Nombres:

- Geraldine Gómez.
- Brayan González.
- Santiago Roa.

Taller 1

1. Evaluar el valor de un polinomio es una tarea que involucra para la maquina realizar un número de operaciones la cual debe ser mínimas. Para cada una de los siguientes polinomios, hallar P(x) en el valor indicado y el número de operaciones mínimo para hacerlo (sugerencia utilizar el algoritmo Horner).

Método de Horner:

Es un algoritmo que permite calcular el resultado de polinomios para un determinado valor de x de una forma monomial.

El algoritmo toma uno a uno los coeficientes del polinomio por el orden de las potencias y los suma con el valor calculado de y anterior multiplicado por el valor del x que se desea evaluar.

```
rm(list=ls())
Horner <-function(P,Xn)
{
    y <- 0
    i <- 1
    while (i < length(P) + 1)
    {
        y = Xn*y + P[i]
        cat("y= ",y,"\tXn=",Xn,"\tOperaciones Mínimas = ",i*2,"\n")
        i <- i + 1

    }


    cat("El valor del polinomio es: ",y)

 }
```

La condición de las operaciones mínimas del algoritmo está dada por la relación 2^n, ya que solo realiza una operación de suma y otra de multiplicación el número de veces que indica el mayor exponente del polinomio.

a. $2x^4 - 3x^2 + 3x - 4$

$x_0 = -2$

```
P <- c (2,0,-3,3,-4)
Horner(P ,-2)
```

Resultado:

```
y=  2   Xn= -2  Operaciones Minimas =  2
y=  -4  Xn= -2  Operaciones Minimas =  4
y=  5   Xn= -2  Operaciones Minimas =  6
y=  -7  Xn= -2  Operaciones Minimas =  8
y=  10  Xn= -2  Operaciones Minimas =  10
El valor del polinomio es:  10
```

b. $7x^5 + 6x^4 - 6x^3 + 3x - 4$

$x_0 = 3$

```
P2 <- c(7,6-6,0,3,-4)
Horner(P2,3)
```

Resultado:

```
y=  7           Xn= 3   Operaciones Minimas =  2
y=  21          Xn= 3   Operaciones Minimas =  4
y=  63          Xn= 3   Operaciones Minimas =  6
y=  192         Xn= 3   Operaciones Minimas =  8
y=  572         Xn= 3   Operaciones Minimas =  10
El valor del polinomio es:  572
```

c. $-5x^6 + 3x^4 + 2x^2 - 4x$

$x_0 = -1$

```
P3 <- c(-5,0,3,0,2,-4,0)
Horner(P3,-1)
```

Resultado:

```
y=  -5          Xn= -1  Operaciones Minimas =  2
y=  5           Xn= -1  Operaciones Minimas =  4
y=  -2          Xn= -1  Operaciones Minimas =  6
y=  2           Xn= -1  Operaciones Minimas =  8
y=  0           Xn= -1  Operaciones Minimas =  10
y=  -4          Xn= -1  Operaciones Minimas =  12
y=  4           Xn= -1  Operaciones Minimas =  14
El valor del polinomio es:  4
```

2. La eficiencia de un algoritmo esta denotada por T(n**)**

Leer n
Mientras n>0 repita
$d < -mod(n, 2)$
$n < -fix(n, 2)$
Mostrar d
Fin

Algoritmo descrito en R:

```
rm(list=ls())
eficiencia <- function(n)
{
 x <- n
 i <- 0
 while ( n > 0)
 {
  d <- n%%2
  n <- floor(n/2)
  i <- i+1
  cat("i= ",i*2,"\tn= ",n,"\t","d= ",d,"\n")
  }
}
```

a. El algoritmo anteriormente descrito representa de forma binaria un número, el cual se probará con n = 73, para corroborar su veracidad.

En la siguiente tabla se denotan los resultados de la ejecución del algoritmo los cuales son la cantidad de operaciones que está realizando el código representado con la variable i, el número n y el residuo d que entre otras cosas me permite determinar si el numero era par o impar:

Resultado:

```
i=  2   n=  36   d=  1
i=  4   n=  18   d=  0
i=  6   n=  9    d=  0
i=  8   n=  4    d=  1
i=  10  n=  2    d=  0
i=  12  n=  1    d=  0
i=  14  n=  0    d=  1
```

Como se puede observar las operaciones por ciclo van aumentando de dos en dos, lo que nos ayuda a determinar de manera deductiva la eficiencia del código.

b. Eficiencia del algoritmo:

$\boldsymbol{T(n) = 2 + k}$ (k: número de iteraciones necesarias).

$$\boldsymbol{n_1 = \frac{n}{2}}$$
$$\boldsymbol{n_2 = \frac{n}{4}}$$
$$\boldsymbol{n_3 = \frac{n}{8}}$$
$$\boldsymbol{.}$$
$$\boldsymbol{.}$$
$$\boldsymbol{.}$$
$$\boldsymbol{n_k = \frac{n}{2^k}}$$

$\boldsymbol{\frac{n}{2^k} < 2}$ (Tamaño de n después de k iteraciones)

$\mathbf{n < 2^{k+1}}$ (Formula de acotamiento)

$$\mathbf{\ln(n) < \ln(2^{k+1})}$$
$$\mathbf{\ln(n) < (k+1) * \ln(2)}$$
$$\boldsymbol{\frac{\ln(n)}{\ln(2)} < k+1}$$
$$\boldsymbol{\frac{\ln(n)}{\ln(2)} - 1 < k}$$

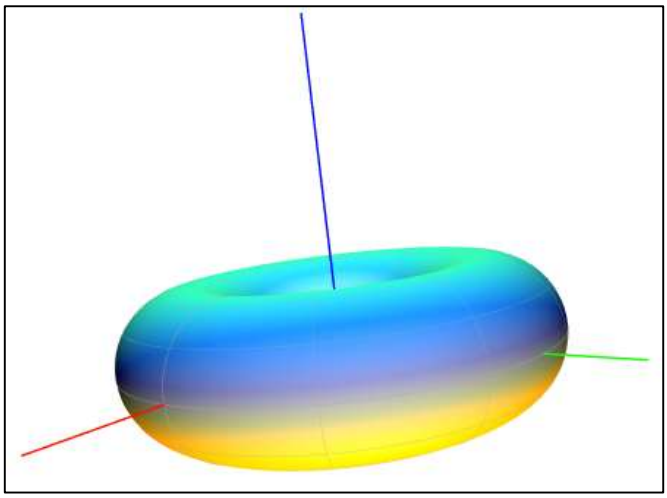
$\boldsymbol{O = O\ (log_2(n))}$

Como es posible evidenciar, se pudo obtener la eficiencia del algoritmo a través de la aplicación de proceso deductivo, llegando a la conclusión de que la complejidad es de $\log_2(n)$. Debido a que en cada iteración se está partiendo n a la mitad, reduciendo así su tamaño y el rango de evaluación de las diferentes operaciones. A pesar de que su complejidad es una de las más optimas, caso contrario es su Convergencia debido a que su pendiente es ½, convirtiéndola en lineal.

3. Una partícula se mueve en el espacio con el vector posición **R(t) = (2cos(t), sen(t),0).** Se requiere conocer el tiempo en el que el objeto se encuentra más cerca del punto **P (2,1,0)**. Utilice el método de Newton con cuatro decimales de precisión.

Descripción del Procedimiento:

En primera instancia se realiza la identificación de la superficie parametrizada, correspondiendo a un elipsoide, como es posible evidenciarlo en la siguiente gráfica:



Posteriormente se procede a definir la función que me permitirá encontrar la distancia entre el punto P y el vector V, y ponerla en términos que cumplan con el método de newton, para lo cual es necesario realizar el siguiente proceso:

R(t) = (2cos(t), sen(t),0), P (2,1,0)

f(t) = d'
f´(t) = d''

d = $\sqrt{(2\cos(t)-2)^2+(\sin(t)-1)^2(t)+(0-0)^2}$

d'= $\frac{8\sin(t)-3\sin(2t)-2\cos(t)}{2\sqrt{4\,cos^2(t)+sin^2(t)-8\cos(t)+2\sin(t)+5}}$

d' = 3 sin(t)cos(t) – 4sin(t) + cos(t) = 0

d''= - sin(t) - 4cos(t) - 3cos(2t)

Por último, se procede a tomar un valor inicial que se encuentre entre el dominio de la función y el punto P, siento este 0, procediendo a calcular la distancia a través del siguiente algoritmo:

