Método de bisección

1 Se considera el problema de encontrar las soluciones reales de e^x-3=0 en [0,2].

1.1 Apartado a) ¿Se puede utilizar el método de bisección para resolver dicho problematomando [0,2] como intervalo inicial? ¿Por qué? En caso afirmativo,elabora un programa que calcule las 25 primeras iteraciones de dichométodo y halla una cota del error que se comete si consideramos la última de las iteraciones del apartado anterior comoel valor exacto de la solución.

```
(%i1) kill(all);
(%o0) done
```

Definimos la función

```
(%i1) f(x):=exp(x)-3$
```

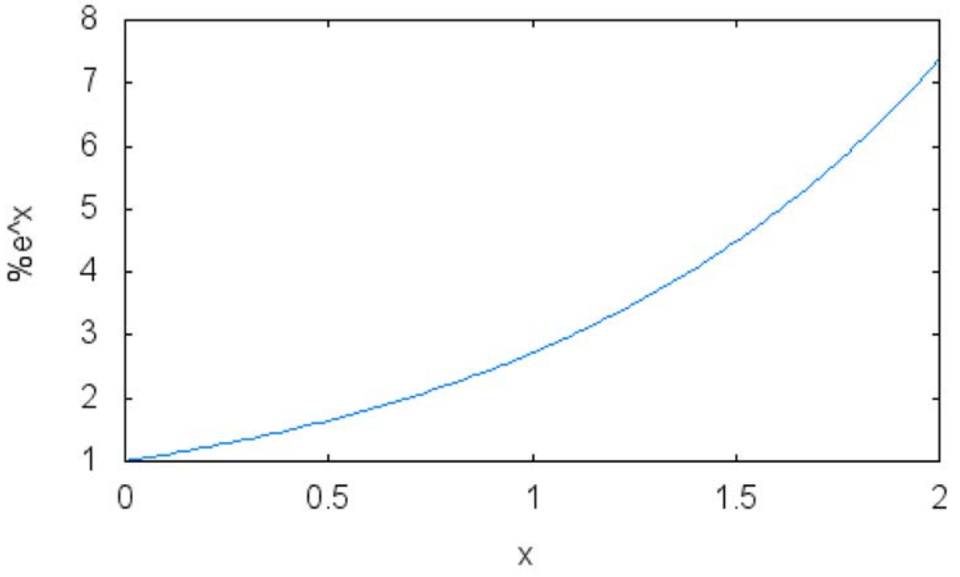
Paso 1 del método de bisección: Considerar el intervalo I = [a1,b1] = [0,2]
tal que f(a1)*f(b1) < 0 siendo f continua y tal que
f tiene un único cero en el intervalo I --> x = (a1 + b1)/2

```
(%i2) a1:0.0$ b1:2.0$
      is(f(a1)*f(b1)<0);
(%o4) true
```

Hago la derivada para saber que el 0 es único, (es todo el rato positiva)

```
(%i5) wxplot2d([diff(f(x),x,1)],[x,0,2]);
```

(%t5)

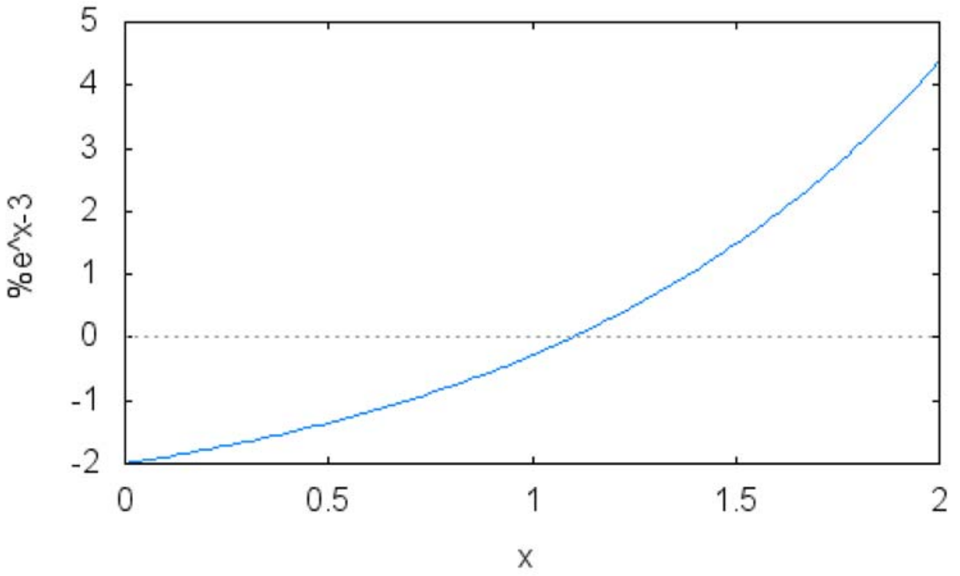


(%o5)

