Actividad 2 Metodos Numericos

ROMMEL NICOLAS ZAMBRANO GAONA

01 Julio Del 2021

1. EJERCICIOS

1. **Cree un repositorio en GitHub para los proyectos que se desarrollarán en el semestre. El repositorio debe estar asociado a su correo institucional. Los ejercicios de la actividad deben estar en el repositorio.**

   - Adjunto el link de mi repositorio donde estan mis programas https//github.com/RommelZambrano/ 1.git

2. **Utilice todos los métodos estudiados (Newton, Bisección y Secante) para encontrar la raíz de la siguiente función:** $f(x) = x + cosx$ **en el intervalo** $[-2, 0]$

METODO NEWTON

$f(x) = x + cos(x)$

$x_0 = 0$

$i = 0, 1, 2, 3$

PROCESO

Iteracion $i = 0$

$f(x_i) = f(x_0) = f(10) \longrightarrow cos(0) = 1$

$f^{'}x_i = f^{'}(x_0) = f^{'}(0) \longrightarrow -1(0) = 1$

$x_1 = x_0 + \frac{f(x_0)}{f^{'}(x_0)}$

$x_1 = 0 + \frac{1}{1} = 1$

iteracion 1

$f(x_1) = f(x_1) = f(-1) \longrightarrow -1 + cos(-1) = -0{,}460$

$f^{'}(x_1) = f^{'}x_1 = f^{'}(-1) \longrightarrow --(-1) = 1{,}841$

$x_2 = x_1 + \frac{f(x_1)}{f'(x_1)}$

$x_2 = -1 - \frac{-0{,}460}{1{,}841} = -0{,}750$

iteracion 2

$f(x_1) = f(x_2) = f(-0{,}750) \longrightarrow -0{,}018$

$f'(x_1) = f' x_2 = f'(-0{,}756) \longrightarrow 1{,}681$

$x_3 = x_2 + \frac{f(x_2)}{f'(x_2)}$

$x_3 = -0{,}750 \frac{-0{,}018}{1{,}681} = -0{,}739$

iteracion 3

$f(x_1) = f(x_3) = f(-0{,}739) \longrightarrow 0{,}0001$

$f'(x_1) = f' x_3 = f'(-0{,}739) \longrightarrow 1{,}673$

$x_4 = x_3 + \frac{f(x_3)}{f'(x_3)}$

$x_4 = -0{,}739 - \frac{0{,}0001}{1{,}673} = -0{,}739$

METODO SECANTE

$f(x_i - 1) = f(x_0 = f(-2) = -2 + cos(-2) \longrightarrow -2{,}416$

$f(x_1) = f(x_1 = f(0) = cos(0) = 1$

$x_2 = x_1 - \frac{f(x_1)(x_0 - x_1)}{x_0 - x_1}$

$x_2 = 0 - \frac{1(-2-0)}{-2{,}416-1}$

$x_2 = -0{,}585$

iteracion 2

$f(x_i) = f(x_2) = +(-0{,}585) \longrightarrow -0{,}249$

$f(x_{i-1}) = f(x_1 = f(0) = 1$

$x_3 = x_2 - \frac{f(x_2)(x_1 - x_0)}{x_1 - x_2}$

$x_2 = 0 - \frac{(0{,}249)+(0{,}585)}{1-0{,}249}$

$x_2 = -0{,}779$

iteracion 3

$f(x_{i-1}) = f(x_2) = f(-0{,}585)) \longrightarrow -0{,}249$

$f(x_i) = f(x_3 = f(-0{,}779) = -0{,}067$

$x_4 = x_3 - \frac{f(x_3)(x_2 - x_3)}{x_2 - x_3}$

$x_4 = -0{,}779 - \frac{(-0{,}067)(-0{,}585 + 0{,}779)}{0{,}249 + 0{,}067}$

$x_2 = -0{,}738$

iteracion 4

$f(x_{i-1}) = f(x_3) = f(-0{,}779)) \longrightarrow -0{,}067$

$f(x_i) = f(x_4 = f(-0{,}738) = -0{,}002$

$x_5 = x_4 - \frac{f(x_4)(x_3 - x_4)}{x_3 - x_4}$

$x_5 = -0{,}738 - \frac{(-0{,}002)(-0{,}779 + 0{,}738)}{-0{,}067 - 0{,}002}$

$x_5 = -0{,}738$

METODO BISECCION

Por el tema de este metodo contiene muchas iteraciones he hecho la iteracion solucion que en este caso seria la 21.

