla con este criterio. Lo cual es cierto, pues cualquier ecuación que contenga dos o más funciones de distinto tipo, o ambas trascendentes sobre su variable independiente; se vuelven indespejables, ya que siempre se aplicará una o más funciones a la variable. Un ejemplo es la ecuación $x = e^x$, porque, como se puede observar; nunca se conseguirá obtener una función explícita para x. Por otro lado, existe una gran variedad de funciones que, aunque sean perfectamente resolvibles vía analítica; son complicadas o tardadas de resolver. Un ejemplo podría ser cualquier raíz de orden mayor al 2.

Por eso y más, se desarrolló una serie de algoritmos para poder encontrar soluciones numéricas a ecuaciones de una variable, sin importar cuál sea, y para comenzar, se verán los métodos cerrados. Los métodos cerrados para resolver ecuaciones de una variable, toman su nombre del hecho de que sólo pueden realizarse dentro de un intervalo cerrado definido por unos valores iniciales entre los cuales debe localizarse la raíz o , de otra forma, estos métodos nunca la encontrarán. Estos métodos son tres: el método gráfico, el método de bisección y el método de falsa posición.

1. Método gráfico: consiste en realizar una gráfica de la función que represente a la ecuación original, dejando como variable independiente a la incógnita del problema, y así, ver la intersección de ésta con el eje de la variable independiente; obteniendo, así, una aproximación a la raíz de la ecuación.
   Como se puede observar, este método toma la naturaleza de ser un método cerrado, pues se debe conocer el intervalo donde se encuentra la respuesta al problema para poder verla al momento de hacer la gráfica.

2. Método de bisección: este método consta de escoger un intervalo adecuado e ir dividiéndolo a la mitad, para luego tomar como nuevo intervalo, al subintervalo izquierdo o derecho del anterior, dependiendo del comportamiento de la función en dicho intervalo. Este método tiende a no estancarse; pero, por ser de fuerza bruta, tiende a ser ineficiente.

3. Método de falsa posición: es similar al método anterior, pero se vuelve

un poco más eficiente, ya que se va guiando a la respuesta, mediante una línea recta armada con los puntos de iteraciones anteriores; por lo que se puede ahorrar iteraciones en comparación del método de bisección.

Capítulo 2

Método y resultados

2.1. Método y pseudocódigo

Como se pudo observar, tanto el método de bisección como el de falsa posición solamente difieren en un paso: definir el resultado provisional. Por tanto, el pseudocódigo será, en esencia,el mismo:

```
//importar paquetes

                        main(){
                                //graficar la funcion
                                biseccion/falsaposicion()
                                }

                        biseccion/falsaposicion(){

                                //definir variables
                                //limite inferior xl
                                //limite superior xu
                                //resultado provisional xr
                                //Error e
                                //Contador de iteraciones i
                                //lista de iteraciones iteraciones
    []
                                //lista de error error[]
                                //Bandera para soluciones exactas
    flag=false
while (e>=errordeseado \&\& false!=true){

                                        //guardar el valor anterior
     de xr en una variable temp
                                        //calcular el error
    porcentual aproximado y guardarlo en e
                                        //guardar i en una lista de
     iteraciones
                                        i++;
                                        //guardar i en iteraciones
                                        //guardar e en error

                                        //obtener nuevo xr
                                        xr=(xl+xu)/2; //biseccion
                                        xr=xu-(f(xu)(xl-xu))/(f(xl)
    -f(xu)); //falsa posicion
                                        //comprobar la posicion de
    la raiz
if (f(xr)f(xl) < 0){
                                                xl=xr;
                                                }
```

```
                                        else if( f(xr)f(xl) > 0){
                                                xu=xr;
                                        }
                                        else{
                                                flag=true;
                                                }
                                }

                                //graficar iteraciones vs error
                                //guardar la grafica
                                print(xr);
                                return;

                        }
```

2.2. Problemas

De lo visto en el apartado anterior, los siguientes códigos serán más o menos extensos dependiendo de los métodos solicidatos; no obstante, contendrán una o más secciones de lo expuesto en el pseudocódigo anterior, ya que, es literalmente el mismo código; pero con valores iniciales y funciones diferentes; además de más o menos métodos. Inclusive, las funciones programadas para cada uno de los métodos es casi la misma, como ya se dijo.

