Resolución de ecuaciones no lineales

```
--> kill(all) $
```

Queremos resolver la ecuación exp(x) - 2*cos(x) = 0.

```
--> f(x) := exp(x) - 2*cos(x) $
    wxplot2d(f(x), [x,0,1]) $
```

1 Bisección

1.1 Primer intento

```
--> kill(all) $
```

```
--> f(x) := exp(x) - 2*cos(x) $
    a : 0.0 $
    b : 1.0 $
    m : (a + b)/2.0 $

    for n : 1 thru 15 step 1 do (
        if f(a)*f(m) < 0
            then
                b : m
            else
                a : m,
        m : (a+b)/2,
        print("Paso: ", n, ", Intervalo: [a,b]=", [a, b], " -> Punto medio: m=", m)
        ) $
```

1.2 EJERCICIO

Modifica la rutina anterior incluyendo un criterio de parada
con una tolerancia de 10^-8.

2 Secante

2.1 Fallo aparente

```
--> kill(all) $
```

```
--> f(x) := exp(x) - 2*cos(x) $
    x0 : 0.0 $
    x1 : 1.0 $

    for n : 2 thru 12 step 1 do (
        x2 : x0 - f(x0)*(x1-x0)/(f(x1) - f(x0)),
        print("Solución aproximada: x", n, "=", x2),
        x0 : x1,
        x1 : x2
        ) $
```

2.2 EJERCICIO

Modifica la rutina anterior incluyendo un criterio de parada
con una tolerancia de 10^-8.

3 Newton-Raphson

3.1 EJERCICIO

Implementa el método de Newton-Raphson para resolver la ecuación

    exp(x) - 2*cos(x) = 0

tomando como punto inicial x0 = 0.5 y un número máximo de 7 iteraciones.

3.2 EJERCICIO

Modifica la rutina del ejercicio anterior y establece un criterio de parada con una tolerancia de 10^-15.

4 Un procedimiento para implementar el método de bisección

Recordemos el programa con inclusión de un criterio de parada que realizamos al principio de esta sesión.

Diseñamos el procedimiento: comando "block".

Primer intento: fijando un número determinado de pasos.

```
--> biseccion(fn, a1, b1) := block( [a, b, c],
        a : float(a1),
        b : float(b1),
        for n:1 thru 15 step 1 do (
            c: (a+b)/2,
            if fn(a)*fn(c) < 0
                then
                    b : c
                else
                    a : c
            ),
        c
        )$
```

```
--> biseccion(f,0,1);
```

```
--> biseccion(f,1,2);
```

```
--> biseccion(f,-1,0);
```

```
--> biseccion(f,-2,-1);
```

```
--> biseccion(f,-1,-2);
```

Controlemos el intervalo.

```
--> biseccion(fn, a1, b1) := block( [a, b, c],
        a : float(a1),
        b : float(b1),
        if fn(a)*fn(b) > 0
            then
                return ("Intervalo no adecuado"),
        for n:1 thru 15 step 1 do (
            c: (a+b)/2,
            if fn(a)*fn(c)<0
                then
                    b:c
                else
                    a:c
            ),
            c
        ) $
```

```
--> biseccion(f,0,1);
```

```
--> biseccion(f,1,2);
```

```
--> biseccion(f,-1,0);
```

```
--> biseccion(f,-2,-1);
```

```
--> biseccion(f,-1,-2);
```

Controlemos el número de iteraciones.

```
--> kill(all)$
```

```
--> f(x) := exp(x) - 2*cos(x) $
```

```
--> biseccion(fn, a1, b1) := block( [a, b, c, m, maxiter:300, tol:10^(-15), prec:10^-10],
        a : float(a1),
        b : float(b1),
        if fn(a)*fn(b)>0
            then
                return ("Intervalo no adecuado"),
        for n:1 while ( n <= maxiter and abs(b-a) > tol and abs(fn((a+b)/2)) > prec) step 1 do (
            m:n,
            c: (a+b)/2,
            if fn(a)*fn(c)<0
                then
                    b:c
                else
                    a:c
            ),
        [m,{c}]
        ) $
```

```
--> biseccion(f,0,1);
```

```
--> biseccion(f,1,2);
```

```
--> biseccion(f,-1,0);
```

```
--> biseccion(f,-2,-1);
```

```
--> biseccion(f,-1,-3);
```

¿qué pasa si, en el intervalo inicial, uno de los extremos es solución?
¿y si la solución es justo el punto medio?

```
--> f(x) := x-1 $
```

```
--> biseccion(f,0,1);
```

```
--> biseccion(f,0,2);
```

```
--> biseccion(f,1,2);
```

Versión definitiva.

```
--> kill(all) $
```

```
--> biseccion(fn,a1,b1):= block( [a, b, c, m, maxiter:300, tol:10^(-15), prec:10^-10],
        a : float(a1),
        b : float(b1),
        c : (a+b)/2,
        if abs(fn(a)) < prec then return([0,{a}]),
        if abs(fn(b)) < prec then return([0,{b}]),
        if fn(a)*fn(b) > 0 then return ("Intervalo no adecuado"),
        if abs(fn(c)) < prec then return([1,{c}]),
        for n : 1 while ( n <= maxiter and abs(b-a) > tol and abs(fn(c)) > prec) step 1 do (
            m : n,
            c : (a+b)/2,
            if fn(a)*fn(c) < 0 then b : c else a : c
            ),
        [m,{c}]
    ) $
```

```
--> f(x) := x^2-1 $

--> biseccion(f,0,2);

--> biseccion(f,0,1);

--> biseccion(f,0,2);

--> biseccion(f,-1,0);

--> biseccion(f,0,5);

--> biseccion(f,5,2);

--> biseccion(f,1,0);

--> biseccion(f,1,-1);
```