Iteración Funcional

Se basan en la idea de transformar la ecuación no lineal f(x)=0
en otra ecuación equivalente del tipo g(x)=x en cuyo caso resolver
f(x)=0 equivale a encontrar los puntos fijos de la función g

-Paso 1: Se considera un numero real x^(0) como aprox inicial
-Paso 2: Construimos {x(k)} mediante:
    x^(k+1) = g(x^(k)), k>=0

Cuando converge es + veloz que bisección y + lento que N-Rapson

1 Se quiere aproximar la unica solución real s de la ecuacion x^2-3=0 en el intervalo [1,2]. Para ello se consideran, sobre dicho intervalo, las siguientes funciones:
g1(x)=x^2+x+3
g2(x)=3/x
g3(x)=1/2(x+3/x)*
a)Justifica, porque gi(x) tiene a s como único punto fijo en [1,2]

Definimos las 4 funciones

```
--> f(x):= x^2-3;
```
(%o2) $f(x):=x^2-3$

```
--> g1(x):= x^2+x-3;
```
(%o3) $g1(x):=x^2+x-3$

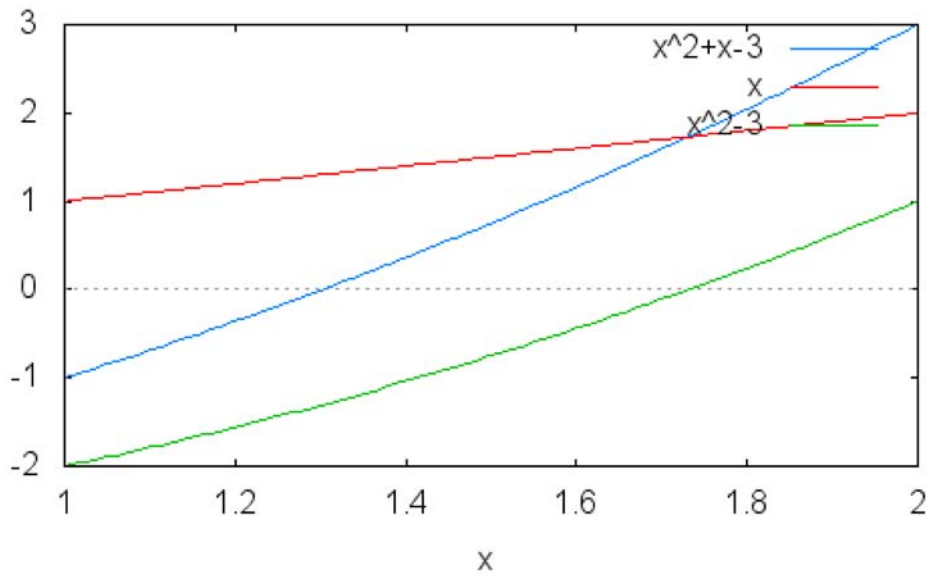
```
--> g2(x):=3/x;
```
(%o4) $g2(x):=\frac{3}{x}$

```
--> g3(x):= 1/2*(x+3/x);
```
(%o5) $g3(x):=\frac{1}{2}\left(x+\frac{3}{x}\right)$

Declaramos la identidad

```
--> h(x):=x;
```
(%o6) $h(x):=x$

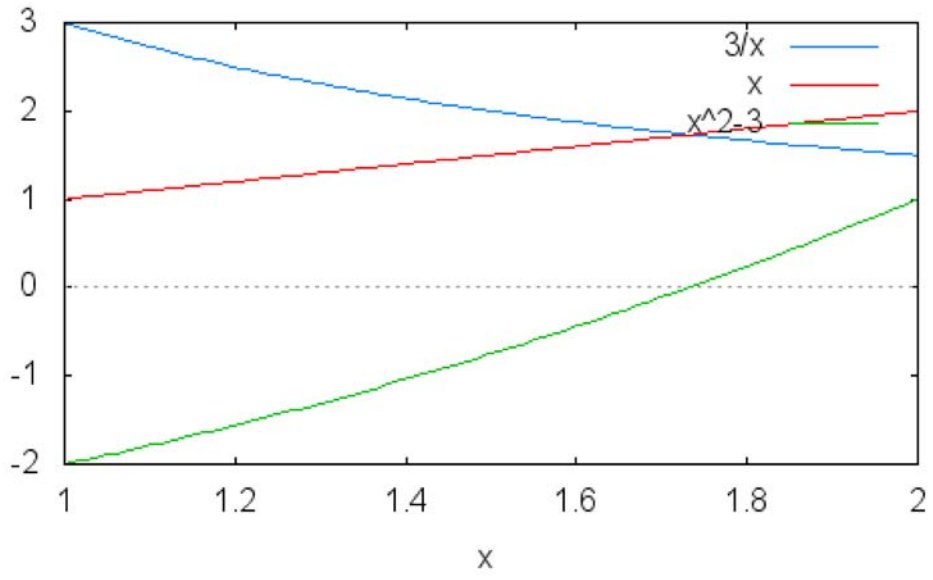
```
--> wxplot2d([g1(x),h(x),f(x)],[x,1,2]);
```