2.2.1. 5.3

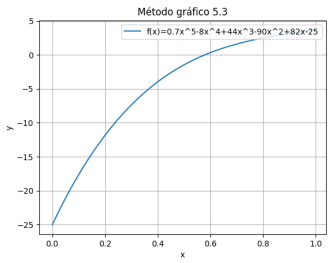
Determine las raíces reales de $f(x) = 0{,}7x^5 - 8x^4 + 44x^3 - 90x^2 + 82x - 25$:

- a: Gráficamente.
- b: Usando el método de bisección con $E_s = 10\,\%$. Utilice como valores iniciales xl=0.5 y xu=1.
- c: Realice el mismo cálculo que en b, pero con el método de la falsa posición y $E_s = 0{,}2\,\%$.

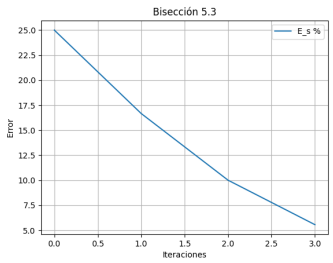
```
import math
import matplotlib.pyplot as plt

def main():

    x=[]
    y=[]
    i=0
    while(i<1):
        x.append(i)
        y.append(0.7*i**5-8*i**4+44*i**3-90*i**2+82*i-25)
        i+=0.01

    plt.plot(x,y,label='f(x)=0.7x^5-8x^4+44x^3-90x^2+82x-25')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.legend(loc='upper right')
    plt.grid(True)
    plt.title('M todo gr fico 5.3')
    plt.savefig('53graph.png')
```

```
21
22      bis()
23      fakepos()
24      return(0)
25
26  def bis():
27
28      xr=1
29      xl=0.5
30      xu=1
31      e=100
32      i=0
33      temp=1
34      flag=False
35      error=[]
36      iteraciones=[]
37
38
39      while(e>=10 and flag!=True):
40
41          temp=xr
42          xr=(xu+xl)/2
43
44          e=abs((xr-temp)/(temp))*100
45          error.append(e)
46          iteraciones.append(i)
47          i+=1
48          if ((0.7*xl**5-8*xl**4+44*xl**3-90*xl**2+82*xl-25)*(0.7*xr
    **5-8*xr**4+44*xr**3-90*xr**2+82*xr-25) < 0):
49              xu=xr
50          elif((0.7*xl**5-8*xl**4+44*xl**3-90*xl**2+82*xl-25)*(0.7*xr
    **5-8*xr**4+44*xr**3-90*xr**2+82*xr-25) > 0):
51              xl=xr
52          else:
53              flag=True
54      plt.clf()
55      plt.cla()
56      plt.plot(iteraciones,error,label='E_s %')
57      plt.xlabel('Iteraciones')
58      plt.ylabel('Error')
59      plt.legend(loc='upper right')
60      plt.grid(True)
61      plt.title('Bisecci n 5.3')
62      plt.savefig('53bis.png')
63
64      print(f'Bisecci n: {xr}')
65      return(0)
66
67  def fakepos():
68
69      xr=1
70      xl=0.5
71      xu=1
72      i=0
73      e=100
74      temp=1
75      flag=False
76      error=[]
77      iteraciones=[]
78
79      while(e>=10 and flag!=True):
80
81          temp=xr
```

