PONTIFICIA UNIVERSIDAD JAVERIANA

ANALISIS NUMÉRICO
TALLER 1

JOHAN DANIEL ORTEGÓN PARRA
RICARDO RISCANEVO

BOGOTÁ D.C
2019

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 uno 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 en C

```
double horner(double p[],int n, double x){
  double y = p[0];
  int i;
  for(i = 1; i<n; i++){
    y = x*y + p[i];
  }
  return y;
}

double eval(double p[],int n, double x){
  double s = 0;
  int i;
  for(i = 0; i<n; i++){
    s = s + p[i]*pow(x,n-i-1);
  }
  return s;
}
```

$P(x) = 2x^4 - 3x^2 + 3x - 4$ en $x_0 = -2$

$P(x) = 7x^5 + 6x^4 - 6x^3 + 3x - 4$ en $x_0 = 3$

$P(x) = -5x^6 + 3x^4 + 2x^2 - 4x$ en $x_0 = -1$

Código Fuente (python 3)

```
from sympy import *

x = symbols('x')
P1 = input("Por favor ingrese el polinomio a evaluar (use como variable
x): ")
Polinomio = Poly(P1)
reemplazo_variable = int(input("Por favor ingrese el valor por el que
reemplazará la variable: "))
coeficientes = Polinomio.all_coeffs()
resultado = coeficientes[0]
iter = 0
cant_operaciones = 0
while iter < len(coeficientes)-1:
    resultado = reemplazo_variable*resultado + coeficientes[iter+1]
    cant_operaciones = cant_operaciones+2
    iter = iter+1

print("El resultado es: ", resultado, "la cantidad de operaciones fueron:
", cant_operaciones)
```

Evaluación de los polinomios

$2x^4 - 3x^2 + 3x - 4$ en $x_0 = -2$

Resultado: 10

Número de operaciones: 8

$7x^5 + 6x^4 - 6x^3 + 3x - 4$ en $x_0 = 3$

Resultado: 2030

Número de operaciones: 10

$-5x^6 + 3x^4 + 2x^2 - 4x$ en $x_0 = -1$

Resultado: 4

Número de operaciones: 12

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

6. Dado el siguiente algoritmo

```
Leer n
Mientras n>0 repita
    d ← mod(n,2)            Produce el residuo entero de la división n/2
    n ← fix(n/2)            Asigna el cociente entero de la división n/2
    Mostrar d
fin
```

a) Recorra el algoritmo con **n = 73**

b) Suponga que **T(n)** representa la cantidad de operaciones aritméticas de división que se realizan para resolver el problema de tamaño **n**. Encuentre **T(n)** y exprésela con la notación **O( )** Para obtener **T(n)** observe el hecho de que en cada ciclo el valor de **n** se reduce aproximadamente a la mitad.

Código fuente punto 2 (python 3)

```python
n = int(input("Ingrese el numero n: "))
while n > 0:
    d = n%2
    n = (n-d)/2
    print('valor de d: ', d)
```

Calculo de T(n)

Dado que T(n) depende del numero de divisiones y al mismo tiempo el número de divisiones dependen directamente del número “n” $T(n) = \log_2 n$

Calculo de complejidad en términos o( )