Dibujamos la función

```
(%i6) wxplot2d([f(x)],[x,0,2])$
```

(%t6)



Por tanto, por todas estas cosas, sí podemos usar el método de bisección para resolver el problema.
(Se verifican las condiciones del Teorema de los ceros de Bolzano)

Tomamos una precisión de 32 decimales

Si hago la gráfica para demostrar que hay un 0 decir lo de f(0)<0 y f(2)>2 y solo corta una vez al eje x y por tanto el 0 es único, sino hago lo de 0 de bolzano y ver que la solución es única,con la derivada unicidad del 0

```
(%i7) fpprec:32;
(%o7) 32
```

```
(%i8) a:a1$ b:b1$
```

Paso 2 del método de biseccion:
Vamos eligiendo los puntos medios de los intervalos
- Si f(x^(k-1)) = 0 , entonces s = x^(k-1) (fin).
- Si f(a^(k-1))*f(x^(k-1))< 0 , se dene Ik := [a^(k-1); x^(k-1)].
- Si f(x^(k-1))*f(b^(k-1))< 0 , se dene Ik := [x^(k-1); b^(k-1)].
--> x(k) = (a(k)+b(k))/2 k-ésima iteración

```
(%i10) kill(all)$
       f(x):=exp(x)-3$
       a1:0.0$ b1:2.0$
       a:a1$ b:b1$
       maximoiteraciones:25$
       for  k:1 thru maximoiteraciones do (
           c: (a+b)/2,
           print("iteracion",k,"=",bfloat(c)),
               if(f(a)*f(c)) <0 then
                   b:c
               else
                   a:c
       );
```

iteracion 1 = 1.0b0
iteracion 2 = 1.5b0
iteracion 3 = 1.25b0
iteracion 4 = 1.125b0
iteracion 5 = 1.0625b0
iteracion 6 = 1.09375b0
iteracion 7 = 1.109375b0
iteracion 8 = 1.1015625b0
iteracion 9 = 1.09765625b0
iteracion 10 = 1.099609375b0
iteracion 11 = 1.0986328125b0
iteracion 12 = 1.09814453125b0
iteracion 13 = 1.098388671875b0
iteracion 14 = 1.0985107421875b0
iteracion 15 = 1.09857177734375b0
iteracion 16 = 1.098602294921875b0
iteracion 17 = 1.0986175537109375b0
iteracion 18 = 1.09860992431640625b0
iteracion 19 = 1.098613739013671875b0
iteracion 20 = 1.0986118316650390625b0
iteracion 21 = 1.09861278533935546875b0
iteracion 22 = 1.098612308502197265625b0
iteracion 23 = 1.0986120700836181640625b0
iteracion 24 = 1.09861218929290771484375b0
iteracion 25 = 1.098612248897552490234375b0
(%o7) *done*

Ahora suponiendo que esa es la solución exacta ver cual es la cota
de error, coger el intervalo y entre 2 elevado a las iteraciones

```
(%i8) (b1-a1)/2^25 /*cota del error cometido en la iteracion 25*/;
(%o8) 5.960464477539063 10^-8
```

```
(%i9) bfloat((b1-a1)/2^25);
(%o9) 5.9604644775390625b-8
```

1.2 Apartado b) Modifica el programa que hayas elaborado para el apartado anterior para que, en caso de que no haya un cambio de signo en los extremos del intervalo inicial el programa se interrumpa devolviendo un mensaje avisando de que dicho intervalo es incorrecto. Compruébalo como intervalo inicial a [0,1]

```
(%i10) kill(all)$
       f(x):=exp(x)-3$
       a1:0.0$ b1:1.0$
       a:a1$ b:b1$
       maximoiteraciones:25$
       if(f(a)*f(b))<0 then
           for  k:1 thru maximoiteraciones do (
           c: (a+b)/2,
           print("iteracion",k,"=",bfloat(c)),
               if(f(a)*f(c)) <0 then
                   b:c
               else
                   a:c
       )
       else
           print("intervalo inicial incorrecto")$
intervalo inicial incorrecto
```

