Tarea 1

Gustavo Hernández Angeles[1]

[1] *Universidad Autonoma de Nuevo Leon*

Publication date: 03/02/2023

Abstract— En esta tarea se encontrarán soluciones a ecuaciones que no poseen una solución analítica mediante varios métodos numéricos, entre ellos: Bisección, Newton-Raphson, Secante, etc. Además, se desarrollará un programa en FORTRAN para que, dado un intervalo, detecta las raíces que hay y entre qué puntos dados por un grado de precisión a elección. En la última página se muestran los códigos para los métodos numéricos utilizados.

Keywords— Solución numérica

I. Solución de ecuaciones sin solución analítica

a. *Primer problema*

Resolver las siguientes ecuaciones numéricamente usando el método de Newton-Raphson.

1. ¿Cuál es la raíz de 8?. Resolver numéricamente.
2. $-0.8x^4 + 6.6x^3 - 16x^2 + 11.7x + 10 = 0$
3. $xe^{0.5x} + 1.2x - 5 = 0$
4. $\cos x \cosh x + 1 = 0$
5. $x - 2e^{-x} = 0$

SOLUCIÓN

1. Calcular la raíz de 8.

Como primer paso para cada ejercicio, lo que haremos será graficar la función para asegurar un punto donde el algoritmo vaya a converger. Para este caso, creamos una función auxiliar $f(x) = x^3 - 8$ con el objetivo de hallar la raíz y por tanto, la solución al problema. La gráfica se muestra:

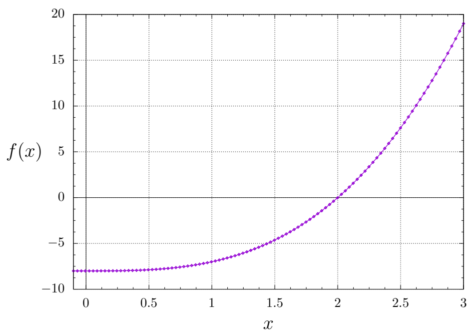
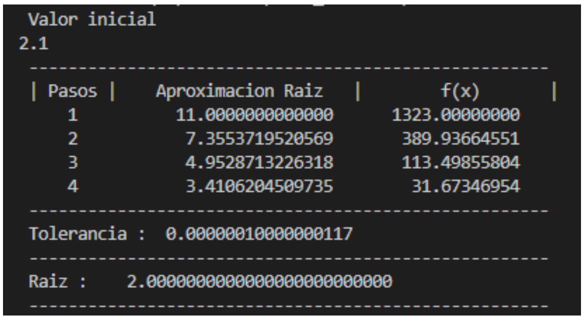


Fig. 1: Gráfica $f(x) = x^3 - 8$.

Podemos observar que la ecuación tiene una única solución en los reales alrededor del 2, por lo tanto en nuestro código pondremos un valor cercano. Corriendo el programa con un valor inicial de 2.1:



```
 Valor inicial
2.1
 --------------------------------------------------
 | Pasos |     Aproximacion Raiz     |     f(x)      |
     1         11.0000000000000       1323.00000000
     2          7.3553719520569        389.93664551
     3          4.9528713226318        113.49855804
     4          3.4106204509735         31.67346954
 --------------------------------------------------
 Tolerancia :  0.00000010000000117
 --------------------------------------------------
 Raiz :    2.00000000000000000000000000
 --------------------------------------------------
```

Fig. 2: Solución numérica a $f(x) = 0$.

$$\therefore 8^{1/3} = 2 \tag{1}$$

2. $-0.8x^4 + 6.6x^3 - 16x^2 + 11.7x + 10 = 0$

Para este polinomio, procedemos a graficarlo para visualizar valores aproximados de las raíces y ejecutar el programa alrededor de estos.

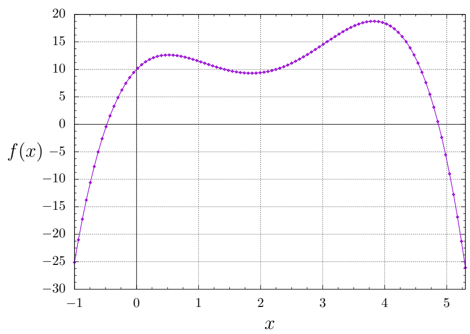


Fig. 3: Gráfica $f(x) = -0.8x^4 + 6.6x^3 - 16x^2 + 11.7x + 10$.