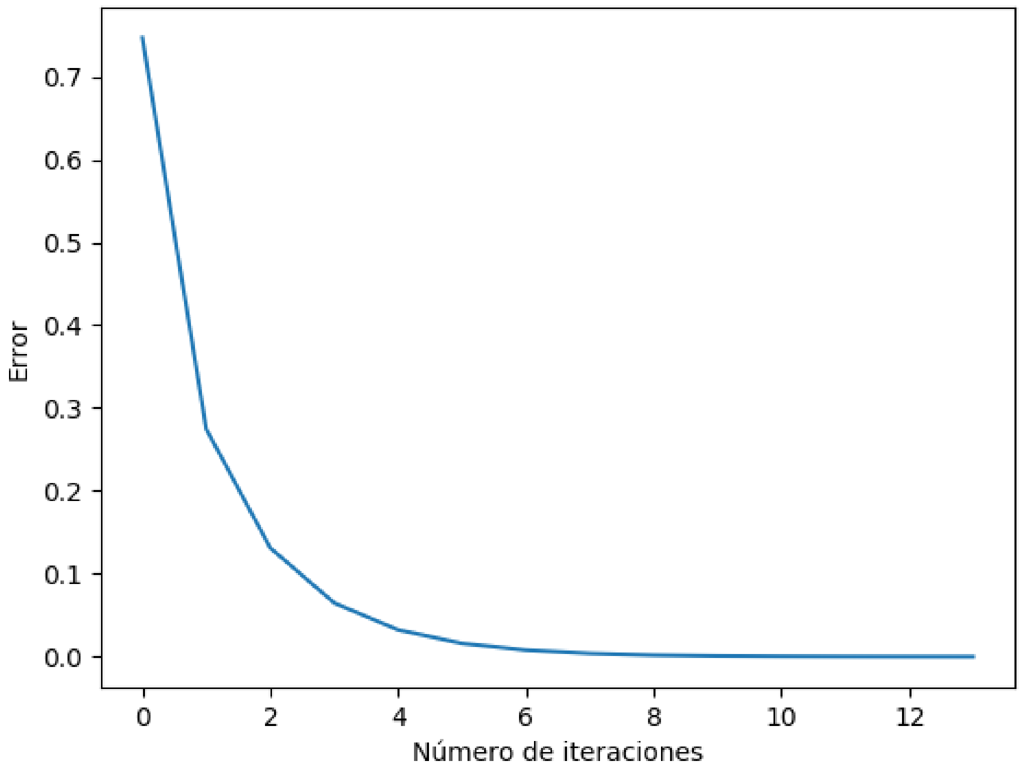
Al observar el ciclo (while) del algoritmo podemos notar que la variable de la cual depende su finalización está avanzando de la forma $n^{1/2}$ lo cual expresa un decrecimiento $2^x, 2^{x-1}, ., ., ., 2^3, 2^2, 2^1, 2^0, 2^{-1}$ y traduciendo al condicional del ciclo (while) quedaría $n^{1/2x} > i$

Despejando x: $x > \frac{2\,ln(1)}{ln(2)}$ Por lo cual la complejidad es de $o(log\, n)$

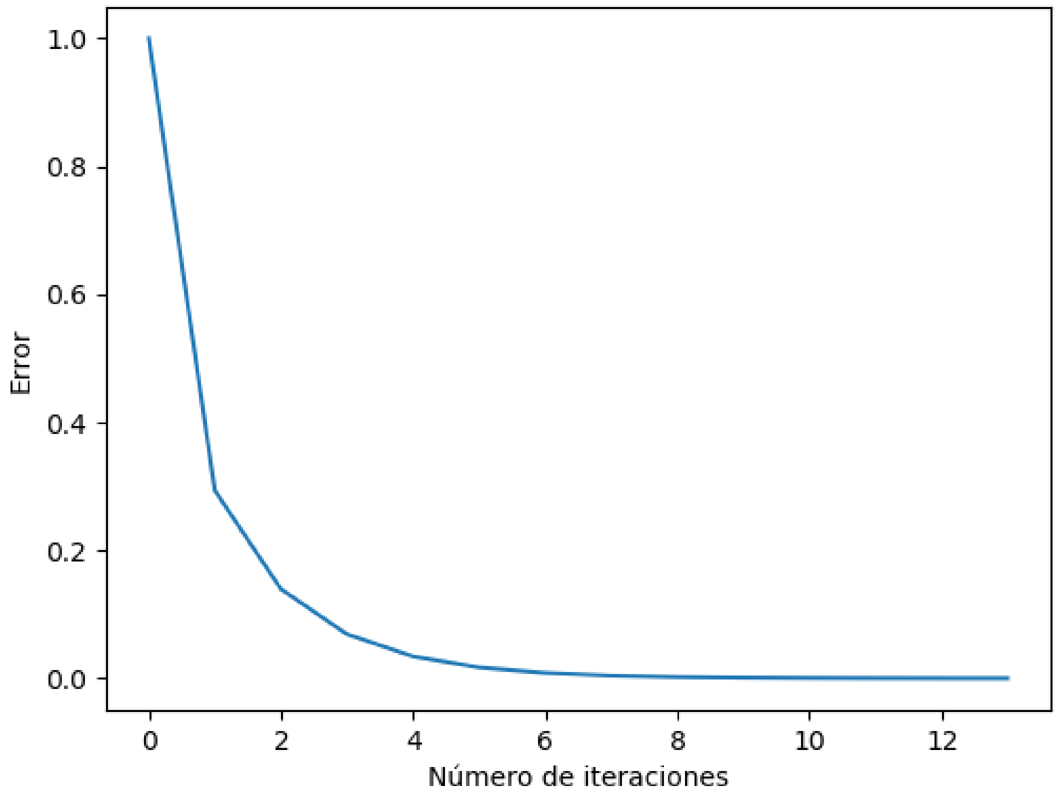
3. Utilice el método de Newton para resolver el problema, muestre gráficamente cómo se comporta la convergencia a la solución