(%t7)

(%o7)

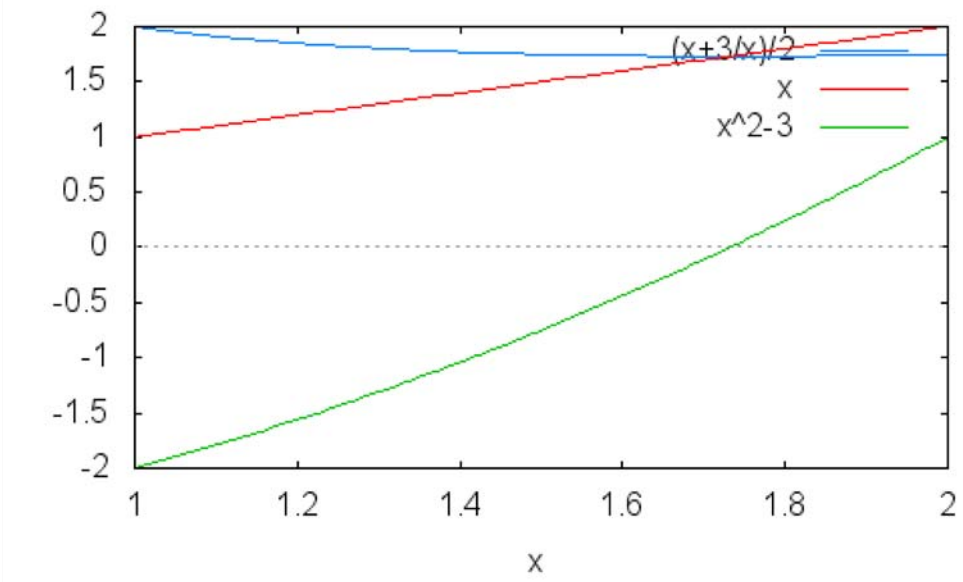
```
--> wxplot2d([g2(x),h(x),f(x)],[x,1,2]);
```



(%t8)

(%o8)

```
--> wxplot2d([g3(x),h(x),f(x)],[x,1,2]);
```



(%t9)

(%o9)

Las funciones tienen un punto fijo que además es único, como vemos en la gráfica f solo corta una vez al eje y nuestra g corta una vez h

1.1 Para cada gi . Calcule las 10 primeras iteraciones del metodo que consideran x^(0)=2. El progrma debe mostrar en pantalla todas las iteraciones calculada con 32 digitos.¿Pára cuás/es de las funciones gi, consideradas aproximan a s. Justifica la respuesta y da un valor aproximado para s

```
--> fpprec:32$
```

Para g1

```
(%i12) x[0]:bfloat(2.0)$
       niter:10$
       for k:1 thru niter do(
         x[k]:g1(x[k-1]),
         print ("iteracion",k,"=",bfloat(x[k]))
       );
iteracion 1 = 3.0b0
iteracion 2 = 9.0b0
iteracion 3 = 8.7b1
iteracion 4 = 7.653b3
iteracion 5 = 5.8576059b7
iteracion 6 = 3.431154746547537b15
iteracion 7 = 1.17728228947556962997757473139 03b31
iteracion 8 = 1.38599358911283892634393086983 66b62
iteracion 9 = 1.92097822906188897801739541260 39b124
iteracion 10 = 3.69015735652975119977678258739 3b248
(%o13) done
```