$c = \frac{a+b}{2} = \frac{-0{,}743 - 0{,}735}{2} = -0{,}739$

$f(a) = f(-0{,}743) = -0{,}006$

$f(b) = f(-0{,}735) = 0{,}007$

$f(c) = f(-0{,}739) = 0{,}000$

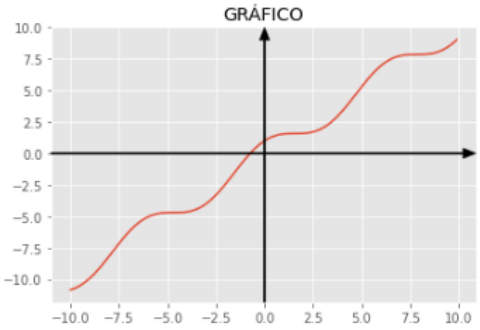
$f(b) * f(c) < 0$

Raiz de la funcion $f(x) = x + cos(x) \longrightarrow -0{,}739$

CODIGOS DE CADA METODO

Metodo de newton

```
%matplotlib inline
from matplotlib import pyplot as plt
from matplotlib import style
import matplotlib.pyplot as mp
import numpy as np
x=np.arange(-10,10,0.1)
y=x+np.cos(x)
fig, ax = plt.subplots()
ax.plot(x,y)
ax.set_title("GRÁFICO")
xmin,xmax=ax.get_xlim()
ymin,ymax=ax.get_ylim()
ax.grid(True, linestyle="-")
ax.annotate("",xy=(xmax,0),xytext=(xmin,0),
            arrowprops=dict(color="black",width=1.5,headwidth=8,headlength=10))
ax.annotate("",xy=(0, ymax), xytext=(0, ymin),
            arrowprops=dict(color="black",width=1.5,headwidth=8,headlength=10))
plt.show()
def f(x):
    return x+np.cos(x)
def df(x):
    return 1-np.sin(x)
def newton(f,df,x0,nmax,tol):
    xi=x0
    for i in range(0,nmax):
        xi=xi-f(xi)/df(xi)
        if abs(f(xi))<tol:
            print("Cálculo más cercano de la raiz es: ",xi,)
            print("Número de interaciones igual a: ""=",i,)
            break
    return None
newton(f,df,1,20,1e-6)
```



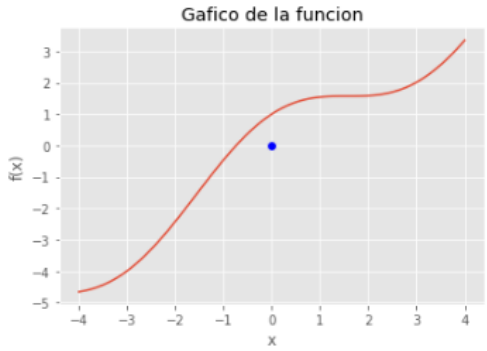

```
Cálculo más cercano de la raiz es:  -0.7390851385832758
Número de interaciones igual a: = 6
```