Ejemplo. Una partícula se mueve en el espacio con el vector de 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.

Gráficas, tablas y análisis



Grafica 1. Expresa el proceso de convergencia de la función $2\,cos(t) - 2$ usando el método de Newton, donde le eje X de la grafica expresa el numero de la iteración y el Y el valor del error.



Grafica 2. Expresa el proceso de convergencia de la función $\sin(t) - 1$ usando el método de Newton, donde le eje X de la gráfica expresa el numero de la iteración y el Y el valor del error.

DATOS GRAFICA 1	
ITERACIÓN	ERROR
1	0,747
2	0,2747
3	0,1315
4	0,0651
5	0,0325
6	0,0162
7	0,0081
8	0,0041
9	0,002
10	0,001
11	0,0005
12	0,0003
13	0,0001
14	0,0001

DATOS GRAFICA 2	
ITERACIÓN	ERROR
0	0,2934
1	0,1396
2	0,069
3	0,0344
4	0,0172
5	0,0086
6	0,0043
7	0,0021
8	0,0011
9	0,0005
10	0,0003
11	0,0001
12	0,0001

Viendo el comportamiento de las gráficas y las tablas de datos de donde se obtienen podemos ver que la disminución del error con el paso de las iteraciones describe un comportamiento que puede ser atribuido a la convergencia cuadrática.

Código fuente (python 3)

```
import matplotlib.pyplot as plt
#MODULO PARA ACCIONES GRAFICAS
from pylab import *
#MODULO PARA EL MANEJO DE OPERACIONES MATEMATICAS Y MODULOS
from sympy import *
from array import array
t =  symbols('t')
PosX = 2*cos(t) - 2
PosY = sin(t) - 1
derivadaX = PosX.diff(t)
derivadaY = PosY.diff(t)
#Declaracion del X y Y inicial
X0 = 5
Y0 = 0
#Declarciond e la tolerancia
Tol = 0.0001
cont_iteraciones = 0.0
err = 1.0
#Arreglos comportamiento de Newton
ArrayX = []
ArraYNumIteraciones = []

ArrayY = []
ArraYNumIteracionesY = []
#Ciclo del algoritmo (Iteraciones)
while (err > Tol) & (cont_iteraciones < 50.0):

    modificar = PosX.evalf(subs={t: X0})/derivadaX.evalf(subs={t: X0})
    X1 = X0 - modificar
    err = abs(X1 - X0)

print("{:^10}{:.3f}{:^10}{:.4f}{:^10}{:.4f}{:^10}{:.4f}".format('#iteraci
on: ', cont_iteraciones,'\t X1: ' ,X1, 'error: ', err, 'Modificar: ',
modificar))
    ArrayX.append(err)
    ArraYNumIteraciones.append(cont_iteraciones)
    X0=X1
    cont_iteraciones = cont_iteraciones+1
#REINICIAR VARIABLES
err = 1
cont_iteraciones = 0
while (err > Tol) & (cont_iteraciones < 20.0):

    modificar = PosY.evalf(subs={t: Y0})/derivadaY.evalf(subs={t: Y0})
    Y1 = Y0 - modificar
    err = abs(Y1 - Y0)
    print("{:^10}{:.3f}{:^10}{:.4f}{:^10}{:.4f}{:^10}{:.4f}".format('#i:
', cont_iteraciones, 'Y1: ', Y1, 'E: ', err, ' Modificar: ', modificar))
    ArrayY.append(err)
    ArraYNumIteracionesY.append(cont_iteraciones)
    Y0=Y1
```

```
    cont_iteraciones = cont_iteraciones+1


plt.plot(ArraYNumIteracionesY,ArrayY)
plt.ylabel("Error")
plt.xlabel("Número de iteraciones")
plt.show()
```