Observamos que las raíces aproximadamente son $x_1 \sim -0.5$ y $x_2 \sim 5$. Por lo que seleccionaremos los valores $x_1 = -1$ y $x_2 = 5$, respectivamente, para evaluar las dos raíces numéricamente. Los resultados son:

```
Valor inicial
-1
---------------------------------------------------
| Pasos |   Aproximacion Raiz   |         f(x)     |
    1        -0.6236881613731        -5.24319410
    2        -0.4930534660816        -0.49672189
    3        -0.4778514206409        -0.00619461
    4        -0.4776569902897        -0.00000076
    5        -0.4776569604874         0.00000019
---------------------------------------------------
Tolerancia :  0.00000099999999748
---------------------------------------------------
Raiz :   -0.47765696048736572265625000
---------------------------------------------------
```

Fig. 4: Primera solución numérica a $f(x) = 0$.

```
Valor inicial
5
---------------------------------------------------
| Pasos |   Aproximacion Raiz   |         f(x)     |
    1         4.8780484199524        -0.53339875
    2         4.8661127090454        -0.00477755
    3         4.8660039901733        -0.00000626
    4         4.8660039901733        -0.00000626
---------------------------------------------------
Tolerancia :  0.00000099999999748
---------------------------------------------------
Raiz :    4.86600399017333984375000000
---------------------------------------------------
```

Fig. 5: Segunda solución numérica a $f(x) = 0$.

Por lo tanto, nuestras 2 raíces son aproximadamente: $x_1 \simeq -0.477$ y $x_2 \simeq 4.866$.

3. $xe^{0.5x} + 1.2x - 5 = 0$

Procedemos a graficar la función auxiliar en GNUPLOT, cuyas raíces serán las mismas que las soluciones que buscamos.

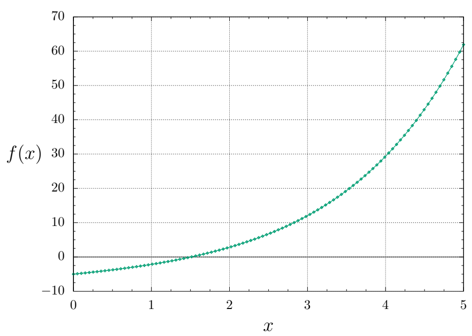


Fig. 6: Gráfica $f(x) = xe^{0.5x} + 1.2x - 5$.

La raíz se encuentra aproximadamente en $x \sim 1.5$, así que escogemos el valor $x_0 = 1$ para ejecutar nuestro programa.

```
Valor inicial
1
---------------------------------------------------
| Pasos |   Aproximacion Raiz   |         f(x)     |
    1         1.5856876373291         0.40667403
    2         1.5068992376328         0.00940857
    3         1.5049887895584         0.00000517
    4         1.5049877166748        -0.00000018
    5         1.5049877166748        -0.00000018
---------------------------------------------------
Tolerancia :  0.00000099999999748
---------------------------------------------------
Raiz :   1.50498771667480468750000000
---------------------------------------------------
```

Fig. 7: Solución numérica a $f(x) = 0$.

Por lo tanto, el valor de x que satisface nuestra ecuación es $x \simeq 1.5049$.

4. $\cos x \cosh x + 1 = 0$

Para esta ecuación, armamos una función auxiliar de la misma forma y la graficamos en GNUPLOT.

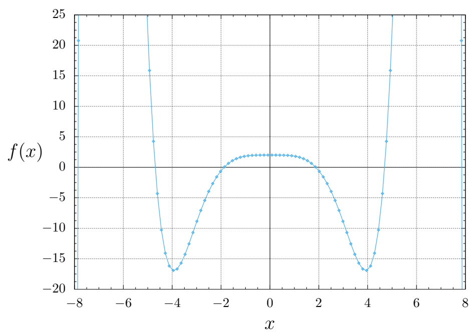


Fig. 8: Gráfica $f(x) = \cos x \cosh x + 1$.