Para g2

```
(%i14) x[0]:2.05$
       float(x[0]);
       niter:10$
       for k:1 thru niter do(
         x[k]:g2(x[k-1]),
         print ("iteracion",k,"=",bfloat(x[k]))
       );
(%o15) 2.05
iteracion 1 = 1.463414634146341652964906643319 4b0
iteracion 2 = 2.049999999999999822364316059975b0
iteracion 3 = 1.463414634146341652964906643319 4b0
iteracion 4 = 2.049999999999999822364316059975b0
iteracion 5 = 1.463414634146341652964906643319 4b0
iteracion 6 = 2.049999999999999822364316059975b0
iteracion 7 = 1.463414634146341652964906643319 4b0
iteracion 8 = 2.049999999999999822364316059975b0
iteracion 9 = 1.463414634146341652964906643319 4b0
iteracion 10 = 2.049999999999999822364316059975b0
(%o17) done
```

Para g3

```
(%i19) x[0]:bfloat(2.0)$
       niter:10$
       for k:1 thru niter do(
         x[k]:g3(x[k-1]),
           s:x[k],
         print ("iteracion",k,"=",bfloat(x[k]))
       );
iteracion 1 = 1.75b0
iteracion 2 = 1.732142857142857142857142857142857b0
iteracion 3 = 1.732050810014727540500736377025b0
iteracion 4 = 1.732050807568877295254353946072 2b0
iteracion 5 = 1.732050807568877293527446341505 9b0
iteracion 6 = 1.732050807568877293527446341505 9b0
iteracion 7 = 1.732050807568877293527446341505 9b0
iteracion 8 = 1.732050807568877293527446341505 9b0
iteracion 9 = 1.732050807568877293527446341505 9b0
iteracion 10 = 1.732050807568877293527446341505 9b0
(%o20) done
```

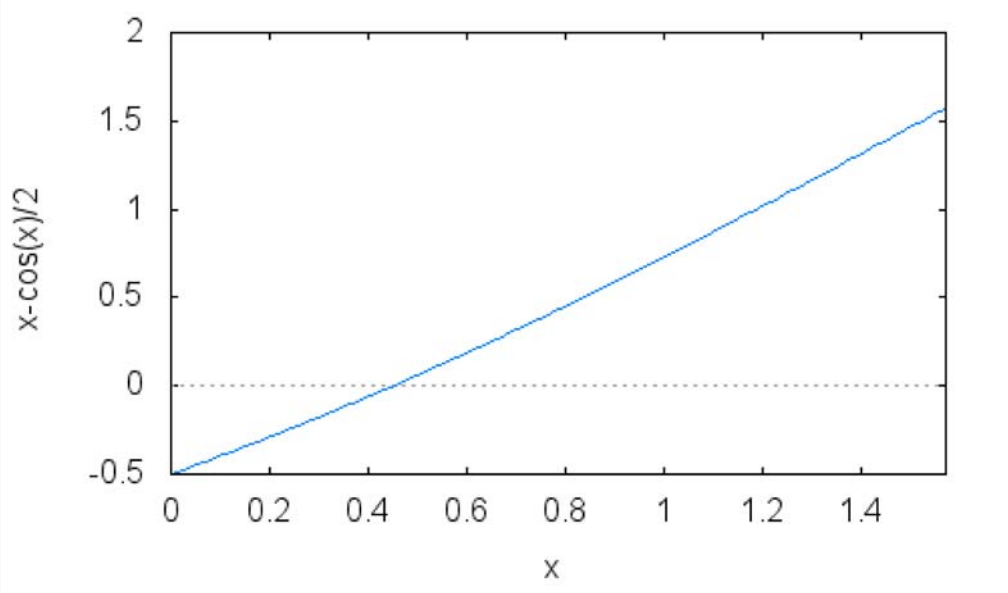
2 Dada la ecuación x-(1/2)cos(x)=0, se pide:

2.1 Razona por qué tiene una única solución s en el intervalor [0,pi/2].

```
(%i21) f(x):=x -1/2 *cos(x);
```

$$(\%o21)\quad f(x):=x-\frac{1}{2}\cos(x)$$

```
(%i22) wxplot2d([f(x)],[x,0,%pi/2]);
```



