Universidad Autónoma de Querétaro

FACULTAD DE INGENIERÍA

Tarea 3: Métodos cerrados

Análisis numérico

Autor:
J.A. Salinas Sánchez

Febrero 2022

Índice general

1. Introducción **2**
1.1. Métodos 2
1.1.1. Punto fijo 2
1.1.2. Newton-Rapshon 2
1.1.3. Secante 3

2. Métodología **4**
2.1. Pseudocódigo general 4
2.2. Pseudocódigo **punto fijo** 4
2.3. Pseudocódigo **Newton-Raphson** 5
2.4. Pseudocódigo **secante** 5
2.5. Código final 6

3. Ejercicios **8**
3.1. 6.1 8
3.1.1. Resultados 8
3.1.2. Error 8
3.2. 6.3 8
3.2.1. Resultados 8
3.2.2. Error 8
3.3. 6.5 8
3.3.1. Resultados 9
3.3.2. Error 9
3.4. 6.7 9
3.4.1. Resultados 9
3.4.2. Error 9
3.5. 6.13 9
3.5.1. Resultados 9
3.5.2. Error 9
3.6. 6.19 9
3.6.1. Resultados 10
3.6.2. Error 10

4. Conclusiones **11**

5. Referencias **12**

Capítulo 1

Introducción

Siguiendo con el arte de obtener respuestas de lo aparentemente irresolvible, seguimos con ecuaciones de una sola variable; pero con métodos más eficientes. La cuestión es que los métodos abiertos dependen de tener conocimiento previo, aunque sea vago, de la ubicación de las raíces; pues dependen de un intervalo inicial que encierre éstas. Entonces, todo se dificulta cuando la expresión a resolver es completamente desconocida o tiene raíces en valores muy grandes, ya sea positivos o negativos; ya que, no se puede localizar un intervalo tentativo para los métodos cerrados visualmente o analíticamente. Además, las estimaciones con un grado de incertidumbre tan alto llevan a establecer intervalos muy grandes, lo que resta aún más eficiencia a estos métodos. Y, para terminar, por el simpleme hecho depender de parámetros iniciales en lugar de la función $per\ se$, suelen realizar más iteraciones que los métodos abiertos.

Ahora bien, del párrafo anterior ya se puede intuir que los métodos abiertos, en contraposición con los métodos cerrados, son aquéllos que no dependen de intervalos iniciales donde se encuentre encerrada la raíz a buscar. También, se puede deducir que estos métodos son más eficientes que los métodos cerrados y que dependen únicamente de la función a estudiar. Y así es. Los métodos abiertos usan solamente la función cuyas raíces quieren ser descubiertas, para hallarlas. El cómo lo hagan, ya depende del método.

1.1. Métodos

1.1.1. Punto fijo

Este método consiste en conseguir una función $g(x) = f(x)$ a partir de una ecuación $f(x) = 0$ para ir iterando sobre ella; de modo que el valor anterior que da la función $g(x)$ se va asignando a x despejada. Esto es como despejar la equis del despeje anterior, cada iteración; lo que, tarde o temprano, va a llevar a un valor específico: la raíz.

1.1.2. Newton-Rapshon

Este método se aprovecha de la definición de recta tangente a un punto, la cual básicamente es una recta que para por un solo punto de una función. Para lograr eso, su pendiente necesita estar dada por la derivada de la función en ese punto y se aplica la la fórmula de la recta en su forma punto pendiente $y - y_0 = y'(x_0)(x - x_0)$. La lógica detrás de esto es que, al trazar una recta tangente a una función en un punto, ésta cruzará el eje x en algún punto, y al conseguir la recta tangente evaluada en ese mismo punto; uno se va acercando cada vez más a una raíz de la función.

1.1.3. Secante

Es igual que el método NR, excepto que utiliza rectas secantes en lugar de tangentes, para aproximar las raíces. Esto es útil, pues se dan pasos más grandes, lo que puede reducir el número de iteraciones si se tiene un factor de tolerancia menor; pero, al contrario, si el factor de tolerancia es mayor. No obstante, este mismo hecho hace que pueda encontrar raíces más fácilmente a funciones muy inclinadas o con indeterminaciones; pues llega más rápido, y al no depender de la derivada; tampoco depende de la derivabilidad de la función.