1.3 Apartado c) Modifica el programa del apartado anterior para que el cálculo de las iteraciones de bisección se pare en el momento en que la cota del error correspondiente a una iteración sea menor que 10^(-3).Si al superar el maximo de iteraciones no se encuentra la solución que salga el mensaje aumentar el maximo de iteraciones

Ek = (b-a)/2^k es una cota del error que se comete en la k-esima
iteración al aproximar s, y sugiere como criterio de parada
el que consiste en detener el cálculo de las iteraciones
cuando Ek <= tol / para un valor real tol > 0 prefijado.

```
(%i8) kill(all)$
      f(x):=exp(x)-3$
      a1:0.0$ b1:2.0$
      a:a1$ b:b1$
      maximoiteraciones:50$
      tol:10^(-20)$
      if(f(a)*f(b))<0 then
          for  k:1 thru maximoiteraciones do (
          c: (a+b)/2,
      if((b1-a1)/(2^k))< tol then
             print("Se han realizado",k,"Iteraciones"),
      if((b1-a1)/(2^k))< tol then
             print("iteracion",k,"=",bfloat(c)),
      if((b1-a1)/(2^k))< tol then
             print("La cota del error es ",bfloat((b1-a1)/(2^k)) ),
       if((b1-a1)/(2^k))< tol then
             return (false),
       if(k = maximoiteraciones) then
          print("No se ha alcanzado tolerancia, aumentar iteraciones."),
        if(k = maximoiteraciones) then
          print("Se han realizado",k,"Iteraciones"),
      if(k = maximoiteraciones) then
          print("La cota del error es ",bfloat((b1-a1)/(2^k)) ),
          if(f(a)*f(c)) <0 then
                  b:c
              else
                  a:c
      )
      else
          print("intervalo inicial incorrecto");
```

No se ha alcanzado tolerancia, aumentar iteraciones.
Se han realizado 50 *Iteraciones*
La cota del error es 1.7763568394002504646778106689453b-15
(%o8) *done*

1.4 Apartado d) Modifica el programa del apartado anterior de modo que éste calcule en primer lugar el número de iteraciones que hay que realizar para garantizar un error menor que 10^(-3) y 10^(-20) y, a continuación, muestre la última iteración que hay que calcular y por ultimo muestre la cota de error

El número de iteraciones N que es preciso realizar para garantizar
el error que se comete en la N-ésima iteración
sea menor que una cierta cantidad tol > 0, es la parte entera del
número real : 1+(ln((b-a)/tol))/ln(2) tal que I = [a,b]

```
(%i9) kill(all)$
      f(x):=exp(x)-3$
      a1:0.0$ b1:2.0$
      a:a1$ b:b1$
      tol:10^(-20)$
      if(f(a)*f(b))<0 then(
          numeroiteraciones:entier(log((b1-a1)/tol)/log(2) +1),
          print("Tenemos que realizar ", numeroiteraciones, " para garantiz
          for  k:1 thru numeroiteraciones do (
          c: (a+b)/2,
         if( k = numeroiteraciones) then
            print("Iteracion ",k," = ",bfloat(c)),
          if( k = numeroiteraciones) then
            print("La cota del error es : ",(b1-a1)/2^numeroiteraciones ),

          if(f(a)*f(c)) <0 then
                  b:c
              else
                  a:c
          )
      )
      else
          print("intervalo inicial incorrecto")$
```

Tenemos que realizar 68 *para garantizar cota de error menor que tol*

Iteracion 68 = 1.0986122886681095600636126619065b0

La cota del error es : $6.776263578034403\ 10^{-21}$

Newton-Raphson

0.1 Apartado e)¿Se puede aproximar s usando el método de Newton-Raphson que considera x(0) = 2 como aproximación inicial?¿Por qué? En caso afirmativo, elabora un programa que calcule las 6 primeras iteraciones de dicho método. El programa debe mostrar en pantalla todas las iteraciones calculadas con 32 dígitos significativos.