(%t22)

(%o22)

Tiene una única solución porque corta el eje X una sola vez, y vemos que la función es continua y por tanto la solución es unica.
Sino aplicar Teorema de los 0 de bolzano

```
--> diff(f(x),x);
```

$$(\%o23)\quad \frac{\sin(x)}{2}+1$$

Podemos comprobar que la f' es creciente y solo habra una solución
Dibujo

2.2 Describe un método de iteración funcional =! N-R que considerando x^(0) como aproximación inicial, permita aproximar s. Razona la respuesta. ¿Qué orden de convergencia tiene la sucesión de iteraciones generada por dicho método? Justifica la respuesta.

Buscamos nuestra función g, en nuestro caso

```
--> g(x):=cos(x)/2;
```

$$(\%o24)\quad g(x):=\frac{\cos(x)}{2}$$

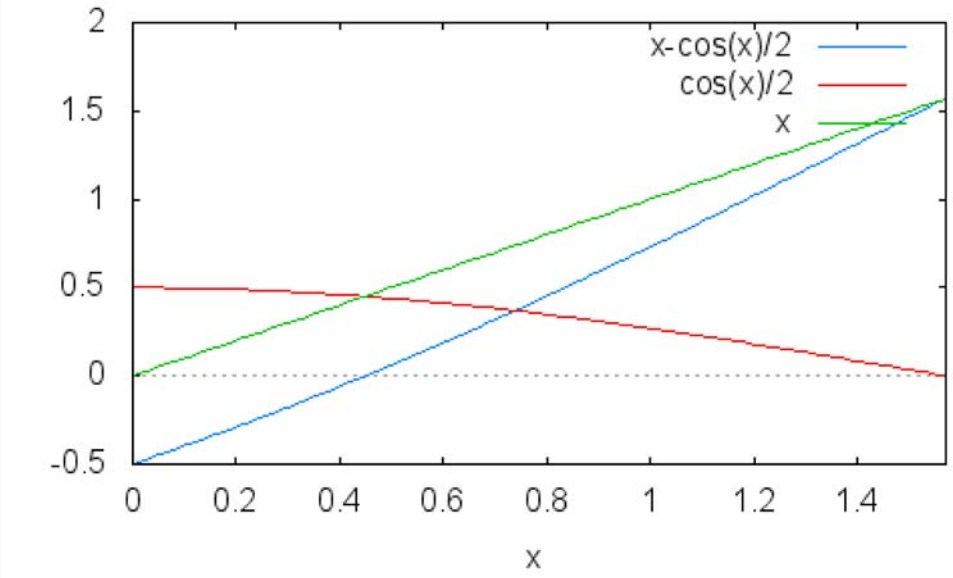
Vemos que g verifica las hipótesis el Corolario 5, es decir:
- g pertece a C'([a,b])(por ser cos funcion continua)
- g([a,b]) c [a,b] (los extremos en primer lugar, y
|g'(x)| <=1 para todo x en [a,b]
(g es contractil y verifica 1),2),3) del Teorema 9 de convergencia

Tiene al menos orden de convergencia 1 por el TH 10

La convergencia local tenemos C1(Us) entorno de un punto fijo
con |g'(s)|<1. |x^(k) - s| < E la sucesion converge a s con al menos
orden de convergencia 1 , y satisface 3) del global
    a) |x^(k) -s| <=L^k/1-k |x^(1)-x^(0)

```
--> wxplot2d([f(x),g(x),h(x)],[x,0,%pi/2]);
```

(%t25)



(%o25)

```
--> x[0]:bfloat(1.0)$
    niter:12$
    for k:1 thru niter do(
      x[k]:g(x[k-1]),
      print ("Iteracion",k,"=",bfloat(x[k]))
    );
Iteracion 1 = 2.70151152934069858700468303721 49b-1
Iteracion 2 = 4.81865284058521046065427460076 85b-1
Iteracion 3 = 4.43066015445310465085854300724b-1
Iteracion 4 = 4.51720737876996162519125639699 32b-1
Iteracion 5 = 4.49848653984115492100259209763 04b-1
Iteracion 6 = 4.50256461170206212878579386547 58b-1
Iteracion 7 = 4.50167760485884790696775044053 89b-1
Iteracion 8 = 4.50187059826385387698476580569 63b-1
Iteracion 9 = 4.50182861010957124041550957953 27b-1
Iteracion 10 = 4.50183774530589405940470596992 2b-1
Iteracion 11 = 4.50183575780418947341353175893 76b-1
Iteracion 12 = 4.50183619021590863883028065428 72b-1
(%o28) done
```