En total, la función posee 6 raíces y por tanto, 6 valores x_1, x_2, x_3, x_4, x_5 y x_6 que satisfacen la ecuación de nuestro problema. Sin embargo, con un análisis de la paridad de la función, podemos observar que 3 soluciones son el negativo de las otras 3, respectivamente. Es decir: $x_1 = -x_4$, $x_2 = -x_5$ y $x_3 = -x_6$. Computando para valores aproximados de las raíces:

```
Valor inicial
2
---------------------------------------------------
| Pasos |   Aproximacion Raiz   |         f(x)     |
    1         1.8852746486664        -0.04240226
    2         1.8751792907715        -0.00031127
    3         1.8751040697098         0.00000005
    4         1.8751040697098         0.00000005
---------------------------------------------------
Tolerancia :  0.00000099999999748
---------------------------------------------------
Raiz :   1.87510406970977783203125000
---------------------------------------------------
```

Fig. 9: Primera solución numérica a $f(x) = 0$.

```
Valor inicial
4.3
-----------------------------------------------------------
| Pasos |     Aproximacion Raiz    |         f(x)        |
    1         5.0248427391052         24.38548088
    2         4.7702345848083          4.40969801
    3         4.6994395256042          0.28848183
    4         4.6941199302673          0.00154488
    5         4.6940913200378          0.00001009
    6         4.6940913200378          0.00001009
-----------------------------------------------------------
Tolerancia :  0.00000099999999748
-----------------------------------------------------------
Raiz :    4.69409132003784179687500000
-----------------------------------------------------------
```

Fig. 10: Segunda solución numérica a $f(x)=0$.

```
Valor inicial
7.9
-----------------------------------------------------------
| Pasos |     Aproximacion Raiz    |         f(x)        |
    1         7.8566861152649         -2.49276185
    2         7.8547611236572         -0.00475407
    3         7.8547573089600          0.00016682
    4         7.8547573089600          0.00016682
-----------------------------------------------------------
Tolerancia :  0.00000099999999748
-----------------------------------------------------------
Raiz :    7.85475730895996093750000000
-----------------------------------------------------------
```

Fig. 11: Tercera solución numérica a $f(x)=0$.

Y por tanto, los valores que satisfacen la ecuación son: $x_1=-x_4\simeq 1.875$, $x_2=-x_5\simeq 4.694$ y $x_3=-x_6\simeq 7.854$.

5. $x-2e^{-x}=0$

Realizamos una función auxiliar $f(x)=x-2e^{-x}$ y la graficamos en GNUPLOT.

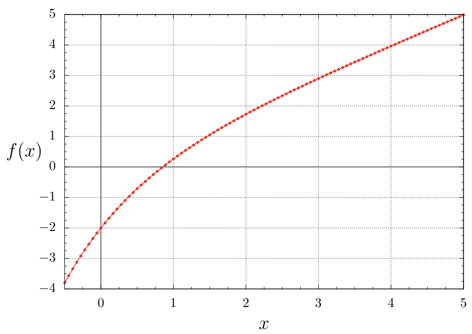


Fig. 12: Gráfica $f(x)=x-2e^{-x}$.

Procedemos a evaluar numéricamente la raíz con un valor inicial de $x_0=1$, y resulta:

```
Valor inicial
1
-----------------------------------------------------------
| Pasos |     Aproximacion Raiz    |         f(x)        |
    1         0.8477662205696         -0.00897527
    2         0.8526000976562         -0.00001001
    3         0.8526055216789          0.00000006
    4         0.8526054620743         -0.00000006
-----------------------------------------------------------
Tolerancia :  0.00000099999999748
-----------------------------------------------------------
Raiz :    0.85260546207427978515625000
-----------------------------------------------------------
```

Fig. 13: Solución numérica a $f(x)=0$.

Por lo tanto, la solución de la ecuación $x\simeq 0.852$.

b. Segundo problema

Un detector de temperatura mediante la resistencia de un material, principalmente Niquel, se basa en el cambio de resistencia eléctrica con los cambios de temperatura. La expresión de la resistencia a una temperatura dada, está dada por la siguiente expresión:

$$R_T=R_0(1+AT+BT^2+CT^4+DT^6) \tag{2}$$

Siendo R_0 la resistencia a 0°C y A, B, C y D constantes. Si $R_0=100\Omega$, determine la temperatura cuando $R_T=300\Omega$.
Datos: $A=5.485\times10^{-3}$, $B=6.65\times10^{-6}$, $C=2.8055\times10^{-11}$ y $D=-2.0\times10^{-17}$.