4. Resolver por dos métodos diferentes, grafique las soluciones y comparar sus soluciones
Encuentre una intersección de las siguientes ecuaciones en coordenadas polares
$r = 2 + \cos(3 * t)\,, r = 2 - e^t$
Para el desarrollo de este punto se empleó el método de newton y el método secante, obteniendo valores similares en los resultados de cada método evaluado sobre una misma función.
$r = 2 + \cos(3 * t)$

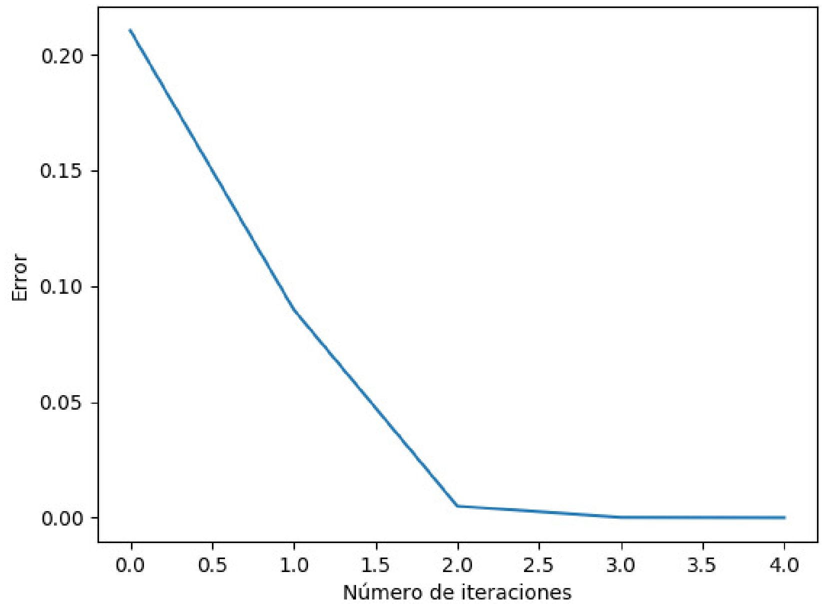
```
from sympy import *

t = symbols('t')
Fn1 = 2+cos(3*t) - 2
derivadaFn1 = Fn1.diff(t)

# Declaracion del X0 inicial
X0 = 1

# Declarcion de la tolerancia
Tol = 0.0001
cont_iteraciones = 0.0
err = 1.0

# Ciclo del algoritmo Newton (Iteraciones)
print('Metodo de Newton')
while (err > Tol) & (cont_iteraciones < 50.0):
    modificar = Fn1.evalf(subs={t: X0}) / derivadaFn1.evalf(subs={t: X0})
    X1 = X0 - modificar
    err = abs(X1 - X0)
    print(
        "{:^10}{:.3f}{:^10}{:.5f}{:^10}{:.4f}".format('#iteracion: ',
cont_iteraciones, '\t X1: ', X1, 'error: ', err))
    X0 = X1
    cont_iteraciones = cont_iteraciones + 1

# Inicializacion de variables
cont_iteraciones = 0.0
err = 1.000
X0 = 1
X1 = 2
# Ciclo del algoritmo Secante (Iteraciones)
print('\nMetodo Secante')
while err > Tol:
    X2 = X1 - (((X1 - X0) * Fn1.evalf(subs={t: X1})) /
(Fn1.evalf(subs={t: X1}) - Fn1.evalf(subs={t: X0})))
    err = abs(Fn1.evalf(subs={t: X2}))
    print(
        "{:^10}{:.3f}{:^10}{:.5f}{:^10}{:.4f}".format('#iteracion: ',
```

```
cont_iteraciones, '\t X2: ', X2, 'error: ', err))
    # print('#iteracion: ', cont_iteraciones,'\t X2: ' ,X2, 'error: ',
err)
    X0 = X1
    X1 = X2
    cont_iteraciones = cont_iteraciones + 1
```