Metodo Secante

```
%matplotlib inline
from matplotlib import pyplot as plt
from matplotlib import style
import matplotlib.pyplot as mp
import numpy as np

def grafica(x,y,x_n):
        style.use("ggplot")
        plt.title("Gafico de la funcion")
        plt.ylabel("f(x)")
        plt.xlabel("x")
        plt.scatter(x_n,0,color='blue')
        plt.plot(x,y, linestyle='solid')

        plt.show()
x_n = 0
ea = 1000
n = 1
x = np.linspace(-4, 4, num = 100)
y = x+np.cos(x)

def secante(f,x0,x1,nmax,tol):
    for i in range(0,nmax):
        x2=(x0*f(x1)-x1*f(x0))/(f(x1)-f(x0));
        if abs(x2-x1) < tol:
            print('Contiene', i ,' iteraciones')
            print(f'Calculo mas sercano a la raiz es ',x2,'')
            break
        else:
            x0=x1
            x1=x2

    return None
```

```
f = lambda x:  x+np.cos(x)

ri= secante (f,-2,0,20,10**(-6))
grafica(x,y,x_n)
```

```
Contiene 5  iteraciones
Calculo mas sercano a la raiz es  -0.7390851332151506
```



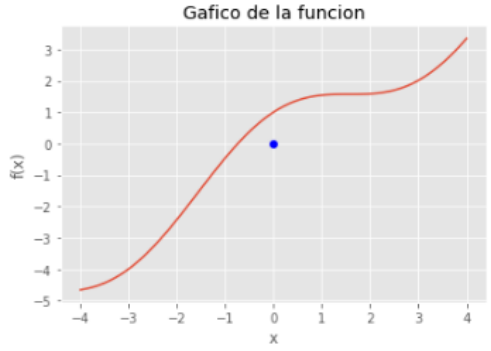
Metodo Biseccion

```
def grafica(x,y,x_n):
        style.use("ggplot")
        plt.title("Gafico de la funcion")
        plt.ylabel("f(x)")
        plt.xlabel("x")
        plt.scatter(x_n,0,color='blue')
        plt.plot(x,y, linestyle='solid')

        plt.show()
x_n = 0
ea = 1000
n = 1
x = np.linspace(-4, 4, num = 100)
y = f(x)



f = lambda x:  x+np.cos(x)
```

```python
def biseccion(a,b,tol):
    m=a
    c=b
    k=0

    if(f(a)*f(b)>0):
        print ("\nla funcion no vambia de signo")
        print()

    while (abs(m-c)>tol):
        m=c
        c=(a+b)/2
        if(f(a)*f(c)<0):
            b=c
        if(f(c)*f(b)<0):
            a=c

        print ("El intervalo es: [",a,",",b,"]")
        k=k+1
        print ()
    print("resultado de la raiz: ")
    print("Numero de iteraciones es ",k,"Calculo mas sercano a la raiz es ",c,)
biseccion (-2,0,1e-6)

grafica(x,y,x_n)
```

```
El intervalo es: [ -1.0 , 0 ]

El intervalo es: [ -1.0 , -0.5 ]

El intervalo es: [ -0.75 , -0.5 ]

El intervalo es: [ -0.75 , -0.625 ]

El intervalo es: [ -0.75 , -0.6875 ]

El intervalo es: [ -0.75 , -0.71875 ]

El intervalo es: [ -0.75 , -0.734375 ]

El intervalo es: [ -0.7421875 , -0.734375 ]

El intervalo es: [ -0.7421875 , -0.73828125 ]

El intervalo es: [ -0.740234375 , -0.73828125 ]

El intervalo es: [ -0.7392578125 , -0.73828125 ]

El intervalo es: [ -0.7392578125 , -0.73876953125 ]

El intervalo es: [ -0.7392578125 , -0.739013671875 ]

El intervalo es: [ -0.7391357421875 , -0.739013671875 ]

El intervalo es: [ -0.7391357421875 , -0.73907470703125 ]

El intervalo es: [ -0.739105224609375 , -0.73907470703125 ]

El intervalo es: [ -0.7390899658203125 , -0.73907470703125 ]

El intervalo es: [ -0.7390899658203125 , -0.7390823364257812 ]

El intervalo es: [ -0.7390861511230469 , -0.7390823364257812 ]
```

```
El intervalo es: [ -0.7390861511230469 , -0.7390842437744141 ]

El intervalo es: [ -0.7390851974487305 , -0.7390842437744141 ]

resultado de la raiz:
Numero de iteraciones es  21 Calculo mas sercano a la raiz es  -0.7390851974487305
```



