Universidad Católica de Santiago del Estero
Departamento Académico Rafaela
Ingeniería en Informática

TRABAJO PRACTICO 1

Análisis matemático

1 DE SEPTIEMBRE DE 2017

RIBERO JOAQUIN, STORANI GIANFRANCO, TRINCHIERI FACUNDO
UCSE-DAR

Universidad Católica de Santiago del Estero
Departamento Académico Rafaela
Ingeniería en Informática

T. Práctico Nro. 1- Raíces de funciones

Actividad Nro. 1

Bisección:

```
public string Biseccion (double xi, double xd, int ite, double toler)
{
    var funcion = new Funcion();
    Boolean flagSalida = false, flgNoRaiz = false;
    double raizFinal =0;
    int c=0;
    double Tole = toler;
    double xant = 0;
    int Ite = ite;
    while (flagSalida == false)
    {
        if (funcion.Function(xi) * funcion.Function(xd) == 0)
        {
            if (funcion.Function(xi) == 0)
            {
                raizFinal = xi;
                flagSalida = true;
            }
            else
            {
                raizFinal = xd;
                flagSalida = true;
            }
        }
        else if (funcion.Function(xi) * funcion.Function(xd) > 0)
        {
            flgNoRaiz = true;
            flagSalida = true;
        }
        else
        {
            double xr = (xi + xd) / 2;
            double error = Math.Abs((xr - xant) / xr);
            c++;

            if (Math.Abs(funcion.Function(xr))< Tole  || error< Tole|| c>= Ite )
            {
                raizFinal = xr;
                flagSalida = true;
            }
            else if (funcion.Function(xi)*funcion.Function(xr)>0)
            {
                xi = xr;
                xant = xr;
            }
            else
            {
                xd = xr;
                xant = xr;
            }
        }
    }
    if (flgNoRaiz == true)
    {
        return "No se encuentra ninguna raiz entre esos dos valores. Ingrese nuevos.";
    }else return "El resultado de la raiz entre la biseccion es " + raizFinal.ToString("0.##");
}
```

Regla falsa:

```
public string ReglaFalsa(double xi, double xd, int ite, double toler)
{
    var funcion = new Funcion();
    Boolean flagSalida = false, flgNoRaiz = false;
    double raizFinal = 0;
    int c = 0;
    double Tole = toler;
    double xant = 0;
    int Ite = ite;
    while (flagSalida == false)
    {
        if (funcion.Function(xi) * funcion.Function(xd) == 0)
        {
            if (funcion.Function(xi) == 0)
            {
                raizFinal = xi;
                flagSalida = true;
            }
            else
            {
                raizFinal = xd;
                flagSalida = true;
            }
        }
        else if (funcion.Function(xi) * funcion.Function(xd) > 0)
        {
            flgNoRaiz = true;
            flagSalida = true;
        }
        else
        {
            double xr = (funcion.Function(xd) * xi - funcion.Function(xi) * xd) /
                (funcion.Function(xd) - funcion.Function(xi));
            double error = Math.Abs((xr - xant) / xr);
            c++;

            if (Math.Abs(funcion.Function(xr)) < Tole || error < Tole || c >= Ite)
            {
                raizFinal = xr;
                flagSalida = true;
            }
            else if (funcion.Function(xi) * funcion.Function(xr) > 0)
            {
                xi = xr;
            }
            else
            {
                xd = xr;
            }
            xant = xr;
        }
    }
    if (flgNoRaiz == true)
    {
        return "No se encuentra ninguna raiz entre esos dos valores. Ingrese nuevos.";
    }
    else return "El resultado de la raiz entre la regla falsa es " + raizFinal.ToString("0.##");
}
```