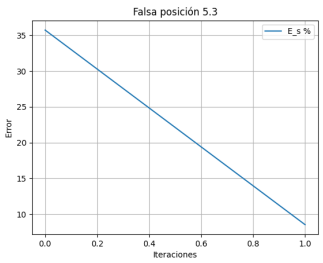
```
        xr=xu-((0.7*xu**5-8*xu**4+44*xu**3-90*xu**2+82*xu-25)*(xl-xu))/((0.7*xl**5-8*xl**4+44*xl**3-90*xl**2+82*xl-25)-(0.7*xu**5-8*xu**4+44*xu**3-90*xu**2+82*xu-25))

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

        if ((0.7*xl**5-8*xl**4+44*xl**3-90*xl**2+82*xl-25)*(0.7*xr**5-8*xr**4+44*xr**3-90*xr**2+82*xr-25) < 0):
            xu=xr
        elif((0.7*xl**5-8*xl**4+44*xl**3-90*xl**2+82*xl-25)*(0.7*xr**5-8*xr**4+44*xr**3-90*xr**2+82*xr-25) > 0):
            xl=xr
        else:
            flag=True
    plt.clf()
    plt.cla()
    plt.plot(iteraciones,error,label='E_s %')
    plt.xlabel('Iteraciones')
    plt.ylabel('Error')
    plt.legend(loc='upper right')
    plt.grid(True)
    plt.title('Falsa posici n 5.3')
    plt.savefig('53fp.png')

    print(f'Posici n falsa: {xr}')
    return(0)



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



(a) Método gráfico



(b) Error vs. iteraciones: bisección



(c) Error vs. iteraciones: falsa posición

2.2.2. 5.11

Determine la raíz real de x^3,5 = 80:

- a: de manera analítica.
- b: con falsa posición con $E_s = 2{,}5\,\%$.Haga elecciones iniciales de 2 a 5.

```
import math
import matplotlib.pyplot as plt

```

```
4
5  def main():
6
7      fakepos()
8      return(0)
9
10
11 def fakepos():
12
13     xr=1
14     xl=2
15     xu=5
16     i=0
17     e=100
18     temp=1
19     flag=False
20     error=[]
21     iteraciones=[]
22
23     while (e>=2.5 or flag==True):
24
25         temp=xr
26         xr=xu-((xu**(7/2)-80)*(xl-xu))/((xl**(7/2)-80)-(xu**(7/2)
       -80))
27         e=abs((xr-temp)/(temp))*100
28         error.append(e)
29         iteraciones.append(i)
30         i+=1
31
32         if ((xl**(7/2)-80)*(xr**(7/2)-80) < 0):
33             xu=xr
34         elif((xl**(7/2)-80)*(xr**(7/2)-80) > 0):
35             xl=xr
36         else:
37             flag=True
38
39     plt.clf()
40     plt.cla()
41     plt.plot(iteraciones,error,label='E_s %')
42     plt.xlabel('Iteraciones')
43     plt.ylabel('Error')
44     plt.legend(loc='upper right')
45     plt.grid(True)
46     plt.title('Falsa posici n 5.11')
47     plt.savefig('511fp.png')
48
49     print(f'Falsa posici n: {xr}')
50     return(0)
51
52
53
54 if __name__ == '__main__':
55     main()
```

2.2.3. 5.13

La velocidad v de un paracaidista que cae, está dada por:

$$v = \frac{gm}{c}\left(1 - e^{-(c/m)t}\right) \tag{2.1}$$

Donde g=9.81m/s^2, c=15kg/s, calcule la masa m de modo que la velocidad sea v=36m/s en t=10s. Utilice el método de la falsa posición para determinas m a un nivel de $E_s = 0{,}1\,\%$.