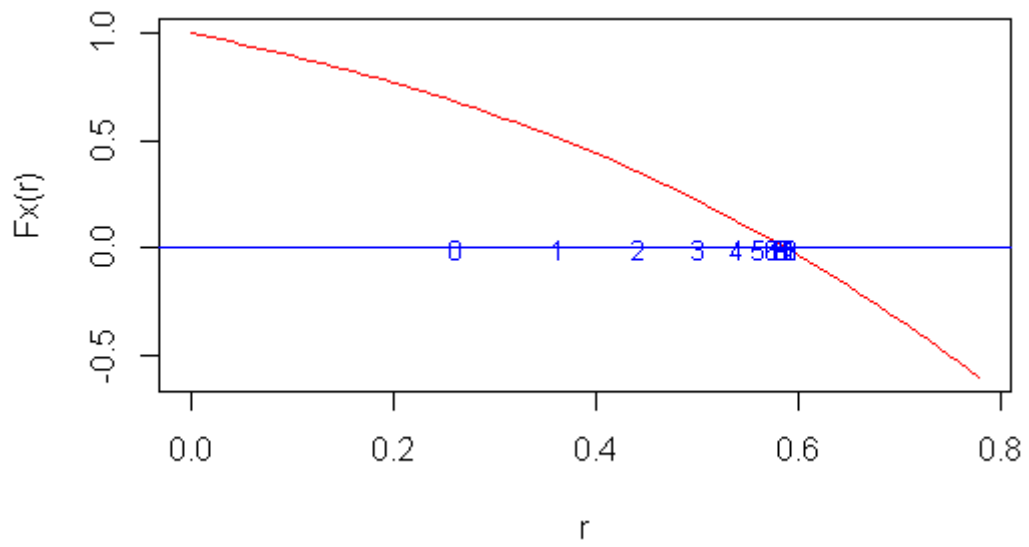
Algoritmo:

```
rm(list=ls())
Fx <- function(t) 3*sin(t)*cos(t)-4*sin(t)+cos(t)
Dx <- function(t) -sin(t) -4*cos(t) -3*cos(2*t)
Hx <- function(t) t - (Fx(t)/Dx(t))

 newton <- function(a,b)
 {


  if((Fx(a)-a)*(Fx(b)-b)<0)
  {
   r<-seq(a,b,0.01)
   plot(r,Fx(r),type="l",col="red")
   abline(h=0,col="blue")


   t<-0
   r<-Hx(t)
   i<-0
   while (Fx(r) != 0 )
   {

    error<-abs(Fx(t)/Dx(t))

    if(error >  1.e-4)
     t<-r
    else break
    r<-Hx(t)

    text(r,0,i,cex=0.8,col="blue")

    i<-i+1

    cat("I=",i,"\tF(t) =",Fx(t),"\tT=",signif(t, digits = 4),"\tE=",error,"\n")
   }

  }
  else
  {
   cat("No tiene raíz la función en ese intervalo.\n")
  }

 }

 newton(0,pi/4)
```

Resultado:

```
I= 1     F(t) = 0.8430906          T= 0.1429        E= 0.1428571
I= 2     F(t) = 0.6778332          T= 0.2636        E= 0.1207865
I= 3     F(t) = 0.507396           T= 0.3646        E= 0.1009444
I= 4     F(t) = 0.3469569          T= 0.4447        E= 0.08014699
I= 5     F(t) = 0.2143954          T= 0.5032        E= 0.05850295
I= 6     F(t) = 0.1200839          T= 0.5416        E= 0.03834727
I= 7     F(t) = 0.06207925         T= 0.564         E= 0.02245124
I= 8     F(t) = 0.03039057         T= 0.576         E= 0.01193781
I= 9     F(t) = 0.0144131          T= 0.5819        E= 0.005936326
I= 10    F(t) = 0.006723327        T= 0.5847        E= 0.002837908
I= 11    F(t) = 0.003110912        T= 0.5861        E= 0.001328921
I= 12    F(t) = 0.00143391         T= 0.5867        E= 0.0006160144
I= 13    F(t) = 0.0006597475       T= 0.587         E= 0.0002841785
I= 14    F(t) = 0.000303301        T= 0.5871        E= 0.0001308026
```

Gráfica:



4. Resolver por dos métodos diferentes, grafique las soluciones y comparar las soluciones.

Encuentre una intersección de las siguientes ecuaciones en coordenadas polares.

r = 2 + cos(3t)
r = 2 – e^t

Descripción del procedimiento:

Se establece la función de relación a través de:

$$\mathbf{f(x)} = 2 + \cos(3t) - (2 - e^{t})$$
$$= \cos(3t) + e^{t}$$

Teniendo en cuenta esta relación anterior de intersección de las curvas, se grafica con una herramienta tecnológica, para hallar el intervalo donde se encuentra el punto de intersección y con esto la convergencia, se adecuándose la función al método de secante y posición falsa.