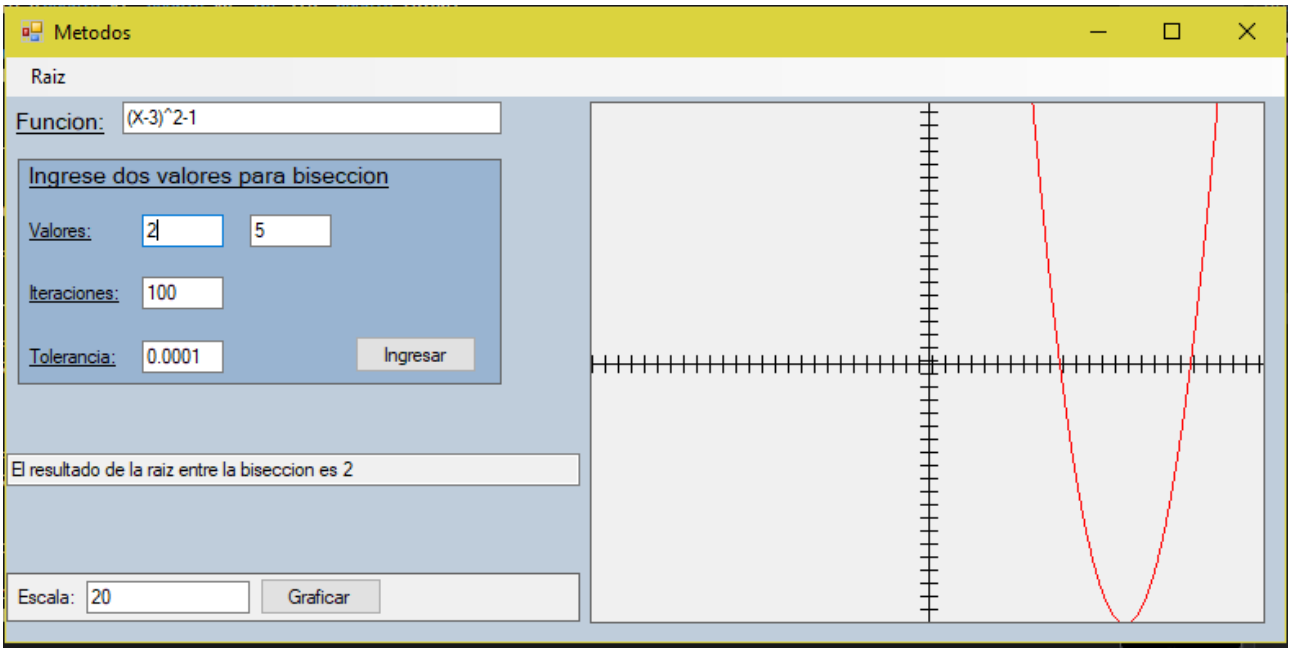
Tangente:

```
public string Tangente(double xi, int ite, double toler)
{
    var funcion = new Funcion();
    Boolean flagSalida = false;
    double raizFinal = 0;
    int c = 0;
    double Tole = toler;
    double xant = 0;
    int Ite = ite;
    double der = (funcion.Function(xi + Tole) - funcion.Function(xi)) / Tole;
    while (flagSalida == false)
    {
        if (Math.Abs(funcion.Function(xi))<Tole)
        {
            raizFinal = xi;
            flagSalida = true;
        }
        else
        {
            c++;
            double xr = xi - funcion.Function(xi) / der;
            double error = Math.Abs((xr - xant) / xr);
            if (Math.Abs(funcion.Function(xr)) < Tole || error < Tole || c >= Ite)
            {
                raizFinal = xr;
                flagSalida = true;
            }
            else
            {
                xi = xr;
                xant = xr;
            }
        }
    }
    return "El resultado de la raiz por la tangente es: " + raizFinal.ToString("0.##");
}
```

Secante:

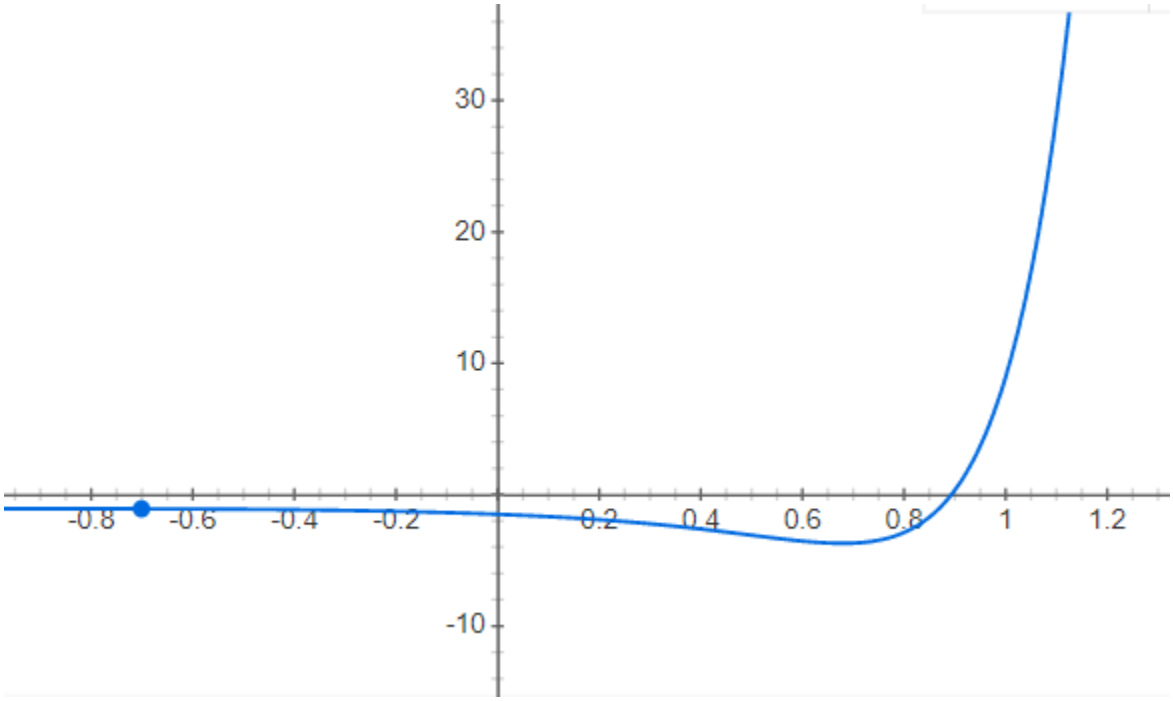
```
public string Secante(double x0, double x1, int ite, double toler)
{
    var funcion = new Funcion();
    Boolean flagSalida = false;
    double raizFinal = 0;
    int c = 0;
    double Tole = toler;
    double xant = x0;
    double x2 = 0;
    int Ite = ite;
    double sec(double xa0, double xa1)
    {
        return ((funcion.Function(xa1) * xa0 - funcion.Function(xa0) * xa1) / (funcion.Function(xa1) - funcion.Function(xa0)));
    };
    while (flagSalida == false)
    {
        if (funcion.Function(x0) * funcion.Function(x1) == 0)
        {
            if (funcion.Function(x0) == 0)
            {
                raizFinal = x0;
                flagSalida = true;
            }
            else
            {
                raizFinal = x1;
                flagSalida = true;
            }
        }
        else
        {
            c++;
            x2 = sec(x0,x1);
            double error = Math.Abs((x2 - xant) / x2);

            if (Math.Abs(funcion.Function(x1)) < Tole || error < Tole || c >= Ite)
            {
                raizFinal = x2;
                flagSalida = true;
            }
            else
            {
                x0 = x1;
                x1 = x2;
                xant = x1;
            }
        }
    }
    return "El resultado de la raiz por la secante es: " + raizFinal.ToString("0.####");
```

Interfaz:


Metodos
Raiz
Funcion:
(X-3)^2-1
Ingrese dos valores para biseccion
Valores:
2
5
Iteraciones:
100
Tolerancia:
0.0001
Ingresar
El resultado de la raiz entre la biseccion es 2
Escala:
20
Graficar

Actividad Nro. 2

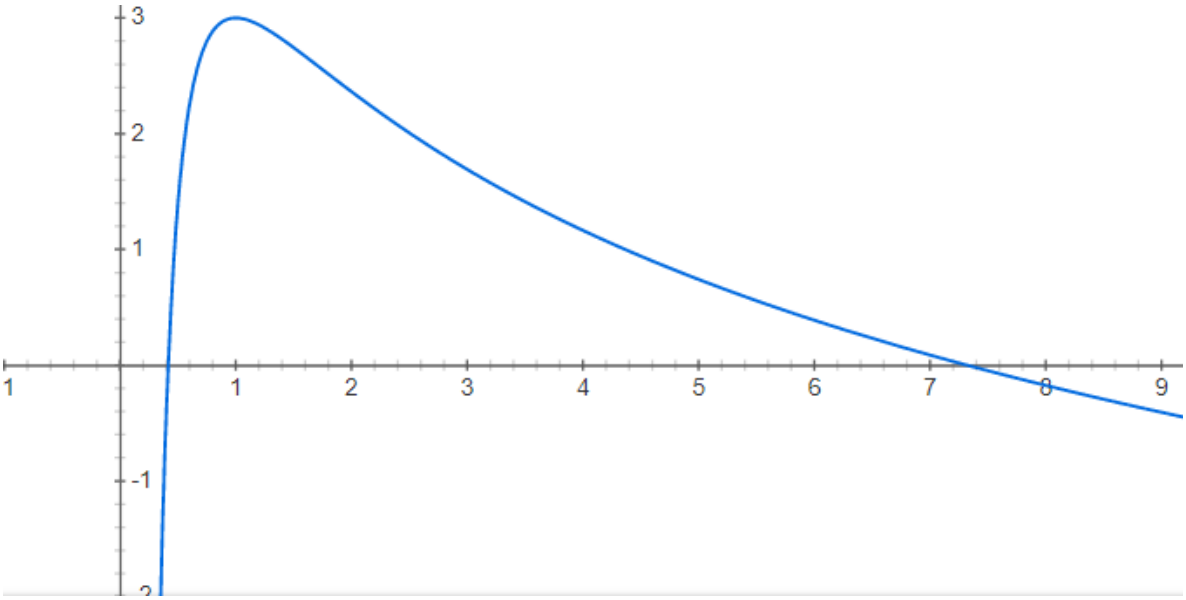
Ejercicio Nro. 1:



Método:	**De la Bisección**	**De la Regla Falsa**
Intervalo	[0,1]	[0,1]
Tolerancia	0.0001	0.0001
X Izquierdo	0	0
X Derecho	1	1
Iteraciones Máximas	100	100
Raíz Encontrada	**0,89325**	**0,893194**
Error	0.0001	0.00004
Iteraciones Usadas	14	16

En el método de la regla falsa se requieren más iteraciones debido a que supone que la raíz se encuentra más cerca del extremo donde el valor absoluto de la función es menor, aunque en este caso posee casi la misma cantidad de iteraciones que la bisección. Aunque para otro tipo de funciones la cantidad de iteraciones de la regla falsa sería mucho mayor a la de la bisección.

Ejercicio Nro. 2:

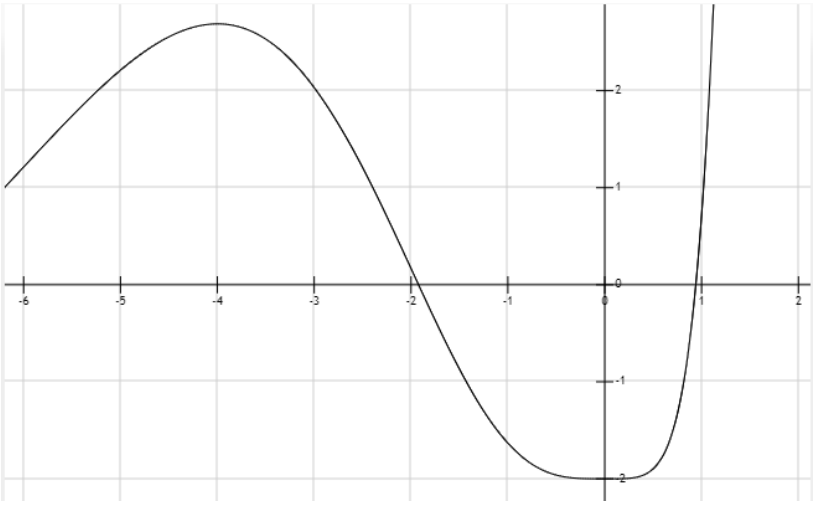


Método:	**Newton-Raphson**	**Secante**
Intervalo	-	[5,7]
Tolerancia	0.0001	0.0001
X Izquierdo	10	5
X Derecho	-	7
Iteraciones Máximas	100	100
Raíz Encontrada	**7,320756**	**7,320435**
Error	0.0001	0,000001
Iteraciones Usadas	8	4

a- El problema que surge con el método de la tangente cuando x=1 es que el método supera las iteraciones, debido a que la función en ese valor es un máximo por lo tanto la tangente es constante.

b- No surgen problemas con el método de la secante entre esos dos valores x0=14 y x1=15.

Ejercicio Nro. 3:

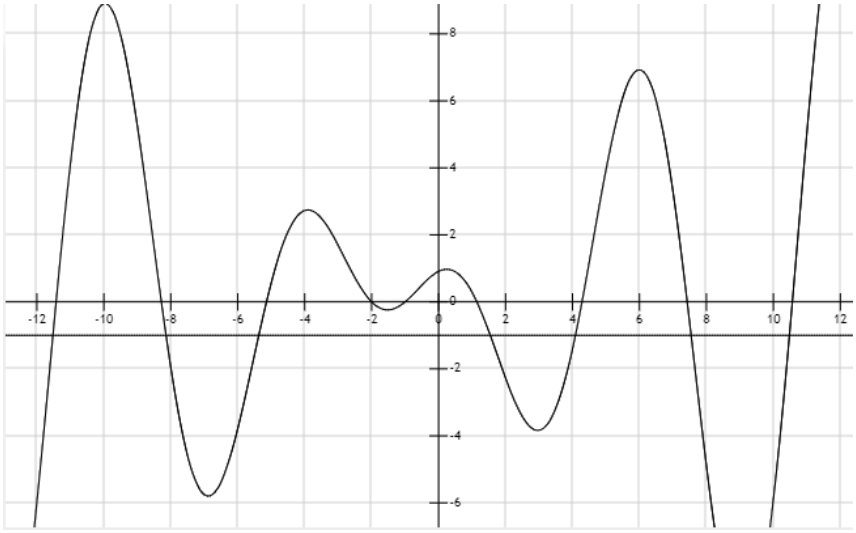


Método:	**Newton-Raphson**	**Secante**
Intervalo	-	[-3,0]
Tolerancia	0.0001	0.0001
X Izquierdo	-3	-3
X Derecho	-	0
Iteraciones Máximas	100	100
Raíz Encontrada	**-1,923512**	**-1,923557**
Error	0.0001	0,000001
Iteraciones Usadas	19	5

a- La función evaluada en esos dos puntos tiene el mismo valor y la recta secante en esos dos puntos es paralela al eje x por lo tanto nunca corta al eje y se pasa en iteraciones.