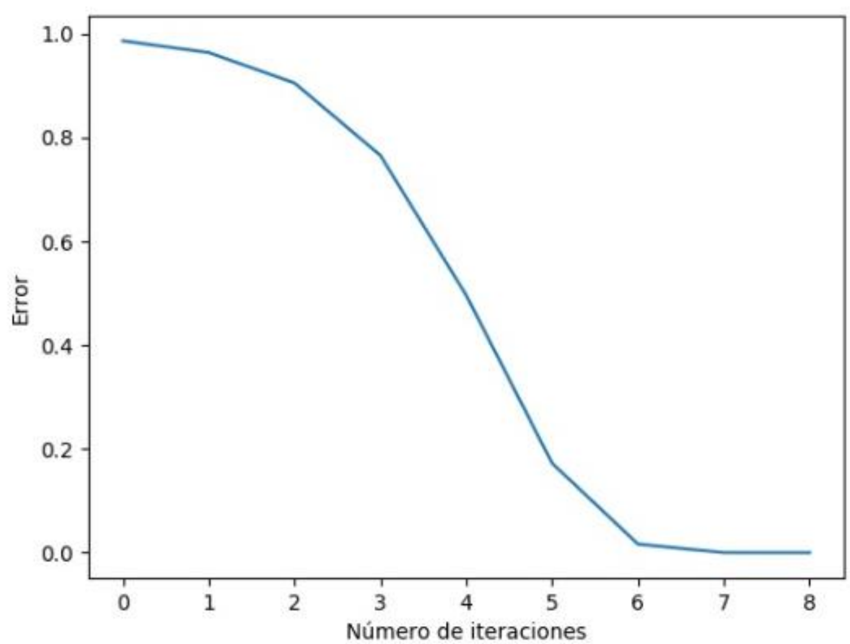
$r = 2 - e^{t}$

```
from sympy import *

t = symbols('t')
Fn1 = 2 - exp(t)
derivadaFn1 = Fn1.diff(t)

# Declaracion del X0 inicial
X0 = 1

# Declarcion de la tolerancia
Tol = 0.0001
cont_iteraciones = 0.0
err = 1.0

# Ciclo del algoritmo Newton (Iteraciones)
print('Metodo de Newton')
while (err > Tol) & (cont_iteraciones < 50.0):
    modificar = Fn1.evalf(subs={t: X0}) / derivadaFn1.evalf(subs={t: X0})
    X1 = X0 - modificar
    err = abs(X1 - X0)
    print(
        "{:^10}{:.3f}{:^10}{:.5f}{:^10}{:.4f}".format('#iteracion: ',
cont_iteraciones, '\t X1: ', X1, 'error: ', err))
    X0 = X1
    cont_iteraciones = cont_iteraciones + 1

# Inicializacion de variables
cont_iteraciones = 0.0
err = 1.000
X0 = 1
X1 = 2
# Ciclo del algoritmo Secante (Iteraciones)
print('\nMetodo Secante')
while err > Tol:
    X2 = X1 - (((X1 - X0) * Fn1.evalf(subs={t: X1})) /
(Fn1.evalf(subs={t: X1}) - Fn1.evalf(subs={t: X0})))
    err = abs(Fn1.evalf(subs={t: X2}))
    print(
        "{:^10}{:.3f}{:^10}{:.5f}{:^10}{:.4f}".format('#iteracion: ',
cont_iteraciones, '\t X2: ', X2, 'error: ', err))
    # print('#iteracion: ', cont_iteraciones,'\t X2: ' ,X2, 'error: ',
err)
    X0 = X1
```

```
    X1 = X2
    cont_iteraciones = cont_iteraciones + 1
```



Grafica 2. Expresa el proceso de convergencia de la función 2-e^t usando el método Secante, donde le eje X de la gráfica expresa el numero de la iteración y el Y el valor del error.



Grafica 2.1. Expresa el proceso de convergencia de la función 2-e^t usando el método de Newton, donde le eje X de la gráfica expresa el numero de la iteración y el Y el valor del error.

5. Resolver los ejercicios 13,14 y 15

13. El siguiente algoritmo permite calcular la raíz n-enésima de un número real a través de operaciones aritméticas básicas, siendo este proceso no muy preciso en el instante de calcular una raíz n no exacta.