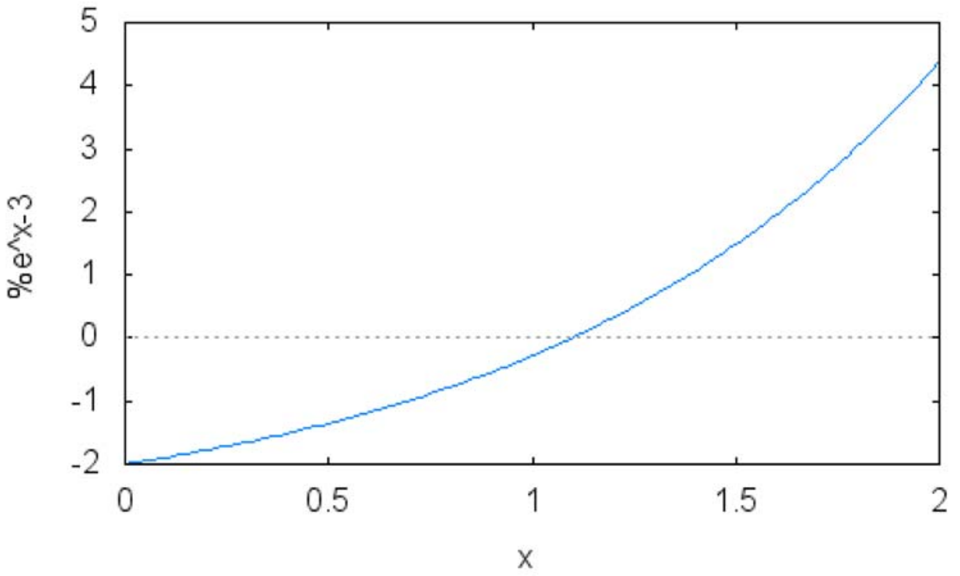
```
Para Newton Raphson partimos de un X^(0) y hacemos intersección del eje
de abscisas con la recta tangente
(x^(k-1),f(x^(k-1))) y hacemos x^(k+1)=x^(k) - f(x^(k))/f'(x^(k))
¡Ojo!
x^(k) pertenezca al dominio de f, que sea derivable y f(x)!=0

Se dibujan las graficas de f,f',f'' en el intervalo [0,2],
se comprueban las condiciones i), ii), iii) del teorema
de convergencia global sobre un intervalo adecuado es decir .
 f perteneciente a C^2([a,b])
i) f(a)·f(b) < 0
ii) f'(x) != 0 para x perteneciente a [a,b]
iii) f''(x) no cambia de signo en [a,b]
```

```
(%i8) kill(all);
(%o0) done
```

```
(%i1) f(x):=exp(x)-3$
```

```
(%i2) wxplot2d([f(x)],[x,0,2]);
```

(%t2)



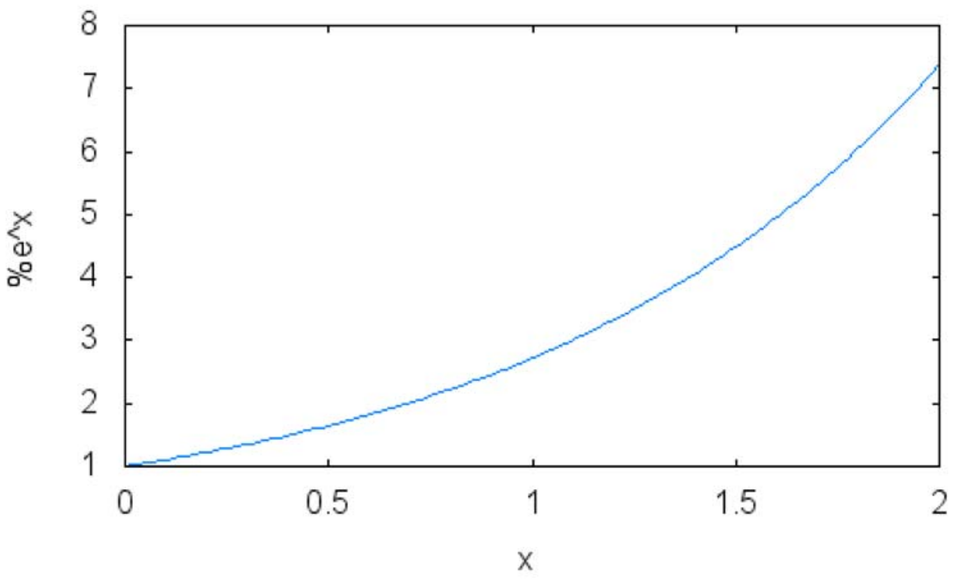
(%o2)

```
(%i3) define(df(x),diff(f(x),x));
```

(%o3) $\mathrm{df}(x):=\%e^{x}$

```
(%i4) wxplot2d([df(x)],[x,0,2]);
```

(%t4)