a. **Posición Falsa – Intersección:**

Algoritmo

```
#Graficar en Polares

rm(list=ls())
polar <- function (theta, r, color=4)
{
 y <- 0
 x <- 0
 ejex <- 1

 for (i in 1:length(r))
 {
   if(is.nan(r[i])== T)
   {
     r[i] <- 0
   }
 }

 angulo <- seq(-max(theta),max(theta),by=theta[2]-theta[1])
 y <- r*sin(theta)
 x <- r*cos(theta)
 plot.new()
 plot.window(xlim = c(-max(r), max(r)), ylim = c(-max(r), max(r)), asp
= 1)

 aux <- max(r)
 # Dibuja los ejes.
 while (aux > 0)
 {
   fi <- aux*sin(angulo)
```

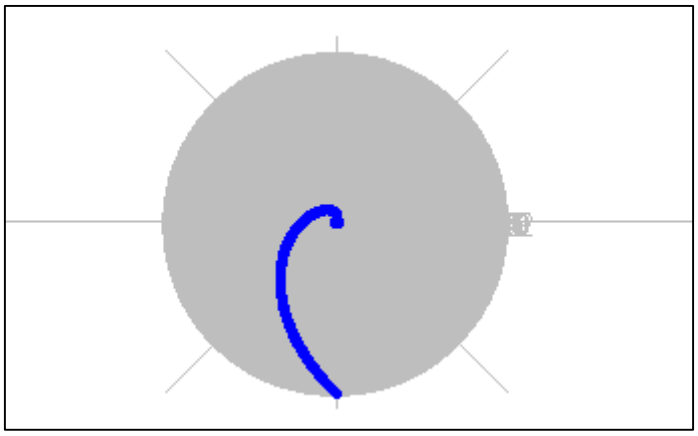
```
  cir <- aux*cos(angulo)
  points(cir,fi,pch="-",col="gray",cex=0.3)
  text(ejex+0.2,-0.2,ejex,col="gray")
  ejex <- ejex + 1
  aux <- aux - 1
  }

 abline(v=((max(cir)+min(cir))/2),col="gray")
 abline(h=((max(cir)+min(cir))/2),col="gray")
 segments(-max(r)+0.5,-max(r)+0.5,max(r)-0.5,max(r)-0.5,col="gray")
 segments(-max(r)+0.5,max(r)-0.5,max(r)-0.5,-max(r)+0.5,col="gray")

 points(x,y,pch=20,col=color,cex=1)
}

dim <- seq(-pi, 3*pi/2, by=pi/600)
r <- cos(3*dim) - exp(dim)
polar(dim,r,"blue")


#  Posicion falsa - Intersección

  rm(list=ls())
  Fr<-function(x) 2+cos(3*x)
  Gr<-function(x) 2-exp(x)
  regulaFalsi <- function(a,b)
 {
  x<-((Fr(b)*a)-(Fr(a)*b))/(Fr(b)-Fr(a))
  error<-abs(Fr(x)/Gr(x))
  while(error>1.e-4)
 {
      x<-((Fr(b)*a)-(Fr(a)*b))/(Fr(b)-Fr(a))
      if(Fr(x)==0)
      {
        break
      }
      if(Fr(x)*Fr(a)<0)
      {
        b<-x
      }
      else
      {
        a<-x
      }
      error<-abs(Fr(x)/Gr(x))
      cat("r=",x,"    \t","E=",error,"\n")
   }
```

```
}
regulaFalsi(pi,3*pi/2)
```

Resultado:

```
r= 1.570796                E= 0.7116229
r= 2.021211e+15            E= 0
```

Gráfica:



b. Secante – Intersección:

```
#Graficar en Polares

rm(list=ls())
polar <- function (theta, r, color=4)
{
  y <- 0
  x <- 0
  ejex <- 1

  for (i in 1:length(r))
 {
    if(is.nan(r[i])== T)
    {
      r[i] <- 0
    }
  }

  angulo <- seq(-max(theta),max(theta),by=theta[2]-theta[1])
  y <- r*sin(theta)
```