Vemos que converge

2.3 Elabora un programa que calcule las 12 primeras iteraciones del método de iteración funcional descrito en el apartado anterior. El programa solo debe mostrar por pantalla, con 32 digitos significativos,el valor aprox de s proporcionado por la ultima iteracion

```
(%i30) x[0]:bfloat(1.0)$
       niter:12$
       for k:1 thru niter do(
         x[k]:g(x[k-1]),
           if(k=niter) then
         print ("Iteracion",k," = ",bfloat(x[k]))
       );
Iteracion 12  =  4.5018361902159086388302806542872b-1
(%o31) done
```

2.4 Programa que calcule iteraciones hasta un numero maximo establecido, de modo que cuando la diferecia en valor de absoluto de dos iteraciones consecutiva sea menor que tol se detenga, informe de ello. El programa debe mostrar todas las iteraciones calculadas Iteraciones 50, tol=10^(-12). ¿Cuantas iteraciones hace?

```
(%i32) x[0]:bfloat(1.0)$
       niter:50$
       tol:10^(-12)$
       for k:1 thru niter do(
         x[k]:g(x[k-1]),
       if(abs(x[k]-x[k-1])) < tol  then
         print ("Se ha alcanzado la tolerancia "),
        if(abs(x[k]-x[k-1])) < tol  then
         print ("Iteracion",k," = ",bfloat(x[k])),
       if(abs(x[k]-x[k-1])) < tol  then
         return(false),
       if(k=niter)  then
         print ("No se ha alcanzado la tolerancia, aumentar el numero de ite
       );
Se ha alcanzado la tolerancia
Iteracion 19  =  4.5018361129469528416166272764251b-1
(%o35) false
```

3 Comprobar que s=-2 es un punto fijo de g(x)=1+x-x^2/4 tal que |g'(s)|>1. Elabora un pro que calcule las 15 primeras iteraciones de itera funcional a g con x^(0)=-2.05 como aprox inicial El programa debe mostrar en pantalla, todas las iteraciones calculadas.¿Permite aproximar s? ¿Por qué?. ¿Era de esperar la respuesta anterior ¿Por qué?. Dibujalas en [-4,4].

```
--> g(x):=1+x-x^2/4;
```

$$(\%o36)\quad g(x):=1+x+\frac{-x^2}{4}$$

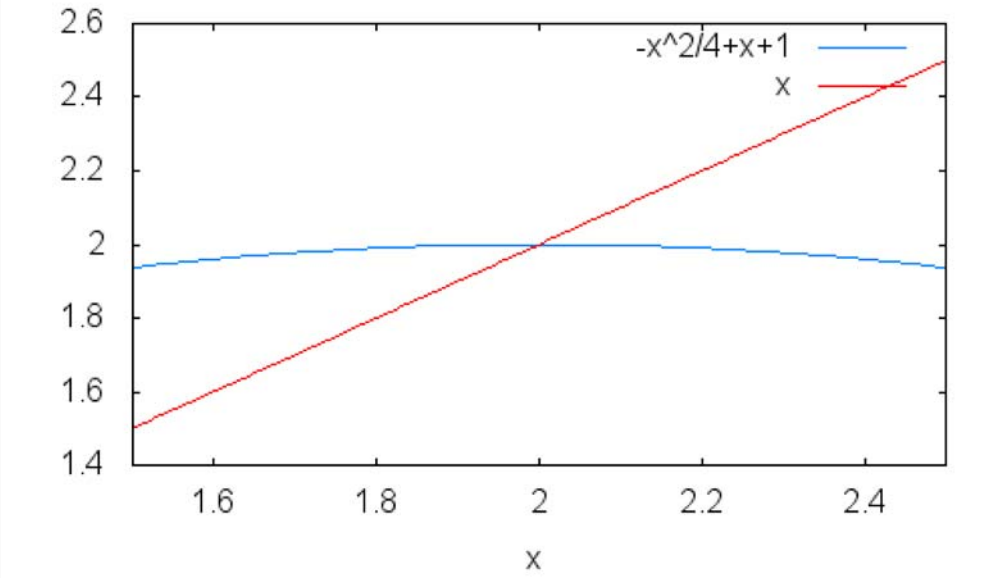
```
--> fpprec:32$
```

```
--> h(x):=x;
```

$$(\%o38)\quad h(x):=x$$

```
--> wxplot2d([g(x),h(x)],[x,1.5,2.5]);
```

(%t39)