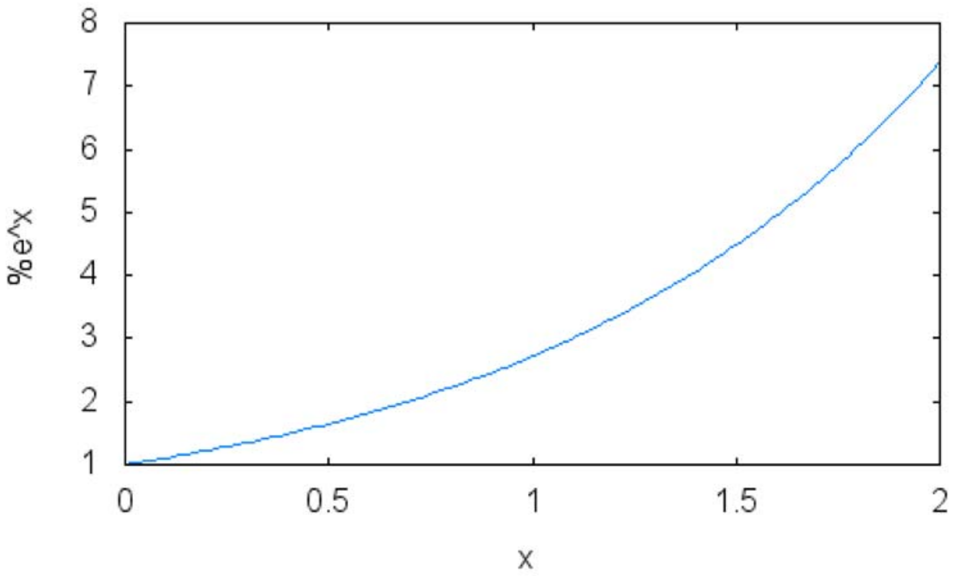
(%o4)

```
(%i5) define(dff(x),diff(df(x),x));
```

(%o5) $\mathrm{dff}(x):=\%e^{x}$

```
(%i6) wxplot2d([dff(x)],[x,0,2]);
```

(%t6)



(%o6)

En estas condiciones podemos usar Newton-Raphson
Habría que comprobar condición iv)
max{|f(a)/f'(a)|,|f(b)/f'(b)| <= b -a
Pero si el signo de f y f' coinciden donde se
da la condicion inicial sabemos que hay conver-
gencia

```
(%i7) x0:bfloat(2.0)$
      niter:6$
      for k:1 thru niter do(
         x1:x0 - f(x0)/df(x0),
         print ("iteracion ",k,"=",bfloat(x1)),
         x0:x1
      );
```

iteracion 1 = 1.4060058497098380756819984849175b0
iteracion 2 = 1.141366984165345661162662616781b0
iteracion 3 = 1.0995133830327887316225171976947b0
iteracion 4 = 1.098612694531720428466069104782b0
iteracion 5 = 1.0986122886681920540193628388166b0
iteracion 6 = 1.0986122886681096913952452403143b0

(%o9) *done*

0.2 Apartado f)Programa que calcule las iteraciones del método anterior hasta un número máximo establecido, de modo que cuando la diferencia en valor absoluto de dos iteraciones consecutivas sea menor que una cierta tolerancia (tol)se detenga el cálculo de las iteraciones,e informe de ello mediante el siguiente mensaje.El programa ademas debe mostrar en pantalla el número de iteraciones que ha calculado y el valor aproximado de s proporcionado por la última de las iteraciones calculadas. Ejecuta el programa para un numero maximo de 20 iteraciones y tol = 10^(-5)

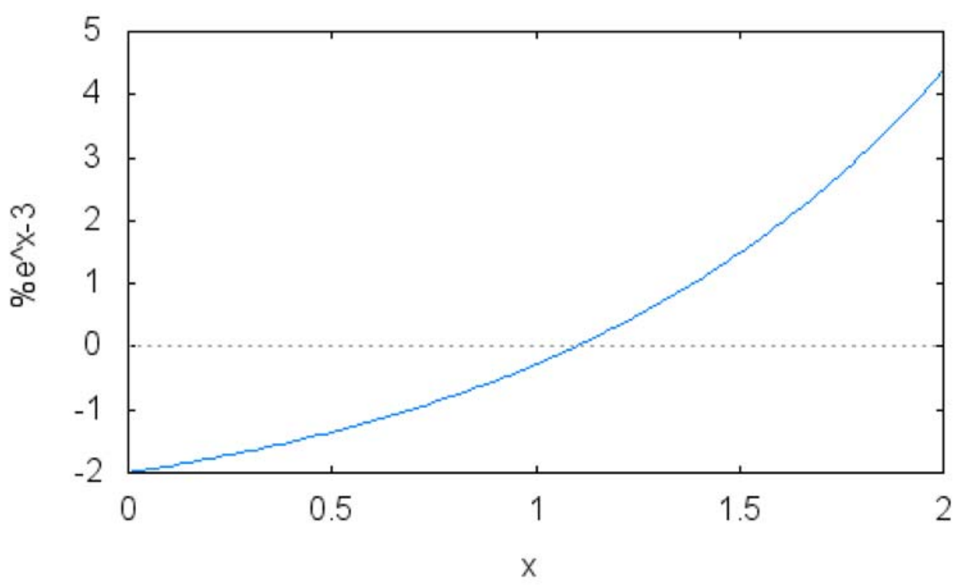
```
(%i10) x0:bfloat(2.0)$
       niter:20$
       tol:10^(-5)$
       for k:1 thru niter do(
           x1: x0 - f(x0)/df(x0),
           if(abs(x0-x1))<tol then
               print("Se ha alcanzado la tolerancia en la ", k, " iteracion"
           if(abs(x0-x1))<tol then
               print("iteracion ",k,"=" ,bfloat(x1)),
           if(abs(x0-x1))<tol then
               return(false),
           if(k=niter) then
               print("Aumentar numero de iteraciones"),
            x0:x1
       );
Se ha alcanzado la tolerancia en la 5 iteracion
iteracion 5 = 1.0986122886681920540193628388166b0
(%o13) false
```