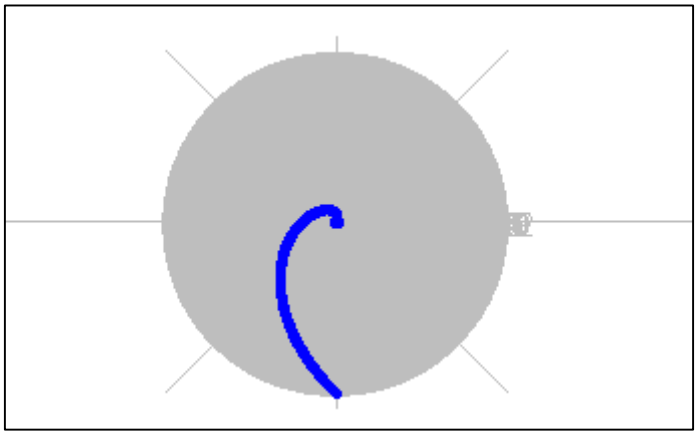
```
  x <- r*cos(theta)
  plot.new()
  plot.window(xlim = c(-max(r), max(r)), ylim = c(-max(r), max(r)), asp
= 1)

  aux <- max(r)
  # Dibuja los ejes.
  while (aux > 0)
  {
    fi <- aux*sin(angulo)
    cir <- aux*cos(angulo)
    points(cir,fi,pch="-",col="gray",cex=0.3)
    text(ejex+0.2,-0.2,ejex,col="gray")
    ejex <- ejex + 1
    aux <- aux - 1
    }

  abline(v=((max(cir)+min(cir))/2),col="gray")
  abline(h=((max(cir)+min(cir))/2),col="gray")
  segments(-max(r)+0.5,-max(r)+0.5,max(r)-0.5,max(r)-0.5,col="gray")
  segments(-max(r)+0.5,max(r)-0.5,max(r)-0.5,-max(r)+0.5,col="gray")

  points(x,y,pch=20,col=color,cex=1)
}

dim <- seq(-pi, 3*pi/2, by=pi/600)
r <- cos(3*dim) - exp(dim)
polar(dim,r,"blue")

 rm(list=ls())
 Fx <- function(x) 2+cos(3*x)
 Gx <- function(x) 2-exp(x)


 #Secante – Intersección

 secante <- function(a,b)
 {
  x<-(Fx(b)*a-Fx(a)*b)/(Fx(b)-Fx(a))
  error <-1
  while (error > 1.e-4)
  {
    a<-b
    b<-x
    x<-(Fx(b)*a-Fx(a)*b)/(Fx(b)-Fx(a))

    if (Fx(x) == 0) break
```

```
    error<-abs(Fx(x)/Gx(x))
    cat("r=",x,"\t","E=",error,"\n")
  }
}
secante(pi,3*pi/2)
```

Resultado:

```
r= 2.021211e+15          E= 0
```

Gráfica:



5.

13. Encuentre una formula iterativa de convergencia cuadrática y defina un intervalo de convergencia apropiado para calcular la raíz real n-esima de un número real.

Descripción del procedimiento:

Se procede a tomar la siguiente relación matemática $\sqrt[n]{x} = r$, para realizar la formula iterativa.

a. **Formula iterativa:**

f(x) = $x - \frac{x^n - r}{nx^{(n-1)}}$

$\mathbf{x_0} = \frac{v}{2}$

$\mathbf{x_1}$ = f(x_0)

$\mathbf{x_2}$ = f(x_1)

$\mathbf{x_3}$ = f(x_2)

$\mathbf{x_{k+1}}$ = f (xk), **k**: creciente

n: Raíz enésima
r: Número al cual se le va a calcular la raíz enésima

Demostración:

$\sqrt[n]{x^n} = r$
$(\sqrt[n]{r})^n = (x)^n$
$x^n - r = 0$

g(x) = $x^n - r$
g'(x) = $n\, e^{n-1}$
tg(e) = $e - \frac{e^n - r}{n\, e^{n-1}}$

Teorema de Convergencia cuadrática:

$$\begin{aligned} \boldsymbol{e_{n+1}} &= |x_n - \sqrt[n]{r}| \\ &= |(x_n^n - r) - (s^n - r)| \\ &= |(n\, s^{n-1})(x_n - s) + \tfrac{1}{n}(n(n-1))s^{n-2}(x_n - s)^2| \\ &= 0 + \tfrac{1}{n}(n(n-1))s^{n-2}e^2 n \\ &\lim_{n\to\infty} \tfrac{1}{n}|n^2 - n\,(s^{n-2})| \neq \infty \end{aligned}$$

Intervalo de Convergencia:

$r = 2^n r_1$

$\frac{1}{n} < \sqrt[n]{r} < 1$

b. Algoritmo:

A partir de la demostración anterior y debido a la convergencia cuadrática se opta por tomar el método de Newton Raphson y adecuarlo para que cumpla la convergencia deseada y permita hallar la raíz enésima de cualquier número real, indicando la condición de unicidad de la raíz y que debe existir en el intervalo de convergencia.

```
Fx <- function(x,n,v) x-(x^n-v)/(n*x^(n-1))
Gx <- function(x,n,v) x^n-v
Hx <- function(x,n) n*x^(n-1)
```

```
calcularRaiz <- function(v,n)
{

  x <- v/2
  error <- 1
  i <- 0
  while(error > 1.e-8)
  {

    x <- Fx(x,n,v)

    error <- abs(Gx(x,n,v))/Hx(x,n)
    cat(" x= ",x," \terror= ",error,"\n")
    i <- i + 1

  }


}
```

- Se toma como valor de prueba $\sqrt[3]{120}$

  calcularRaiz (120,3)

  Resultado:

```
x=  40.01111   error=  13.31205
x=  26.69906   error=  8.843573
x=  17.85549   error=  5.826366
x=  12.02912   error=  3.733273
x=  8.295849   error=  2.184066
x=  6.111783   error=  0.966422
x=  5.145361   error=  0.204246
x=  4.941115   error=  0.008675378
x=  4.932439   error=  1.527651e-05
x=  4.932424   error=  4.731362e-11
```

  Calculo con calculadora: 4,93242414

14. El siguiente es un procedimiento intuitivo para calcular una raíz real positiva de la ecuación **f(x) = 0** en un intervalo **[a, b]** con precisión **E.**

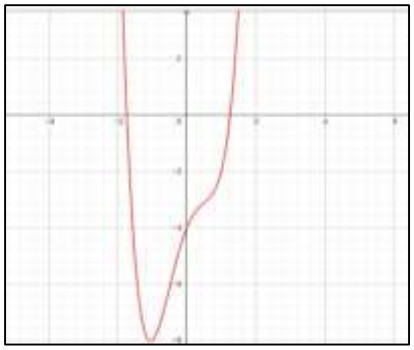
A partir de x = a evalúe **f(x)** incrementando x en un valor d: Inicialmente **d = (b-a) /10.** Cuando **f** cambie de signo, retroceda x al punto anterior **x-d**, reduzca **d** al valor **d/10** y evalué nuevamente **f** hasta que cambie de signo. Repita este procedimiento hasta que **d** sea menor que **E.**

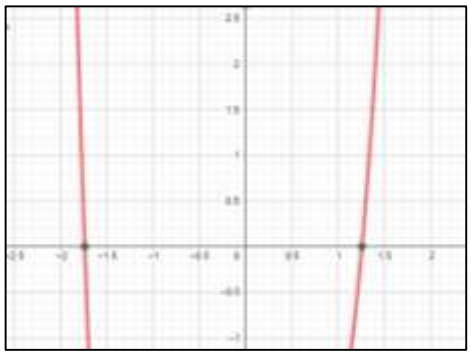
a. Las condiciones para que la raíz exista, sea única y pueda ser calculada son:

1. La función debe ser continua en el intervalo [a, b].
2. El valor de x debe existir en el intervalo [a, b].
3. Debe existir un cambio de signo cuando se evalúa el valor a y b en la función.
4. Debe tener una pendiente m < 1 para que exista una única raíz.

Al momento de realizar el codigo se tuvo en cuenta la idea de hacerlo de una manera recursiva, pero ya que se estan usando intervalos con valores muy pequeños esta forma no seria muy efectiva ya que estaria realizando muchas operaciones.

Se uso el procedimiento que se da en el texto, que es ir dandole un valor a d, y buscar la condición de que la función cambie de signo, si cambia de signo se realiza otra vez un cambio de d y se sigue repitiendo el proceso hasta que encuentre el resultado.





Los valores que se van a mostrar son la aproximación que tiene x al momento de que el intervalo va disminuyendo y también va a mostrar el error que va a tener ese valor x en el intervalo actualizado, de manera que se muestre el valor máximo que se alcanza antes de que se cumpla con la condición del error.

b. El orden de convergencia es lineal.

c. **Algoritmo:**

```
rm(list=ls())
Fx <- function(x) 2*x^4-3*x^2+3*x-4
RaizReal <- function(a, b)
{
 Hx <- Fx(a)-a
 if(Hx>0){
  Hx<- Fx(b)-b
  if(Hx<0){
   x<-a
   d<-(b-a)/10
   while (d > 1.e-8)
   {
    x<-(x+d)
    if (Fx(x)*Fx(x-d) < 0)
    {
     x<-(x-d)
     d<-d/10
    }
    cat("x= ",x,"   \tE= ",d,"\n")
   }
  }
  else
  {
   cat("La función no tiene raiz real positiva.")
  }
 }
}
RaizReal(-2, -1)
```