Capítulo 2

Métodología

2.1. Pseudocódigo general

```
main(){
  //pedir la funci n f
  //pedir x0
  //pedir e%
  //pedir m todo

  if(m todo==NR){
    NR()
  }
  else if(m todo==punto fijo){
    punto fijo()
  }
  else if(m todo=secante){
    secante()

  }


}



////funciones de los m todo
```

2.2. Pseudocódigo punto fijo

```
FUNCTION Fixpt(x0, es, imax iter, ea)
xr = x0
iter = 0
DO
xrold = xr
xr = g(xrold)
iter = iter + 1
lF xr     0 THEN
ea  x r   xrold
xr 100=
END IF
IF ea < es OR iter     imax EXIT
END DO
Fixpt = xr
END Fixpt
```

2.3. Pseudocódigo Newton-Raphson

```
NR(){
  f=obtenerfuncion()
  resultado=valorinicial
  //derivar la funci n
  //iniciar la variable error
  while(error>error deseado){
    //asignar resultado a una variable temporal
    resultado=x-f(temporal)/f'(temporal)
    //calcular el nuevo error

  }
  return resultado

}
```

2.4. Pseudocódigo secante

```
def secant(f,a,b,N):
    '''Approximate solution of f(x)=0 on interval [a,b] by the secant method.

    Parameters
    ----------
    f : function
        The function for which we are trying to approximate a solution f(x)=0.
    a,b : numbers
        The interval in which to search for a solution. The function returns
        None if f(a)*f(b) >= 0 since a solution is not guaranteed.
    N : (positive) integer
        The number of iterations to implement.

    Returns
    -------
    m_N : number
        The x intercept of the secant line on the the Nth interval
            m_n = a_n - f(a_n)*(b_n - a_n)/(f(b_n) - f(a_n))
        The initial interval [a_0,b_0] is given by [a,b]. If f(m_n) == 0
        for some intercept m_n then the function returns this solution.
        If all signs of values f(a_n), f(b_n) and f(m_n) are the same at any
        iterations, the secant method fails and return None.

    Examples
    --------
    >>> f = lambda x: x**2 - x - 1
    >>> secant(f,1,2,5)
    1.6180257510729614
    '''
    if f(a)*f(b) >= 0:
        print("Secant method fails.")
        return None
    a_n = a
    b_n = b
    for n in range(1,N+1):
        m_n = a_n - f(a_n)*(b_n - a_n)/(f(b_n) - f(a_n))
```

```
        f_m_n = f(m_n)
        if f(a_n)*f_m_n < 0:
            a_n = a_n
            b_n = m_n
        elif f(b_n)*f_m_n < 0:
            a_n = m_n
            b_n = b_n
        elif f_m_n == 0:
            print("Found exact solution.")
            return m_n
        else:
            print("Secant method fails.")
            return None
    return a_n - f(a_n)*(b_n - a_n)/(f(b_n) - f(a_n))
```

2.5. Código final

```
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():
    l=['Newton-Raphson[NR]','Punto fijo[PF]','Secante[Sec]']
    func=str(input('Introduzca su funci n: '))
    print(l)
    metodo=str(input('Seleccione el m todo para encontrar una ra z: ')
            )
    x0=float(input('Ahora introduzca su valor inicial: '))
    emax=float(input('Finalmente, introduzca el factor de tolerancia: '))
    if(metodo=="NR"):
        print(NR(func,x0,emax))
    elif(metodo=="PF"):
        print(PF(func,x0,emax))
    elif(metodo=="Sec"):
        print(Sec(func,x0,emax))

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()
```

Capítulo 3

Ejercicios

3.1. 6.1

Utilice la iteración simple de punto fijo para localizar la raíz de:

$$f(x) = \sin\left(\sqrt{(x)}\right) - x \tag{3.1}$$

Haga una elección inicial de x0 = 0.5 e itere hasta que ea 0.01Compruebe que el proceso converge en forma lineal según se des- cribió en el cuadro 6.1.