Secante

Partiendo de dos condiciones iniciales vamos a calcular
x^(k+1)= x^(k) - ((x^(k))-x^(k-1))/(f(x^(k))-f(x^(k-1))*f(x^(k)
comprobar que x^(k) y x^(k-1) esta en el dominio y que
f(x^(k)) != f(x^(k-1)). Para la secante solo evaluamos f(x^((k))
porque f(x^(k-1)) ya esta evaluado

0.1 Apartado g) Elabora un programa que calcule las 6 primeras iteraciones del método de la secante que considera como aproximación iniciales x^(-1)=0 y x^0=2.El programa debe mostrar por pantalla todas la iteraciones calculadas

```
(%i14) f(x):=exp(x)-3;
(%o14) f(x):=exp(x)-3

(%i15) wxplot2d([f(x)],[x,0,2]);
(%t15)
```




```
(%o15)

(%i16) fpprec:32;
(%o16) 32

(%i17) x[-1]:0.0$
       x[0]:2.0$
       niter:6$
       for k:1 thru niter do (
           x[k]:x[k-1]-((x[k-1]-x[k-2])/(f(x[k-1])-f(x[k-2])))*f(x[k-1]),
           print("Iteracion ",k,"=",bfloat(x[k]))
       );
Iteracion 1 = 6.2607057099866247895647575205658b-1
Iteracion 2 = 9.0732704083367221592482110281708b-1
Iteracion 3 = 1.1491647450887128023566674528411b0
Iteracion 4 = 1.0936679688714314728770204965258b0
Iteracion 5 = 1.0984882674248903633440477278782b0
Iteracion 6 = 1.0986125955274617105317247478524b0
(%o20) done
```

0.2 Apartado H) Haz un programa que calcule las iteraciones del método del apartado anterior hasta un número máximo establecido,de modo que cuando la diferencia en valor absoluto de dos iteraciones se menor que tol se detenga e informe "se ha alcando la tolerancia". Muestre el nº de iteraciones y el valor aproximado de s

```
(%i21) x[-1]:0.0$
       x[0]:2.0$
       niter:20$
       tol:10^(-5)$
       for k:1 thru niter do (
           if(abs(x[k-1]-x[k-2]))<tol then
               print("Se ha alcanzado la tolerancia en la iteracion",k),
           if(abs(x[k-1]-x[k-2]))<tol then
              print("Iteracion ",k,"=",bfloat(x[k])),
           if(abs(x[k-1]-x[k-2]))<tol then
               return(false),
           x[k]:x[k-1]-((x[k-1]-x[k-2])/(f(x[k-1])-f(x[k-2])))*f(x[k-1]),
           if(k=niter) then
               print("Aumentar el numero iteraciones")
       );
```

Se ha alcanzado la tolerancia en la iteracion 8

Iteracion $8 = x_8$

(%o25) *false*

1 Ejercicio

Se considera el problema de encontrar las soluciones reales de la ecuación x + 1/2 - 2 sen((pi)x) = 0 en [1/2,3/2].