Código Fuente

```
# Ingreso de datos
num = float(input('Ingrese el valor de numero a saca raíz: '))
rz = int(input('Ingrese la raiz a sacar: '))
# Inicialización de variables
a = 0  # resultado
cantidad = 0  # raíz más cercana

#Búsqueda de la raiz más cercana o exacta
while cantidad <= num:
    if ( cantidad == num ):
        break
    a = a + 1
    cantidad = a
    for i in range(rz - 1):  # Calculo de raices cercanas
        cantidad = cantidad * a
if cantidad > num:
    a = a - 1

d = (1) / 10  # Declaración de un menor rango de busqueda
#Búsqueda más precisa del valor de la raiz
while (d > 0.01):
    if cantidad == num:
        break
    cantidad = a
    for i in range(rz - 1):
        cantidad = cant * a
    if cantidad < num:
        if cant + d < num:
            a = a + d
            continue
    else:
        break
    d = d / 10
    a = a + d

print('La raiz ', rz, ' de ', num, ' es: ', a)
```

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

Código Fuente

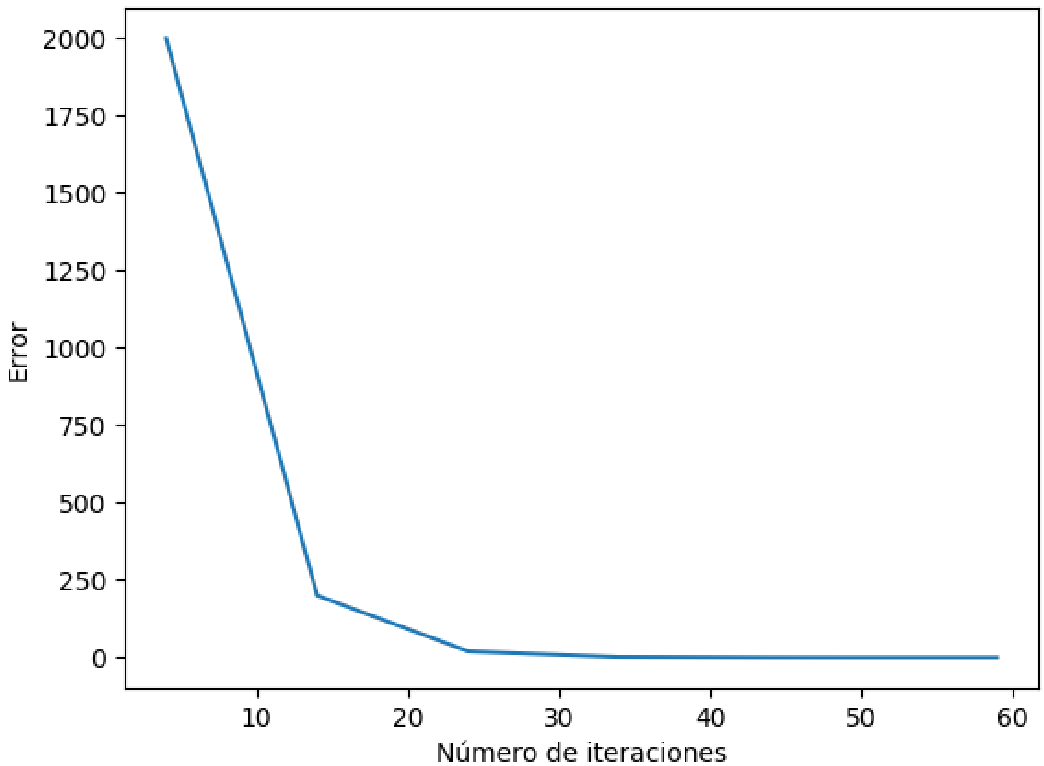
```
#MODULO PARA LE MANEJO DE POLINOMIOS
from sympy import *
#ENTRADA DE POLINOMIO POR CONSOLA
x = symbols('x')
P1 = input("Por favor ingrese el polinomio a evaluar (use como variable
x): ")
```

```
poli = Poly(P1)
#ESPECIFICACION DE LIMITES Y ERROR
a = float(input('Por favor introducir valor de límite innferior(a): '))
b = float(input('Por favor introducir valor de límite superior(b): '))
E = float(input('Por favpr ingrese la precision (E): '))

x1 = a
d = (b-a)/10
x0 = x1
x1 = x1 + d

while abs(d) >= E:
    if poli(x0)*poli(x1) <= 0:
        x1 = x1 - d
        d = d/10
        x1 = x1 + d
    else:
        #print('valor de d = ', d,'X0', x0, "X1", x1, 'polinomio: ',
poli(x0)*poli(x1))
        x0 = x1
        x1 = x1 + d
    if x1 > b:
        print('La función ha superado el límite superior establecido en
el rango, intente con uno nuevo')
        break

print('La raiz es :', x1)
```

a) Condiciones para que la raíz exista, sea única y pueda ser calculada

Para que la raíz exista la función debe estar sometida a un cambio de signo dentro del intervalo establecido, para determinar que la raíz es única la función debe ser derivable, $f'(x)$ debe existir y su solución $f'(x) = 0$ no puede estar contenida en el intervalo dado. Final mente para que la raíz pueda ser calculada la función debe ser continua en el intervalo seleccionado.