Resultado:

```
-1.9          E=   0.1
-1.8          E=   0.1
-1.8          E=   0.01
-1.79         E=   0.01
-1.78         E=   0.01
-1.77         E=   0.01
-1.76         E=   0.01
-1.75         E=   0.01
-1.74         E=   0.01
-1.74         E=   0.001
-1.739        E=   0.001
-1.739        E=   1e-04
-1.739        E=   1e-05
-1.73899              E=   1e-05
-1.73898              E=   1e-05
-1.73897              E=   1e-05
-1.73896              E=   1e-05
-1.73896              E=   1e-06
-1.738959             E=   1e-06
-1.738958             E=   1e-06
-1.738957             E=   1e-06
-1.738957             E=   1e-07
-1.738957             E=   1e-07
-1.738957             E=   1e-07
-1.738957             E=   1e-07
-1.738957             E=   1e-07
-1.738956             E=   1e-07
-1.738956             E=   1e-07
-1.738956             E=   1e-07
-1.738956             E=   1e-08
-1.738956             E=   1e-08
-1.738956             E=   1e-08
-1.738956             E=   1e-08
-1.738956             E=   1e-08
-1.738956             E=   1e-09
```

15. Se pide encontrar el valor de límite superior de la integral por método de punto fijo:

$$\int_0^x (5 - e^u)du = 2$$

Primero realizamos la evaluación de la integral para reducir los términos de la función y con esto obtener una ecuación que al evaluarla en un punto x permita obtener la raíz que será equivalente al límite superior que necesitamos hallar, la reducción está definida de la siguiente manera:

$$(5u - e^u)|_0^x = 2$$
$$5x - e^x + 1 = 2$$
$$f(x) = 5x - e^x - 1$$

Posteriormente se despeja x para obtener la función g(x) necesaria para el método de punto fijo:

$$f(x) = 0$$
$$5x - e^x - 1 = 0$$
$$5x = e^x + 1$$
$$x = \frac{e^x + 1}{5}$$
$$g(x) = \frac{e^x + 1}{5}$$

Después de haber obtenido las dos funciones, se debe buscar el intervalo de convergencia de la función f(x) original, en este caso la integral definida, para esto se toma como guía la gráfica de la función, cabe recordar que vamos a utilizar un graficador para tener mayor precisión:

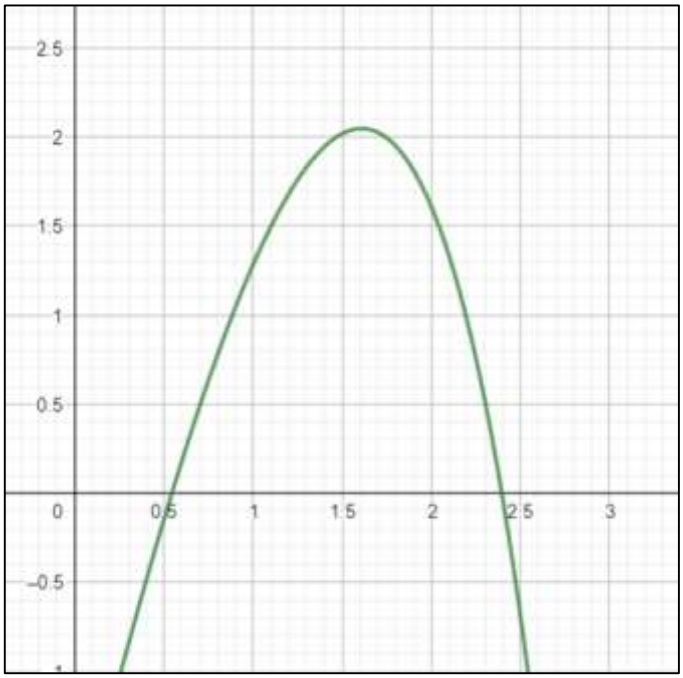


Gráfico función $f(x)$

Con esta grafica es posible denotar que las raíces de la función se encuentran entre (0, 1) y (2, 3); tomaremos el intervalo (0.5, 0.8) para aplicar el método. A continuación se representa el algoritmo que tiene una tolerancia de 1.10^{-8}.