SOLUCIÓN

1. Gráfica

Para esto, creamos una función auxiliar de la temperatura T; $f(T)=R_T-300$. Al graficarla, las raíces que presente serán la solución a nuestro problema original.

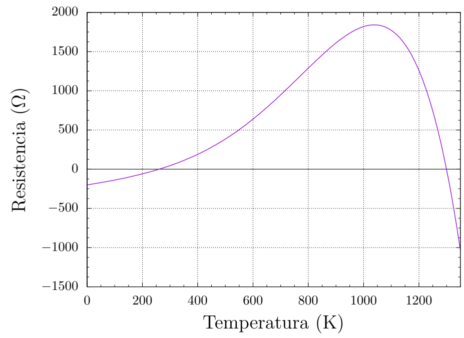


Fig. 14: Función auxiliar para encontrar la solución T.

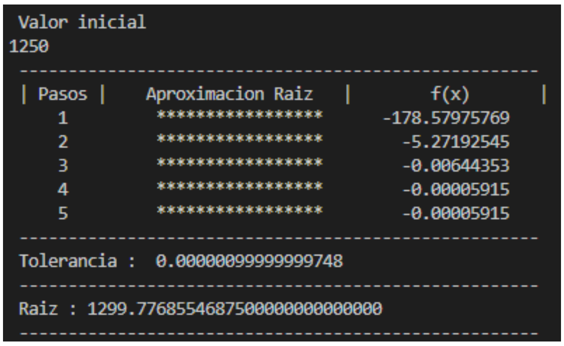
De esta forma, vemos hay 2 raíces que también son físicamente posibles, ya que la T_f del Niquel es de 1455°C. Encontramos que las raíces se encuentran en los intervalos [200,400] y [1200,1400].

2. Solución Númerica

Utilizando el método de Newton, utilizamos como valores iniciales aquellos que estén cercanos a cada una de las raíces, así obtenemos los resultados:

```
Valor inicial
200
-----------------------------------------------------------
| Pasos |     Aproximacion Raiz    |         f(x)        |
    1       265.9005432128906          6.18162918
    2       260.2663879394531          0.05372896
    3       260.2165527343750          0.00001184
    4       260.2165527343750          0.00001184
-----------------------------------------------------------
Tolerancia :  0.00000099999999748
-----------------------------------------------------------
Raiz :  260.21655273437500000000000
-----------------------------------------------------------
```

Fig. 15: Primera solución numérica a $f(T)=0$.



```
Valor inicial
1250
--------------------------------------------------------
| Pasos |     Aproximacion Raiz    |        f(x)        |
   1         *****************       -178.57975769
   2         *****************         -5.27192545
   3         *****************         -0.00644353
   4         *****************         -0.00005915
   5         *****************         -0.00005915
--------------------------------------------------------
Tolerancia :  0.00000099999999748
--------------------------------------------------------
Raiz : 1299.7768554687500000000000000
--------------------------------------------------------
```

Fig. 16: Segunda solución numérica a $f(T) = 0$.

Por lo tanto, las temperaturas necesarias para que $R_T = 300\Omega$ son: $T_1 \simeq 260.216$K y $T_2 \simeq 1299.776$K.

c. *Tercer problema*

Durante un juego de futbol americano, un lanzador, lanza la pelota, estando a una altura h_Q. El recibidor, está alejado 60 pies y a una altura h_R. Se conocen la velocidad inicial, la distancia vertical x y la distancia horizontal y.
Datos: $g = 32.2ft/s^2$, $v_0 = 50ft/s$, $x = 60ft$, $h_Q = 6.5ft$ y $h_R = 7ft$.
Determinar el ángulo θ que hace la pelota con la horizontal antes de dejar la mano del lanzador.

$$h_R = x\tan\theta - \frac{x^2 g}{2v_0^2\cos\theta^2} + h_Q \tag{3}$$

1. *Gráfica*

Similar a los ejercicios previos, armamos una función auxiliar de manera que las raíces coincidan con la solución de nuestro problema. Así:

$$f(\theta) = x\tan\theta - \frac{x^2 g}{2v_0^2\cos\theta^2} + h_Q - h_R \tag{4}$$

Y procedemos a graficarla.