b) Orden de convergencia y factor de convergencia del método



Grafica 3. Expresa el proceso de convergencia de la función $x^2 - 3x - 4 = 0$ usando el método intuitivo expresado en el punto 14, donde el eje X expresa el numero de la iteración y el eje Y el error

DATOS GRAFICA 2	
ITERACIÓN	ERROR
0	2000
14	200.000
24	200.000
34	200.000
44	0,20000
49	0,02000
59	0,00200
69	0,00020
79	0,00002

Gracias a la apreciación de la grafica y los datos de donde procede podemos concluir que la convergencia del método es de carácter lineal y su factor de convergencia, debido a la manera en la que se comporta la disminución del error con el paso de las iteraciones, es de $\frac{1}{10}$

15) Se propone resolver la ecuación $\int_0^x (5 - e^u)\, du = 2$ Con el método de punto fijo

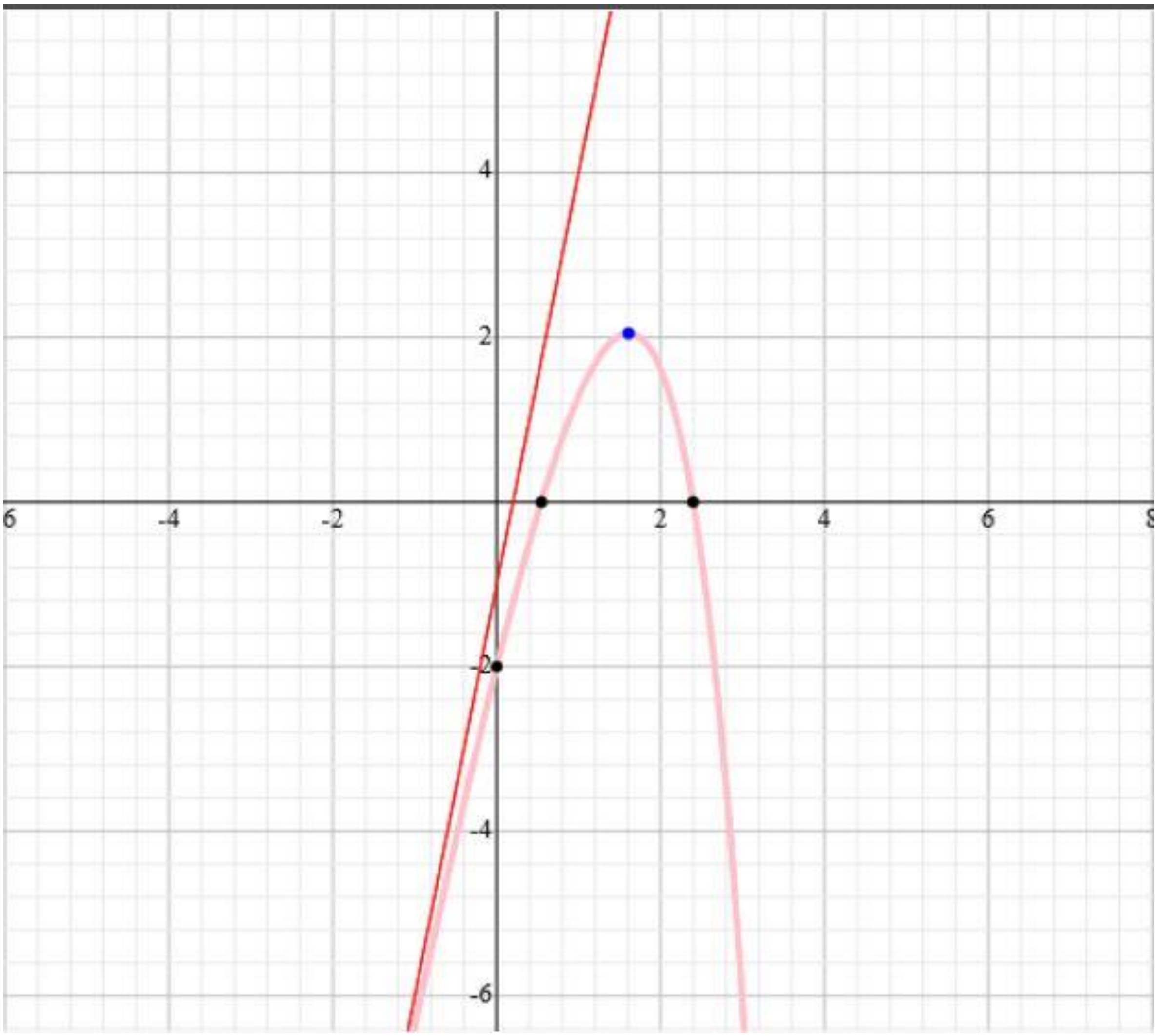
a) Obtener la ecuación f(x) = 0 resolviendo la integral