(%o39)

```
--> g(2);
```

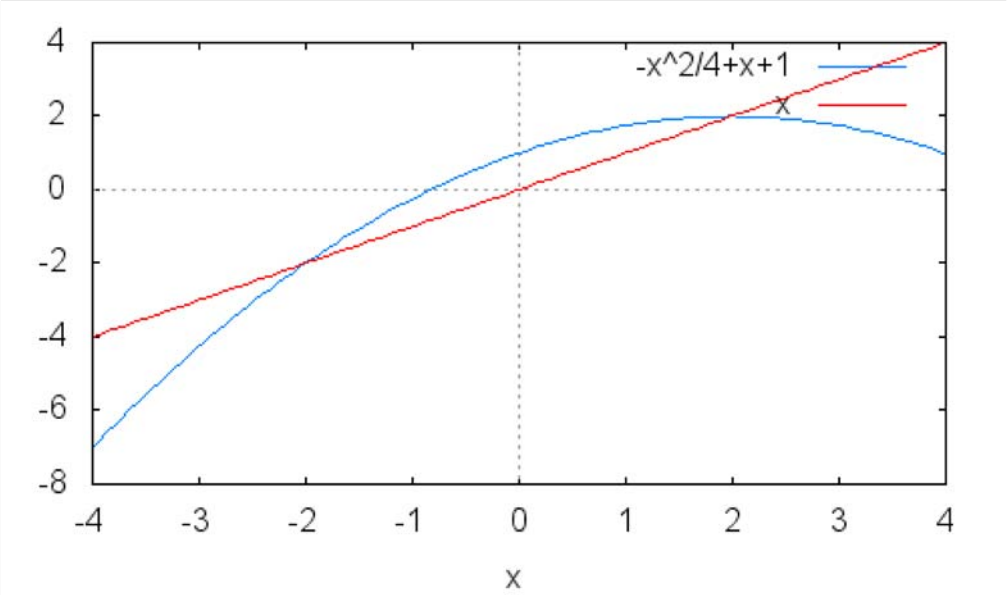
(%o40) 2

Como vemos el punto 2 es un punto fijo,además en la gráfica se veia.

```
(%i43) x[0]:bfloat(-2.05)$
       niter:15$
       for k:1 thru niter do(
         x[k]:g(x[k-1]),
         print ("iteracion",k,"=",bfloat(x[k]))
       );
iteracion 1 = -2.1006249999999996402877400214493b0
iteracion 2 = -2.2037813476562492624774569627278b0
iteracion 3 = -2.4179444047256478066502860675807b0
iteracion 4 = -2.8795581908116646374643938434578b0
iteracion 5 = -3.9525220343793014398010817770426b0
iteracion 6 = -6.8581296424427743780078993229996b0
iteracion 7 = -1.7616615190580838462323955239062b1
iteracion 8 = -9.4202897883831726326564514851565b1
iteracion 9 = -2.3117493903117387118529673084088b3
iteracion 10 = -1.3383570602919856522188896958774b6
iteracion 11 = -4.4780124356441172153989396908845b11
iteracion 12 = -5.0131488434906198807870756296567b22
iteracion 13 = -6.2829153317478349061709413428237b44
iteracion 14 = -9.8687562664780015889089911964864b88
iteracion 15 = -2.4348087561787206276908902866782b177
(%o43) done
```

No lo permite porque como vemos los resultados no son convergentes. Como sabiamos que la derivada en valor absoluto era mayor que 1 no cumplía el Corolario 5 y la función no es contractil y no cumple las hipótesis del Teorema de convergencia

```
(%i44) wxplot2d([g(x),h(x)],[x,-4,4]);
```