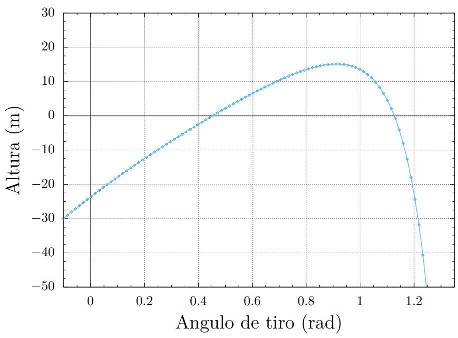
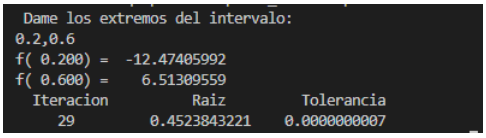


Fig. 17: Gráfica de $f(\theta)$.

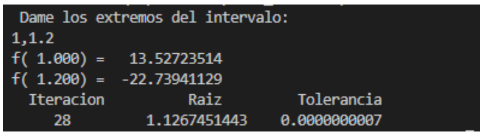
2. *Solución numérica*

De la gráfica podemos obtener que existen dos valores de θ que satisfacen la ecuación (3). En este caso utilizaremos el método de bisección con los intervalos $[0.2, 0.6]$ y $[1, 1.2]$ para la primera y segunda raíz, respectivamente.



```
Dame los extremos del intervalo:
0.2,0.6
f( 0.200) =  -12.47405992
f( 0.600) =    6.51309559
  Iteracion          Raiz          Tolerancia
     29          0.4523843221    0.0000000007
```

Fig. 18: Primera solución numérica a $f(\theta) = 0$.



```
Dame los extremos del intervalo:
1,1.2
f( 1.000) =   13.52723514
f( 1.200) =  -22.73941129
  Iteracion          Raiz          Tolerancia
     28          1.1267451443    0.0000000007
```

Fig. 19: Segunda solución numérica a $f(\theta) = 0$.

Y con una conversión de unidades podemos obtener que: $\theta_1 \simeq 25.919°$ y $\theta_2 \simeq 64.557°$.

d. *Cuarto problema*

La fuerza que actúa entre una partícula con carga q y un disco circular de radio R y una carga Q está dada por la ecuación:

$$F = \frac{Qqz}{2\epsilon_0}\left(1 - \frac{z}{\sqrt{z^2 + R^2}}\right) \tag{5}$$

Donde ϵ_0 es la constante de permitividad en el vacío y z es la distancia de la partícula.
Datos: $\epsilon_0 = 0.885 \times 10^{-12} C^2/(Nm^2)$, $Q = 9.4 \times 10^{-6} C$, $q = 2.4 \times 10^{-5} C$ y $R = 0.1m$.
Determinar la distancia z, si la fuerza es de 0.3 N.

Desde GNUPLOT graficamos la función auxiliar $f(z) = F - 0.3$ para obtener el intervalo de la raíz o raíces cuyo valor es solución al problema inicial.

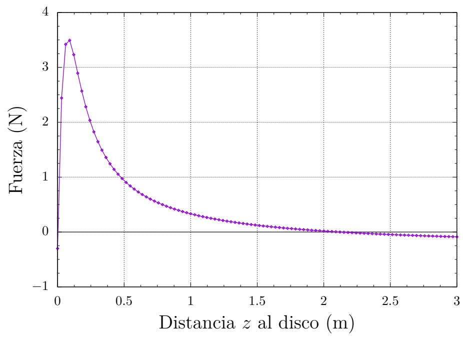
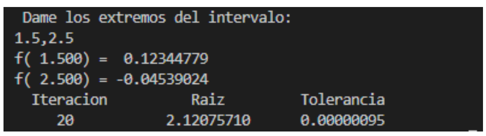


Fig. 20: Gráfica de $f(z)$.

Utilizaremos nuevamente el método de bisección, con ayuda de la gráfica elegimos un intervalo conveniente para ejecutar el programa, en este caso: $[1.5, 2.5]$.



```
Dame los extremos del intervalo:
1.5,2.5
f( 1.500) =  0.12344779
f( 2.500) = -0.04539024
  Iteracion         Raiz         Tolerancia
     20          2.12075710     0.00000095
```

Fig. 21: Solución numérica a $f(z) = 0$.