$\int_0^x (5 - e^u)\, du = 2 \cdots \int_0^x (5)\, du - \int_0^x (e^u)\, du \cdots 5u - e^u + C$

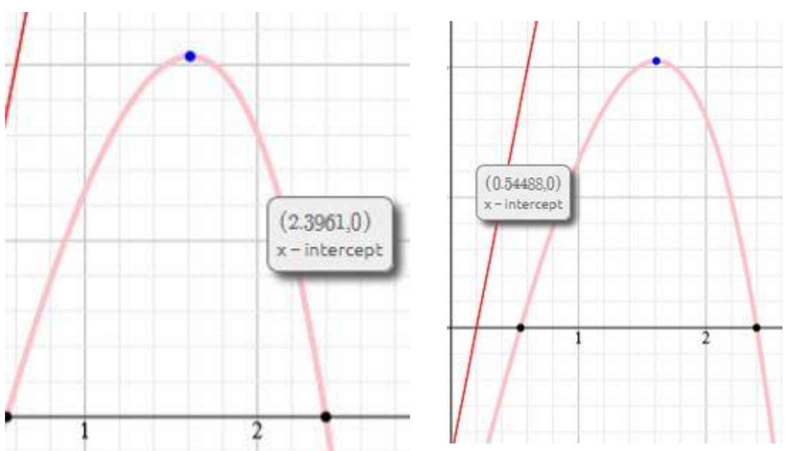
$$\cdots Evaluando \cdots$$

$$5x - e^x + 1 = 2\ Expresando\ de\ la\ forma\ f(x) = 0 \cdots 5x - e^x - 1 = 0$$

b) Mediante un gráfico aproximado localice las raíces reales



Grafica 4. La grafica representa el comportamiento de la función $5x - e^{x} - 1$ resaltando los cortes con los ejes



Graficas 5 y 6. Representan las raíces reales de la función $5x - e^{x} - 1$

Dada la información del gráfico podemos afirmar que las raíces se encuentran en 0.54488 y 2.396

c) Proponer la ecuación equivalente $x = g(x)$, determine el intervalo de convergencia para calcular una de las dos raíces

$$5x - e^x - 1 = 0 \cdots x = \frac{e^x + 1}{5}$$

Intervalo para el cálculo de la primera raíz $[0,2]$

d) Elegir valor inicial y realizar 5 iteraciones con cada iteración verifique que se cumple la condición de convergencia de punto fijo y estime el error de truncamiento del ultimo resultado

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

Valor inicial: 1

PROCESO DE ITERACIÓN

DATOS DE PUNTO FIJO		
ITERACIÓN	APROXIMACIÓN	ERROR
0	0,4983649	0.1973753
1	0,5292055	0.05827717
2	0,5395166	0.0191117
3	0,5430355	0.006480044
4	0,5442447	0.002221853

Validación de la condición de convergencia:

$0{,}4983649 < 0{,}5292055 < 0{,}5395166 < 0{,}5430355 < 0{,}5442447$

Estimación del error de truncamiento:

Valor final: 0,54424

Raíz estimada por la grafica: 0.54488

$E_T = |0{,}54488 - 0{,}54424| = 0{,}00064$