```
import math
import matplotlib.pyplot as plt



def main():
    fakepos()
    return(0)

def fakepos():

    xr=1
    xl=40
    xu=80
    e=100
    i=0
    temp=1
    flag=False
    error=[]
    iteraciones=[]

    while(e>=0.001 or flag!=True):

        temp=xr
        xr=xu-((36-((9.81*xu)/15)*(1-math.e**-(150/xu)))*(xl-xu))
   /((36-((9.81*xl)/15)*(1-math.e**-(150/xl)))-(36-((9.81*xu)/15)
   *(1-math.e**-(150/xu))))

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

        if (((36-((9.81*xl)/15)*(1-math.e**-(150/xl)))*(36-((9.81*
   xr)/15)*(1-math.e**-(150/xu)))) < 0):
            xu=xr
        elif(((36-((9.81*xl)/15)*(1-math.e**-(150/xl)))*(36-((9.81*
   xr)/15)*(1-math.e**-(150/xr)))) > 0):
            xl=xr
        else:
            flag=True

    plt.clf()
    plt.cla()
    plt.plot(iteraciones,error,label='E_s %')
    plt.xlabel('Iteraciones')
    plt.ylabel('Error')
    plt.legend(loc='upper right')
    plt.grid(True)
    plt.title('Falsa posici n 5.13')
    plt.savefig('513fp.png')
    print(f'Falsa posici n: {xr}')




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

2.2.4. 5.15

Como se ilustra den la figura P5.15, la velocidad del agua v (m/s), en la descarga de un tanque cilíndrico a través de un tubo largo, se puede calcular como:

$$v = \sqrt{2gH}\tanh\left(\frac{\sqrt{2gH}}{2L}t\right) \tag{2.2}$$

Determine la carga hidrostática H para obtener v=5m/s en 2.5s, con un tubo de 4m de largo. Hágalo gráficamente, por bisección y con posición falsa.Utilice xl=0 y xu=2 con un nivel de $E_s = 1\,\%$.

```
import math
import matplotlib.pyplot as plt


def main():

    x=[]
    y=[]
    i=0
    while(i<=2):
        x.append(i)
        y.append(5-math.sqrt(2*9.81*i)*math.tanh((math.sqrt(2*9.81*
   i)/2*4)*2.5))
        i+=0.01

    plt.clf()
    plt.cla()
    plt.plot(x,y,label='f(m)')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.legend(loc='upper right')
    plt.grid(True)
    plt.title('M todo gr fico 5.15')
    plt.savefig('515graph.png')
    bis()
    fakepos()
    return(0)


def bis():

    xr=1
    xl=0
    xu=2
    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
```

```
51
52          if ((5-math.sqrt(2*9.81*xl)*math.tanh((math.sqrt(2*9.81*xl)
    /2*4)*2.5))*(5-math.sqrt(2*9.81*xr)*math.tanh((math.sqrt(2*9.81*
    xr)/2*4)*2.5)) < 0):
53              xu=xr
54          elif((5-math.sqrt(2*9.81*xl)*math.tanh((math.sqrt(2*9.81*xl
    )/2*4)*2.5))*5-math.sqrt(2*9.81*xr)*math.tanh((math.sqrt(2*9.81*
    xr)/2*4)*2.5) > 0):
55              xl=xr
56          else:
57              flag=True
58
59              flag=True
60      plt.clf()
61      plt.cla()
62      plt.plot(iteraciones,error,label='E_s %')
63      plt.xlabel('Iteraciones')
64      plt.ylabel('Error')
65      plt.legend(loc='upper right')
66      plt.grid(True)
67      plt.title('Bisecci n 5.15')
68      plt.savefig('515bis.png')
69      print(f'Bisecci n: {xr}')
70      return(0)
71
72  def fakepos():
73
74      xr=1
75      xl=0
76      xu=2
77      i=0
78      e=100
79      temp=1
80      flag=False
81      error=[]
82      iteraciones=[]
83
84      while(e>=0.5 or flag!=True):
85
86          temp=xr
87          xr=xu-((5-math.sqrt(2*9.81*xu)*math.tanh((math.sqrt(2*9.81*
    xu)/2*4)*2.5))*(xl-xu))/((5-math.sqrt(2*9.81*xl)*math.tanh((math
    .sqrt(2*9.81*xl)/2*4)*2.5))-(5-math.sqrt(2*9.81*xu)*math.tanh((
    math.sqrt(2*9.81*xu)/2*4)*2.5)))
88
89          e=abs((xr-temp)/(temp))*100
90          error.append(e)
91          iteraciones.append(i)
92          i+=1
93
94          if ((5-math.sqrt(2*9.81*xl)*math.tanh((math.sqrt(2*9.81*xl)
    /2*4)*2.5))*(5-math.sqrt(2*9.81*xr)*math.tanh((math.sqrt(2*9.81*
    xr)/2*4)*2.5)) < 0):
95              xu=xr
96          elif((5-math.sqrt(2*9.81*xl)*math.tanh((math.sqrt(2*9.81*xl
    )/2*4)*2.5))*5-math.sqrt(2*9.81*xr)*math.tanh((math.sqrt(2*9.81*
    xr)/2*4)*2.5) > 0):
97              xl=xr
98          else:
99              flag=True
100
101
102     plt.clf()
```

```
    plt.cla()
    plt.plot(iteraciones,error,label='E_s %')
    plt.xlabel('Iteraciones')
    plt.ylabel('Error')
    plt.legend(loc='upper right')
    plt.grid(True)
    plt.title('Falsa posici n 5.15')
    plt.savefig('515fp.png')
    print(f'Falsa posici n: {xr}')



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

2.2.5. 5.17

Suponga que el lector está diseñando un tanque esférico para almacenar agua para un poblado pequeño en un país en desarrollo (México xD). El volumen del líquido que puede contener se calcula con :

$$V = \pi h^2 \frac{3R - h}{3} \tag{2.3}$$

Donde V es el volumen (m^3), h es la profundidad de agua en el tanque (m), y R, el radio del tanque en metros.

Si R=3m, ¿a qué profundidad debe llenarse el tanque de modo que contenga 30m^3?. Haga tres iteraciones con el método de falsa posición a fin de obtener la respuesta. Determine el error relativo aproximado después de cada iteración. Utilice 0 y R.