Por lo tanto, el valor de z al cual $F = 0.3$N es: $z \simeq 2.1207$m.

II. Algoritmo que determina si hay una raíz en un intervalo dado

a. Código en FORTRAN del programa

```fortran
program intervaloraiz

    ! Dado un intervalo [a,b], hallar cuantas raices tiene la funcion.

    real*4 :: a,b, liminf(10), limsup(10),raiz(10),f ,sign! Imprime los intervalos de hasta 10 raices y 10
    integer :: n, count, i, ceros
    real*4, dimension(:), allocatable :: x! Vector dinamico, tendra dimension n


    write(*,*) '-------------------------------------------'
    write(*,'(9x,a)')'PROBLEMA 6'
    write(*,*) '-------------------------------------------'
    write(*,*)'Inserte el intervalo [a,b]: '
    read*,a,b
    write(*,*)'Introduzca el valor de dx: '
    read*,dx
    write(*,*) '--------------------------------------------'


    n = int((b-a)/dx) !Numero de pasos
    count = 0 ! Contador de intervalos encontrados
    ceros = 0 !Raices exactas encontradas

    allocate(x(n)) ! Aloja en la memoria el vector con n componentes

    ! Llenar el array
    x(1) = a
    do 1 j = 2,n
        x(j) = a + (j-1.)*dx
    1 continue

    ! Verificamos el numero de raices
    do i=1,n-1
        sign = f(x(i))*f(x(i+1))
        if (sign.lt.0.) then
            count = count + 1
            liminf(count) = x(i)
            limsup(count) = x(i+1)
        else if (sign==0.0.and.f(x(i))==0.0) then
            ceros = ceros+1
            raiz(ceros) = x(i)
        endif
    enddo

    deallocate( x ) ! Se desaloja en la memoria

    print*,'RESULTADO:'
    if (count+ceros /= 0) then
        write(*,10)count+ceros,a,b
        do 2 k = 1,count
            write(*,11)liminf(k),limsup(k)
        2 continue
        do 3 k = 1,ceros
            write(*,13)raiz(k)
        3 continue
    else
        write(*,12)a,b
    end if

    10 format('En total, se encontraron',1x,i1,1x,'raiz/ces en el intervalo [',f8.3,',',f8.3,'].')
    11 format('Se encontro una raiz en [',f8.3,',',f8.3,'].')
    12 format('No se han encontrado raices en el intervalo [',f8.3,',',f8.3,'].')
    13 format('Se ha encontrado una raiz exacta en x =',f13.8)

end program

function f(x)
    real*4 :: f,x
    f = (x-3.98)*(x-3.12)*(x-1.5)*(x-1)*(x-7)
    return
end function
```

b. Breve explicación

1. Variables

Las variables que utilizaremos para el programa son: a y b para el intervalo que introduce el usuario además de un diferencial dx, arrays liminf y limsup para almacenar el subintervalo donde se encuentra una raíz, contadores tanto para los ceros de la función (ceros) como para los subintervalos (count). Por último, un array dinámico x que tendrá dimensión n, para salvar cada uno de los puntos a evaluar.

2. Input

Al dar un intervalo donde encontrar la raíz y un diferencial dx, el programa calculará un valor entero para n: $n = int((b-a)/dx)$. Para después llenar el array dinámico x previamente alojado en la memoria con dimensión n.

3. Verificación de raíces

La parte más importante del programa, realiza un ciclo pasando por todos los subintervalos verificando el signo de la multiplicación $f(x)*f(x+dx)$ determinar si existe una raíz en ese intervalo y se guarda. En caso de que un extremo del intervalo sea una raíz, se añade al array raiz. Después de este proceso se desaloja de la memoria

4. Imprimir resultados

Para imprimir los resultados primero tenemos que saber si hay un resultado satisfactorio o no, para esto utilizamos una condicional if que verifique que el programa encontró soluciones y las imprime con ciclos. De lo contrario se imprime "No se han encontrado raíces en el intervalo".