(%t44)

(%o44)

Si aplicamos la definicón de iteración funcional vemos que hacemos abcisa y empieza a diverger

4 Comprueba que s=0 es un punto fijo de la función g(x)=x+x^2/2 tal que |g'(s)|=1.Dibujalas en [-0.05,0.05] Y deduce si con x^(0)=-0.2 y x^(0)=0.2 aproximan s. Elabora un programa que calcule las 20 primeras iteraciones

```
--> g(x):= x+x^2/2;
```

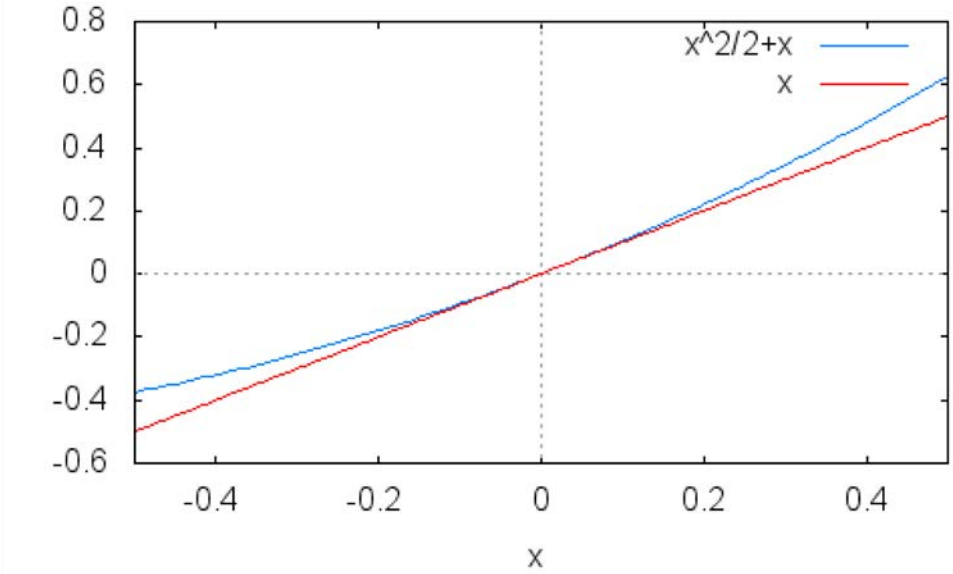
(%o45) $g(x):=x+\frac{x^2}{2}$

```
--> h(x):=x;
```

(%o46) $h(x):=x$

```
--> wxplot2d([g(x),h(x)],[x,-0.5,0.5]);
```

(%t47)



(%o47)

```
--> g(0);
```

(%o48) 0

Como podemos ver es un punto fijo por la grafica ambas funciones se cortan en el 0 y g(0)=0
En el -0.2 converge porque hacemos lo de voy a la grafica azul g(x) hago horizontal a la otra función y voelvo al eje. Al final nos acercamos a la solución pero lentamente. En cambio con 0.2 vemos que nos vamos para el otro lado

```
(%i50) x[0]:bfloat(-0.2)$
       niter:20$
       for k:1 thru niter do(
         x[k]:g(x[k-1]),
         print ("iteracion",k,"=",bfloat(x[k]))
       );
iteracion 1 = -1.8000000000000000888178419700125b-1
iteracion 2 = -1.6380000000000000728306304154103b-1
iteracion 3 = -1.5038478000000000609009731533661b-1
iteracion 4 = -1.3907698897217580517423937039112b-1
iteracion 5 = -1.2940578454139244991156844548302b-1
iteracion 6 = -1.2103285600500580733252830771 5b-1
iteracion 7 = -1.1370837988864157217447413587788b-1
iteracion 8 = -1.0724358206019175859566125481142b-1
iteracion 9 = -1.0149298911364121679517172828541b-1
iteracion 10 = -9.6342575694030369522575585736085b-2
iteracion 11 = -9.1701629748350383853730056264316b-2
iteracion 12 = -8.7497035299098613811553563871604b-2
iteracion 13 = -8.3669169706032759385702587897159b-2
iteracion 14 = -8.0168904726384304371882773223 76b-2
iteracion 15 = -7.6955378083870262543769444059018b-2
iteracion 16 = -7.3994312975854552783237314477026b-2
iteracion 17 = -7.1256733799470194061827930165685b-2
iteracion 18 = -6.8717972743585915082529776257297b-2
iteracion 19 = -6.6356892854591806713834110848721b-2
iteracion 20 = -6.4155274239933918126475767798008b-2
 (%o51) done
```