```
import math
import matplotlib.pyplot as plt



def main():
    fakepos()
    return(0)


def fakepos():

    xr=1
    xl=0
    xu=2
    e=100
    temp=1
    flag=False
    i=0
    error=[]
    iteraciones=[]
    #30-(math.pi*x**2)*(3*3-x)/3

    while(i<3):

        temp=xr
        xr=xu-((30-(math.pi*xu**2)*(3*3-xu)/3)*(xl-xu))/((30-(math.
   pi*xl**2)*(3*3-xl)/3)-(30-(math.pi*xu**2)*(3*3-xu)/3))
```

```
28
29          e=abs((xr-temp)/(temp))*100
30          error.append(e)
31          iteraciones.append(i)
32
33          if ((30-(math.pi*xl**2)*(3*3-xl)/3)*(30-(math.pi*xr**2)
    *(3*3-xr)/3) < 0):
34              xu=xr
35          elif((30-(math.pi*xl**2)*(3*3-xl)/3)*(30-(math.pi*xr**2)
    *(3*3-xr)/3) > 0):
36              xl=xr
37          else:
38              flag=True
39          i+=1
40
41
42      plt.clf()
43      plt.cla()
44      plt.plot(iteraciones,error,label='E_s %')
45      plt.xlabel('Iteraciones')
46      plt.ylabel('Error')
47      plt.legend(loc='upper right')
48      plt.grid(True)
49      plt.title('Falsa posici n 5.17')
50      plt.savefig('517fp.png')
51      print(f'Falsa posici n: {xr}')
52
53
54
55
56  if __name__ == '__main__':
57      main()
```

2.3. Resultados

Problema	Bisección	Falsa posición
5.3	0.59375	0.588017
5.11	NO	3.44349
5.13	NO	62.30246
5.15	1.25	1.27
5.17	NO	2.0268

Capítulo 3

Conclusiones

Capítulo 4

Referencias