3. **Encuentre la intersección de las siguientes funciones:** $f(x) = 3-x$ **y** $g(x) = ln\, x$**, con tres cifras decimales.**

$f(x) = 3 - x,\ g(x) = ln\, x$

$$ln\, x + x - 3$$

$$h(x) = ln\, x + x - 3$$

$$h(x) = \tfrac{1}{x} + 1 - 0$$

$$h'(x) = \tfrac{1}{x} + 1$$

Raices de h $x_0 = 2;\ h = 0$

$$h(2) = ln\, 2 + 2 - 3$$

$$= -0{,}3068$$

$$h'(2) = \tfrac{1}{2} + 1$$

$$h'(2) = \tfrac{3}{2} = 1{,}5$$

Newton

$$x_1 = 2 - (\tfrac{-0{,}3068}{1{,}5})$$

$$x_1 = 0{,}204533 \longrightarrow 2{,}204$$

Ahora $h = 1;\ x_1 = 2{,}204$

$$h_{(2{,}204)} = ln(2{,}204) + 2{,}204 - 3 \longrightarrow -0{,}005$$

$$h'(2{,}204) = \frac{1}{2{,}204+1} = 1{,}453$$

$$x_2 = 2{,}204 - (\frac{-0{,}005}{1{,}454}) \longrightarrow x_2 = 2{,}208$$

Ahora para $h = 2$; $x_3 = 2{,}208$

$$h(2{,}208) = ln(2{,}208) + 2{,}208 - 3 = 0{,}0008 \longrightarrow 0{,}000$$

$$h'(2{,}208) = \frac{1}{2{,}208} + 1 = 1{,}453$$

Ahora evaluamos h_3

$$x_3 = x_2 - \frac{h(x_2)}{h'(x_2)}$$

$$2{,}208 - 0 \longrightarrow 2{,}208$$

Remplazamos

$$f(x) = 3 - (x)$$

$$f_{(2{,}208)} = 3 - 2{,}208 \longrightarrow 0{,}792$$

La interseccion de las funciones se da en $(2{,}208 \Longleftrightarrow 0{,}792)$

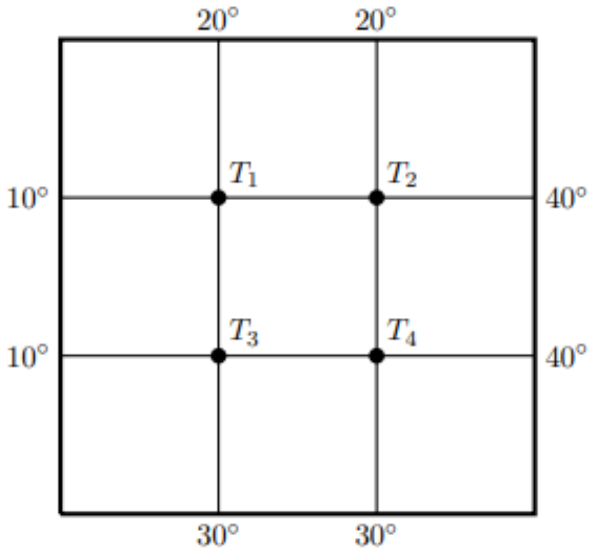
4. **Programe el método de sustitución regresiva. Analice el número de operaciones del algoritmo para matrices de tamaño $3x3$.**

```
%matplotlib inline
from matplotlib import pyplot as plt
from matplotlib import style
import matplotlib.pyplot as mp
import numpy as np
#Insertamos la matriz 3x3 cualquiera
A=np.array([[2,4,-6],[4,5,-8],[2,3,4]])
# Terminos independientes
b=np.array([1,2,3])

n=np.size(b)

x=np.zeros(n)

for i in range(n-1,-1,-1):
    sustitucion=0
    for j in range(i+1,n):
        sustitucion+=A[i,j]*x[j]
    x[i]=(b[i]-sustitucion)*1/A[i,i]
print(x)
```

```
[-0.45  1.6   0.75]
```