1.1 Apartado a) ¿Se puede utilizar el método de bisección para resolver dicho problema tomando [1/2,3/2] como intervalo inicial? ¿Por qué?

```
(%i26) kill(all);
```

(%o0) *done*

Definimos la función

```
(%i1) f(x):=x+1/2-2*sin((%pi)*x)$
```

Paso 1 del método de bisección: Considerar el intervalo I = [a1,b1]=[1/2,3/2] / f(a1)*f(b1) < 0 siendo f continua y tal que f tiene un único cero en el intervalo I --> x = (a1 + b1)/2

```
(%i2) a1:1/2$ b1:3/2$
      is(f(a1)*f(b1)<0);
```

(%o4) *true*

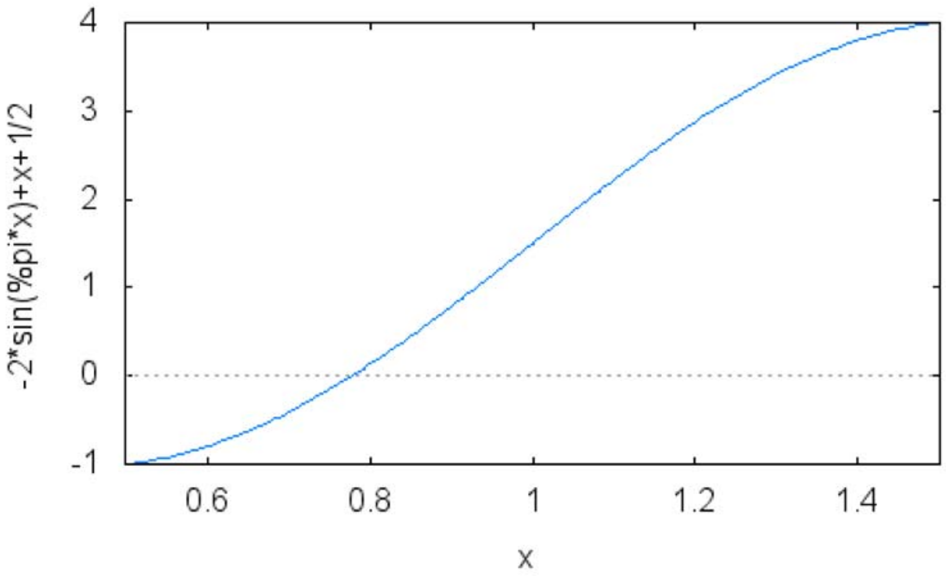
```
(%i5) diff(f(x),x,1);
```

(%o5) $1-2\pi\cos(\pi x)$

Dibujamos la función

```
(%i6) wxplot2d([f(x)],[x,1/2,3/2])$
```

(%t6)



Como vemos, la función tiene un 0 y, por tanto, podemos utilizar el método de bisección.

1.2 Apartado b) Elabora un programa que, partiendo de [1/2,3/2], calcule en primer lugar el número de iteraciones de bisección que hay que realizar para garantizar un error menor que 10^(-5) y, a continuación, muestre todas esas iteraciones.

EL número de iteraciones N que es preciso realizar para garantizar el error que se comete en la N-ésima iteración sea menor que una cierta cantidad tol > 0, es la parte entera del número real :
1+(ln((b-a)/tol))/ln(2) tal que I = [a,b]

```
(%i7) kill(all)$
      f(x):=x + 0.5  - 2*sin((%pi)*x)$
      a1:0.5$ b1:1.5$
      a:a1$ b:b1$
      tol:10^(-5)$
      if (f(a)*f(b)) < 0 then (
          numeroiteraciones:entier(log((b1-a1)/tol)/log(2)+1),
          print("Hay que realizar ", numeroiteraciones," iteraciones para g
          for k:1 thru numeroiteraciones do (
              c:(a+b)/2,
              print("iteracion", k, "=",bfloat( c)),
              if (f(a)*f(c)) < 0 then
                  b:c
              else
                  a:c
          )
      )
      else
          print("Intervalo inicial incorrecto");
```

Hay que realizar 17
iteraciones para garantizar un error menor que tol
iteracion 1 = 1.0b0
iteracion 2 = 7.5b-1
iteracion 3 = 8.75b-1
iteracion 4 = 8.125b-1
iteracion 5 = 7.8125b-1
iteracion 6 = 7.65625b-1
iteracion 7 = 7.734375b-1
iteracion 8 = 7.7734375b-1
iteracion 9 = 7.79296875b-1
iteracion 10 = 7.783203125b-1
iteracion 11 = 7.7880859375b-1
iteracion 12 = 7.79052734375b-1
iteracion 13 = 7.791748046875b-1
iteracion 14 = 7.7911376953125b-1
iteracion 15 = 7.79144287109375b-1
iteracion 16 = 7.79129028320312 5b-1
iteracion 17 = 7.7912139892578125b-1
(%o7) *done*