b-

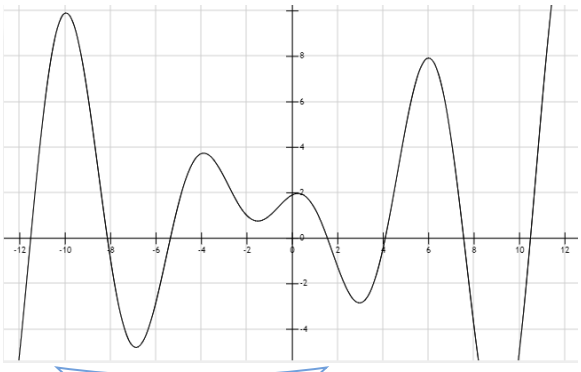
Ejercicio Nro. 4:



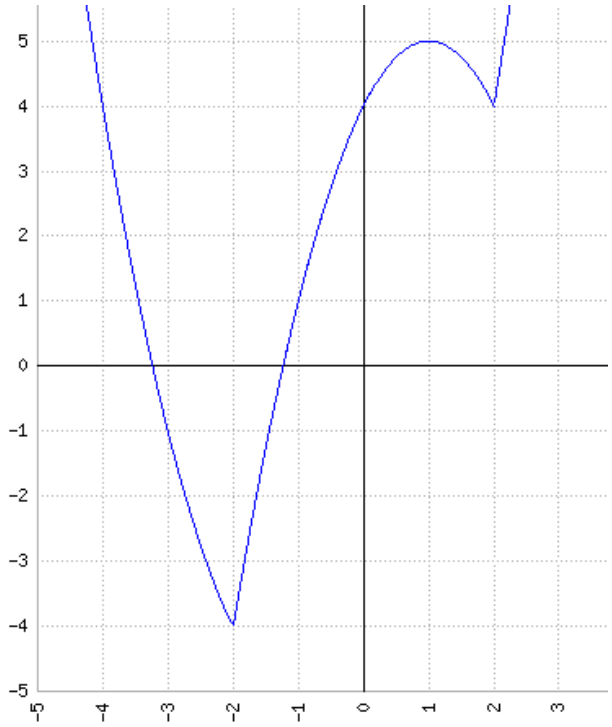
Método:	**Bisección**	**Newton-Raphson**
Intervalo	[-3,-1]	-
Tolerancia	0.0001	0.0001
X Izquierdo	-3	-3
X Derecho	-1	-
Iteraciones Máximas	100	100
Raíz Encontrada	**-5,372559**	**-5,372335**
Error	0,000091	0,000018
Iteraciones Usadas	12	5

a- Para resolverlo tuvimos que desplazar la función hacia arriba una unidad para hacer coincidir la función y=-1 con el eje x y así poder detectar las intersecciones. Utilizando valores en los que la función no sobrepasa un máximo o mínimo, ambos métodos funcionan adecuadamente. Son más efectivos los métodos abiertos en este tipo de funciones ya que encuentran la raíz más rápido.

f(x) = (x+1)*sin(x+2)+1



Ejercicio Nro. 5



Método:	**Bisección X1**	**Bisección X2**
Intervalo	[-2,0]	[-4,-2]
Tolerancia	0.0001	0.0001
X Izquierdo	-2	-4
X Derecho	0	-2
Iteraciones Máximas	100	100
Raíz Encontrada	**-1,236084**	**-3,236084**
Error	0,000198	0,000075
Iteraciones Usadas	13	13

a- El problema que surge si hallamos la raíz por el método de la tangente con valor x0=-2 es que no retorne ninguno valor debido a que la función evaluada en ese punto es un mínimo. Aunque por alguna razón nuestro programa lo calcula igualmente con 7 iteraciones y un resultado de -1,236092 el cual es bastante aproximado. Consideramos que esto se debe a que la función que maneja la aplicación es aproximada a la función real.

b- El problema que surge si intentamos hallar la mayor de las raíces en esos valores por el método de la secante es que no retorna ningún valor y la aplicación se pasa en iteraciones. Debido a que ambos puntos evaluados en la función poseen el mismo valor ***(f(0) =4; f(2)=4)*** y la recta secante queda paralela al eje x.