5. **Una importante parte de la física es la termodinámica que es el estudio de la transferencia de calor. En este ejercicio vamos a determinar la distribución de temperatura de estado estable de una placa delgada cuando se conoce la temperatura en los bordes. Suponga que la placa que se ilustra en la figura representa una sección transversal de una viga de metal, con flujo de calor despreciable en la dirección perpendicular a la placa. Sean T_1, T_2, T_3 y**

T_4 las temperaturas en los cuatro nodos interiores de la malla en la figura. La temperatura en un nodo es aproximadamente igual al promedio de las temperaturas de los cuatro nodos más cercanos, esto es, a la izquierda, arriba, a la derecha y abajo. Escriba un sistema de ecuaciones cuya solución de estimaciones de las temperaturas T_1, T_2, T_2, T_3 y T_4



TEMPERATURA 1

$T_1 = \frac{10+T_2+20+T_3}{4}$

$4T_1 = 30 + T_2 + T_3$

$4T_1 - T_2 - T_3 = 30$

TEMPERATURA 2

$T_2 = \frac{T_1+20+40+T_4}{4}$

$4T_2 = 60 + T_1 + T_4$

$-T_1 + 4T_2 - T_4 = 60$

TEMPERATURA 3

$T_3 = \frac{T_1+10+T_4+30}{4}$

$4T_3 = 40 + T_1 + T_4$

$-T_1 + 4T_3 - T_4 = 40$

TEMPERATURA 4

$T_4 = \frac{T_3+T_2+40+30}{4}$

$-T_2 + T_3 - 4T_4 = 70$

$4T_1 - T_2 - T_3 = 30$

$-T_1 + 4T_2 - T_4 = 60$

$-T_1 + 4T_3 - T_4 = 40$

$-T_2 + T_3 - 4T_4 = 70$

$$\begin{pmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{pmatrix} \begin{pmatrix} \mathrm{T}_1 \\ T_2 \\ T_3 \\ T_4 \end{pmatrix} = \begin{pmatrix} 30 \\ 60 \\ 40 \\ 70 \end{pmatrix}$$ AL RESOLVER OBTENEMOS:

$=[T_1 \longrightarrow 20; T_2 \longrightarrow \frac{55}{2}; T_3 \longrightarrow \frac{45}{2}; T_4 \longrightarrow 30]$

SOLUCION GENERAL $\begin{pmatrix} 20 \\ \frac{55}{2} \\ \frac{45}{2} \\ 30 \end{pmatrix} \longrightarrow \begin{pmatrix} 20 \\ 27{,}5 \\ 22{,}5 \\ 30 \end{pmatrix}$