III. Conclusiones

El método de Newton-Raphson es uno de los más utilizados en la computación como una introducción a los métodos numéricos. La sencillez con la que opera puede resultar más conveniente en ciertas situaciones o muchas dificultades en otras. Lo que me podría llevar de conocimiento en esta tarea es un mayor entendimiento en FORTRAN y GNUPLOT; en particular, sobre los arrays dinámicos o alojables que maneja FORTRAN ya que resultaron de gran utilidad en la resolución del problema 6 y el manejo del comando format para dar una mejor presentación a los resultados, y en GNUPLOT sobre nuevos comandos que ayudan a mejorar la calidad de las gráficas presentadas, entre ellos las nuevas funciones que aprendí fueron: Configurar la terminal para que el archivo de salida pueda ser un .png, un archivo .svg o un código en LaTeX, poner titulos y labels más sofisticados, configurar los medium tics para ambos ejes y modificar los ejes x e y a mi gusto. Además, aprendí a usar LaTeXde una mejor manera en la realización de esta tarea. En conclusión, FORTRAN y GNUPLOT le ganan a Python.

IV. Scripts de las gráficas y códigos de los métodos numéricos utilizados

a. Script general de las gráficas en GNUPLOT

```
set terminal epslatex color
set output 'output.tex'
set zeroaxis lt -1 lc -1
unset key
set grid xtics
set grid ytics
set xlabel '$x$'
set ylabel '$f(x)$'
set mytics 4
set mxtics 4

a = 0
b = 6
f(x) = cos(x) #Funcion a graficar
plot [a:b] f(x) w lp pt 12 lw 2 lc 1 ps 0.6
```

b. Método de Newton en FORTRAN

```
program Newton
    implicit none
    integer :: i, imax
    real ::  x , xnuevo, toler ,f ,df ,dx
    parameter (imax = 100)
```

```
     parameter (toler = 1.e-6)
     print*, 'Valor inicial ' ; read*, x
     i = 0 ; dx = 1
         print*, '--------------------------------------------------------'
         print*, '| Pasos |     Aproximacion Raiz    |          f(x)         | '
     do while (i<imax.and.(abs(dx))>toler)
       i = i + 1                          !
       xnuevo = x - f(x)/df(x)
       dx = xnuevo-x
       x = xnuevo
       write(*,'(2x,i4,9x,f17.13,2x,f17.8)') i, xnuevo, f(xnuevo)
     end do
     if (i>imax-2) then
         write(*,*) 'No converge en 100 iteraciones.'
         else
         print*, '--------------------------------------------------------'
         write(*,'(a14,f20.17)') 'Tolerancia : ', toler
         print*, '--------------------------------------------------------'
         write(*,'(a8,f30.25)') 'Raiz : ', xnuevo
         print*, '--------------------------------------------------------'
     endif
end program
function f(z)
     real ::  z, f
     f = -0.8*z*z*z*z + 6.6*z*z*z - 16*z*z + 11.7*z + 10
end
function df(z)
     real :: df,z
     df = - 3.2*z*z*z + 19.8*z*z - 32*z + 11.7
end
```

c. Método de Bisección en FORTRAN

```
program problema4
     real*8 :: dl = 1.e-9, a,b,c,dx,f
     write(*,*) 'Dame los extremos del intervalo:'
     read*,a,c
     dx = c-a ! Tamano del intervalo
     istep = 0 ! Contador

     ! Se evalua la funcion en los extremos del intervalo
     write(*,20)a,f(a)
     write(6,20)c,f(c)
     20  format ('f(',f6.3,') = ',f13.8)

     do 100 while (abs(dx).gt.dl) ! Define cuando parar
         b = (a+c)/2. ! Se define b
         if ((f(a)*f(b)).lt.0.0) then
             c = b
             dx = c-a
         else
             a=b
             dx = c-a
         end if
         istep = istep +1
     100 end do

     write (*,666) 'Iteracion','Raiz','Tolerancia'
     write (6,999) istep,b,dx
     stop
     666 format(2x,a,10x,a,9x,a)
     999 format(3x,i4,5x,2f16.10,3x)
end

function f(theta)
     real*8 f , theta
     f = 60.*tan(theta)-(60.**2 * 32.2)/(2*50.**2 * cos(theta)**2)+6.5-7.
     return
end function
```