Como vemos, hay que realizar 17 iteraciones

2 Ejercicio

Demuestra que la ecuación x3 - x - 1 = 0 tiene una única solución s en [1,2] y calcula una aproximación a dicha solución usando el método d bisección que toma [1,2] como intervalo inicial de modo que se garantice un error menor que 10^(-4).

```
(%i8) kill(all);
```
(%o0) *done*

Definimos la función

```
(%i1) f(x):=x^3-x-1;
```
(%o1) $f(x):=x^3-x-1$

Paso 1 del método de bisección: Considerar el intervalo I = [a1,b1] = [1,2] / f(a1)*f(b1) < 0 siendo f continua y tal que f tiene un único cero en el intervalo I --> x = (a1 + b1)/2

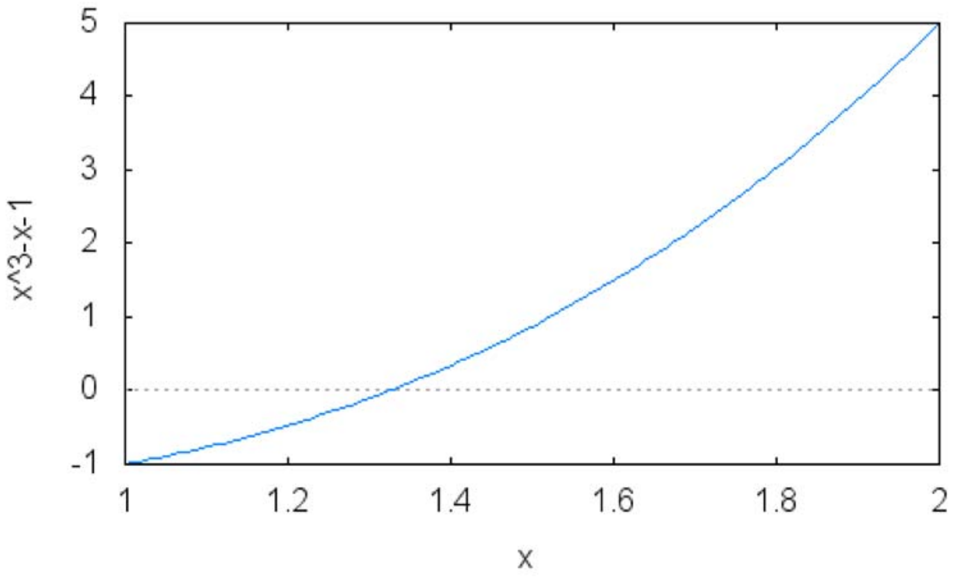
```
(%i2) a1:1/2$ b1:3/2$
      is(f(a1)*f(b1)<0);
```
(%o4) *true*

Dibujamos la función

```
(%i5) diff(f(x),x,1);
```
(%o5) $3x^2-1$

(%i6) wxplot2d([f(x)],[x,1,2]);

(%t6)



(%o6)

```
(%i7) kill(all)$
      f(x):= x^3 -x -1$
      a1:1$ b1:2$
      a:a1$ b:b1$
      tol:10^(-4)$
      if (f(a)*f(b)) < 0 then (
          numeroiteraciones:entier(log((b1-a1)/tol)/log(2)+1),
          print("Hay que realizar ", numeroiteraciones," iteraciones para g
          for k:1 thru numeroiteraciones do (
              c:(a+b)/2,
              print("iteracion", k, "=",bfloat( c)),
              if (f(a)*f(c)) < 0 then
                  b:c
              else
                  a:c
          )
      )
      else
          print("Intervalo inicial incorrecto")$
```

Hay que realizar 14
iteraciones para garantizar un error menor que tol
iteracion 1 = 1.5b0
iteracion 2 = 1.25b0
iteracion 3 = 1.375b0
iteracion 4 = 1.3125b0
iteracion 5 = 1.34375b0
iteracion 6 = 1.328125b0
iteracion 7 = 1.3203125b0
iteracion 8 = 1.32421875b0
iteracion 9 = 1.326171875b0
iteracion 10 = 1.3251953125b0
iteracion 11 = 1.32470703125b0
iteracion 12 = 1.324951171875b0
iteracion 13 = 1.3248291015625b0
iteracion 14 = 1.32476806640625b0

Hay que realizar 14 iteraciones para garantizar un error menor que 10^(-4).

```
(%i8) kill(all);
(%o0) done
```