Programe el método de eliminación gaussiana y halle el valor de estas temperaturas. Utilice como apoyo el ejercicio del literal anterior.

```
%matplotlib inline
from matplotlib import pyplot as plt
from matplotlib import style
import matplotlib.pyplot as mp
import numpy as np
# Ingresamos la Matriz
A = np.array([[4., -1., 0., -1.],
              [-1., 4., -1., 0.],
              [0., -1., 4., -1.],
              [-1., 0., -1., 4.]])
B = np.array([[30.],
              [60.],
              [70.],
              [40.]])

A2=np.copy(A)
B2=np.copy(B)
n = len(B)

print(A)
print(B)
print("")
print("Elimincacion sin piviteo")
for k in range(0, n-1):
    for i in range(k+1, n):
        factor = A[i, k]/A[k, k]
        for j in range(k, n-1):
            A[i, j] -= factor*A[k, j]
        B[i] -= factor*B[k]
x= np.zeros(n)

x[n-1]= B[n-1]/A[n-1, n-1]
for i in range(n-2, -1, -1):
    suma_j=0
    for j in range(i+1, n):
        suma_j += A[i,j] * x[j]
    x[i] = (B[i] - suma_j)/A[i,i]
```

```
print(A)
print(B)
print(x)
print(A2)
print(B2)
print("\x1b[1;28m""La resta es: " , np.linalg.norm(np.dot(A, x)-B))
print("\x1b[1;28m""La solución del ejercicio es: x = ",
     np.linalg.solve(A, B))
```

```
[[ 4. -1.  0. -1.]
 [-1.  4. -1.  0.]
 [ 0. -1.  4. -1.]
 [-1.  0. -1.  4.]]
[[30.]
 [60.]
 [70.]
 [40.]]

Elimincacion sin piviteo
[[ 4.         -1.          0.         -1.        ]
 [ 0.          3.75       -1.          0.        ]
 [ 0.          0.          3.73333333 -1.        ]
 [ 0.          0.          0.          4.        ]]
[[30.        ]
 [67.5       ]
 [88.        ]
 [77.14285714]]
[18.7372449  25.66326531 28.7372449  19.28571429]
[[ 4. -1.  0. -1.]
 [-1.  4. -1.  0.]
 [ 0. -1.  4. -1.]
 [-1.  0. -1.  4.]]
[[30.]
 [60.]
 [70.]
 [40.]]
La resta es:  123.48126355435302
La solución del ejercicio es: x =  [[18.7372449 ]
 [25.66326531]
 [28.7372449 ]
 [19.28571429]]
```

6. **Programe el pivoteo parcial en el algoritmo de eliminación gaussiana. Utilice este nuevo programa para resolver el ejercicio anterior. Compare los resultados.**

```python
import numpy as np
#Datos anteriro matriz dada
A = np.array([[4, -1, 0, -1],
              [-1, 4, -1, 0],
              [0, -1, 4, -1],
              [-1, 0, -1, 4]])
B = np.array([[30],
              [60],
              [70],
              [40]])
#pivoteo
casi_cero = 1e-15
A = np.array(A, dtype=float)
AB = np.concatenate((A, B), axis=1)
# Pivoteo por cada fila
tam = np.shape(AB)
n = tam[0]
m = tam[1]

for i in range(0, n - 1, 1):
    columna = abs(AB[i:, i])
    donde_max = np.argmax(columna)
    if (donde_max != 0):
# Intercambiamos las filas
        temporal = np.copy(AB[i, :])
        AB[i, :] = AB[donde_max + i, :]
        AB[donde_max + i, :] = temporal
AB1 = np.copy(AB)
#Eliminacion Gausiana con pivoteo
for i in range(0, n - 1, 1):
    pivote = AB[i, i]
    adelante = i + 1
    for k in range(adelante, n, 1):
        factor = AB[k, i] / pivote
        AB[k, :] = AB[k, :] - AB[i, :] * factor
```

```python
ult_fila = n - 1
ult_columna = m - 1
X = np.zeros(n, dtype=float)
for i in range(ult_fila, 0 - 1, -1):
    suma = 0
    for j in range(i + 1, ult_columna, 1):
        suma = suma + AB[i, j] * X[j]
    B1 = AB[i, ult_columna]
    X[i] = (B1 - suma) / AB[i, i]
X = np.transpose([X])
print("\x1b[1;28m"+"Metodo de Pivoteo Parcial Por Filas")
print(AB1)
print("\x1b[1;28m"+"La Solución del ejercicio es: ")
print(X)
```

```
Metodo de Pivoteo Parcial Por Filas
[[ 4. -1.  0. -1. 30.]
 [-1.  4. -1.  0. 60.]
 [ 0. -1.  4. -1. 70.]
 [-1.  0. -1.  4. 40.]]
La Solución del ejercicio es:
[[20. ]
 [27.5]
 [30. ]
 [22.5]]
```