a. Algoritmo:

```
rm(list=ls())
Fx <- function(x) 5*x-exp(x)-1
Gx <- function(x) (exp(x)+1)/5

puntoFijo <- function(a,b)
{
 x<-(a+b)/2
 i<-0
 while (Gx(x) != x )
 {
  error<-abs(a-b)/2
  if(error > 1.e-8)
    if (Gx(x) < x) b <- x
       else
       {
         a <- x
       }
  else
  {
    break
  }
   if(i>5)
   {
    break
   }
     x<-(a+b)/2 – Condición de convergencia del punto fijo

     i<-i+1
     cat("I=",i,"\tG(x) =",Gx(x),"      \tX=",x,"\tE=",error,"\n")

 }

}

puntoFijo(0.5,0.8) – Intervalo de convergencia.
```

Resultado:

I= 1	G(x) = 0.5554261	X= 0.575	E= 0.15
I= 2	G(x) = 0.5423444	X= 0.5375	E= 0.075
I= 3	G(x) = 0.548824	X= 0.55625	E= 0.0375
I= 4	G(x) = 0.545569	X= 0.546875	E= 0.01875
I= 5	G(x) = 0.5439529	X= 0.5421875	E= 0.009375
I= 6	G(x) = 0.54476	X= 0.5445313	E= 0.0046875

Punto Adicional

1. Tomar el método de aceleración de Aitken e implementarlo a cualquier problema del taller.

   Descripción del Procedimiento:

   Se sabe que el método se basa en el hecho de que la sucesión (x_n) esta defina por

   :

   $$x_{n+3} = x_n - \frac{(x_{n+1} - x_n)^2}{x_{n+2} - 2x_{n+1} + x_n}$$

   bajo ciertas condiciones, converger más rápidamente a x que la sucesión (x_n).

   Es por esto que se toma el punto No.13, que indica la creación de una formula iterativa de convergencia cuadrática para calcular una raíz enésima, la formula no tiene ningún tipo de cambio, lo que se modifica es la forma en que se calcula el x que se evalúa en dicha fórmula, aplicando el método de aceleración de la siguiente forma:

   $$x_0 = \frac{v}{2} \rightarrow valor\ inicial$$

   $$x_1 = f(x_0) \rightarrow formula\ iterativa$$

   $$x_2 = f(x_2)$$

   $$x_3 = x_0 - \frac{(x_1 - x_0)^2}{x_2 - 2x_1 + x_0} \rightarrow Formula\ de\ Aitken$$

$$x_r = f(x_3) \rightarrow Este\ es\ mi\ x\ mejor\ aproximado$$

$$f(x_r) \rightarrow Formula\ iterativa\ evaluada\ con\ el\ mejor\ x\ aproximado$$

Posteriormente si se cumple la condición de la formula iterativa y el valor x_r está en el intervalo de convergencia, se obtiene la valor de la raíz enésima.

Algoritmo:

```
Fx <- function(x,n,v) x-(x^n-v)/(n*x^(n-1))
Gx <- function(x,n,v) x^n-v
Hx <- function(x,n) n*x^(n-1)
Xn <- function(x,x1,x2) x - (((x1-x2)^2)/(x2-2*x1+x))

calcularRaiz <- function(v,n)
{

  x <- v/2
  error <- 1
  i <- 0
  while(error > 1.e-8)
  {
   x1<-(Fx(x,n,v))
   x2<-(Fx(x1,n,v))
   xr<-(Xn(x,x1,x2))

   if(Fx(xr,n,v)!=0)
   {
     x<-x1
   }
   else
   {
     break
   }

   error <- abs(Gx(xr,n,v))/Hx(xr,n)
   cat(" i=", i,"\t x= ",xr,"\t  error= ",error,"\n")
   i <- i + 1

  }
```

```
}

calcularRaiz(120,3)
```

Resultado:

```
i= 0    x=  33.45889      error=  11.11723
i= 1    x=  22.50878      error=  7.423975
i= 2    x=  15.44808      error=  4.981746
i= 3    x=  11.19677      error=  3.413193
i= 4    x=  8.950034      error=  2.483988
i= 5    x=  7.528817      error=  1.803928
i= 6    x=  6.057049      error=  0.9287372
i= 7    x=  5.144976      error=  0.2038917
i= 8    x=  4.941115      error=  0.008675351
i= 9    x=  4.932439      error=  1.527651e-05
i= 10   x=  4.932424      error=  4.731362e-11
```