3.1.1. Resultados

3.1.2. Error

3.2. 6.3

Utilice los métodos de: (a) iteración de punto fijo y (b) Newton-Raphson, para determinar una raíz de f(x) = –0.9x2 + 1.7x + 2.5 con el uso de x0 = 5. Haga el cálculo hasta que ea sea menor que $es = 0{,}01\,\%$. Asimismo, realice una comprobación del error de su respuesta final.

3.2.1. Resultados

3.2.2. Error

3.3. 6.5

Emplee el método de Newton-Raphson para determinar una raíz real de f(x) = –1 + 5.5x – 4x2 + 0.5x3 con el uso de valores ini- ciales de (a) 4.52 y (b) 4.54. Estudie y use métodos gráficos y analíticos para explicar cualquier peculiaridad en sus resultados.

3.3.1. Resultados

3.3.2. Error

3.4. 6.7

Localice la primera raíz positiva de:

$$f(x) = senx + cos(1 + x2)–1 \tag{3.2}$$

donde x está en radianes. Para localizar la raíz, use cuatro iteraciones del método de la secante con valores iniciales de (a) xi – 1 = 1.0 y x= 3.0; (b) xi – 1 = 1.5 y xi = 2.5, (c) xi – 1 = 1.5 y xi = 2.25, (d) use el método gráfico para explicar su resultado.

3.4.1. Resultados

3.4.2. Error

3.5. 6.13

Usted debe determinar la raíz de la siguiente función fácilmente diferenciable:

$$e^{0,5x} = 5 - 5x \tag{3.3}$$

Elija la mejor técnica numérica, justifique su elección y luego use esa técnica para determinar la raíz. Observe que se sabe que, para valores iniciales positivos, todas las técnicas, salvo la iteración de punto fijo, finalmente convergen. Realice iteraciones hasta que el error relativo aproximado caiga por debajo de 2%. Si usted emplea un método cerrado, use valores iniciales de $xl = 0$ y $xu = 2$. Si escoge el método de Newton-Raphson o el modificado de secante, use un valor inicial de xi = 0.7. Si se decide por el método de secante, use valores iniciales de xi – 1 = 0 y xi = 2.

3.5.1. Resultados

3.5.2. Error

3.6. 6.19

Suponga el lector que está diseñando un tanque esférico (véase la figura P6.19) de almacenamiento de agua para un poblado peque- ño de un país en desarrollo. El volumen del líquido que puede con- tener se calcula con:

$$V = \pi h^2 \frac{3R - h}{3} \tag{3.4}$$

donde V = volumen (m3), h = profundidad del agua en el tanque (m) y R = radio del tanque (m). Si R = 3 m, ¿a qué profundidad debe llenarse el tanque de modo que contenga 30 m3? Haga tres iteraciones del método de Newton- Raphson para determinar la respuesta. Encuentre el error relativo aproximado después de cada iteración. Observe que el valor inicial de R convergerá siempre.

3.6.1. Resultados

3.6.2. Error

Capítulo 4

Conclusiones

Como se mencionó en la introducción, el hecho de que estos métodos dependan únicamente de la función y un valor inicial escogido arbitrariamente; ya que, si bien, una buena elección de un valor inicial eficienta el cómputo; la elección uno inadecuado no inutilizará el algoritmo como lo hace con los cerrados; hace que sean mucho más fiables en la resolución de cualquier problema que tenga de por medio una función arbitraria $f(x)$. Porque, es lo único que se tiene que saber: la función y ya está; el número puede ser cualquiera que se te ocurra. Y el hecho de poder hallar la raíz de cualquier función, teniendo opciones que te pueden hasta hacer prescindir de la derivabilidad de una función; te permite, literalmente, poder resolver cualquier ecuación de una variable, mientras cumpla con los axiomas de la igualdad y otras propiedades matemáticas.

Bibliografía

() hapra. S.C et Canale R.P (2015) *Métodos numéricos para ingenieros.* Mc-GrawHill. pp (113-124)&(135-136)

(2) alls, P., 2022. Secant method. Secant Method - Mathematical Python. Available at: https://personal.math.ubc.ca/ pwalls/math-python/roots-optimization/secant/ (Accessed February 10, 2022).