Converge porque se va estabilizando pero lentamente

```
(%i53) x[0]:bfloat(0.2)$
       niter:20$
       for k:1 thru niter do(
         x[k]:g(x[k-1]),
         print ("iteracion",k,"=",bfloat(x[k]))
       );
iteracion 1 = 2.2000000000000001332267629550188b-1
iteracion 2 = 2.4420000000000001625366508051229b-1
iteracion 3 = 2.7401682000000002022281009317 34b-1
iteracion 4 = 3.1155942882145622576420020636867b-1
iteracion 5 = 3.6009406766523224992772845096576b-1
iteracion 6 = 4.2492793644907868102503161811759b-1
iteracion 7 = 5.1520981203651480615543649881462b-1
iteracion 8 = 6.4793038724586526457066204568113b-1
iteracion 9 = 8.5783728060415372512401335794671b-1
iteracion 10 = 1.2257796806013185135687655014 02b0
iteracion 11 = 1.9770475932888537288899545130858b0
iteracion 12 = 3.9314061863534781228626318834085b0
iteracion 13 = 1.1659383487402677499762924289753b1
iteracion 14 = 7.9629995140561788475666284752935b1
iteracion 15 = 3.2500980581835088118628383510367b3
iteracion 16 = 5.2848187919622908238899988674391b6
iteracion 17 = 1.3964660116757675431754563436481b13
iteracion 18 = 9.7505866088295211271099825627795b25
iteracion 19 = 4.7536969608142790425395715344971b51
iteracion 20 = 1.1298817397627456609443597062801b103
(%o54) done
```

No converge como ya sabiamos anteriormente

Hörner

1 Se considera p(x)= 2x^4 -3x^2 + 3x -4
a) Haz un programa que utilice Hörner para halla
el valor en x=-2 y lo muestre en pantalla.
Comprueba que dicho valor es correcto.

```
(%i55) p(x):=2*x^4 -3*x^2+3*x-4;
(%o55) p(x):=2 x^4 - 3 x^2+3 x - 4
```

```
(%i) Horner_a(polinomio,valor):=block(
     [v,n],
     define(h(x),polinomio),
     n:hipow(expand(h(x)),x),
     b[n]:coeff(h(x),x,n),

     for k:n-1 thru 0 step -1 do (
         b[k]:coeff(h(x),x,k)+valor*b[k+1]),
     print("Valor del polinomio en",valor,"=",b[0])
     )$
```

```
(%i) Horner_a(p(x),-2);
Valor del polinomio en -2 = 10
(%o57) 10
```

Comprobamos el valor

```
(%i) p(-2);
(%o58) 10
```

1.1 Ahora calcule el valor de la primera derivada del polinomio en x=-2 y lo muestre tambien en pantalla. Comprueba que es correcto

```
(%i) Apartado_b(polinomio,valor):=block(
     [v,n],
     define(h(x),polinomio),
     n:hipow(expand(h(x)),x),
     b[n]:coeff(h(x),x,n),

     for k:n-1 thru 0 step -1 do (
         b[k]:coeff(h(x),x,k)+valor*b[k+1]),

     d[n]:b[n],

     for i:n-1 thru 1 step -1 do(
         d[i]:b[i]+valor*d[i+1]),

     print("Valor de la derivada en", valor,"=",d[1])
     )$
```

```
(%i) Apartado_b(p(x),-2);
Valor de la derivada en -2 = -49
(%o66) -49
```

Lo comprobamos

```
(%i) define(d1(x),diff(p(x),x));
(%o63) d1(x):=8 x^3-6 x+3
```

```
(%i64) d1(-2);
(%o64) -49
```

1.2 Dibuja la gráfica del polinomio en un intervalo [a,b] en el que se encuentren todas las soluciones. p(x)=0, justificando la respuesta. ¿Cuántas soluciones reales tiene? ¿Por qué? ¿Cuántas complejas tiene? ¿Por qué?

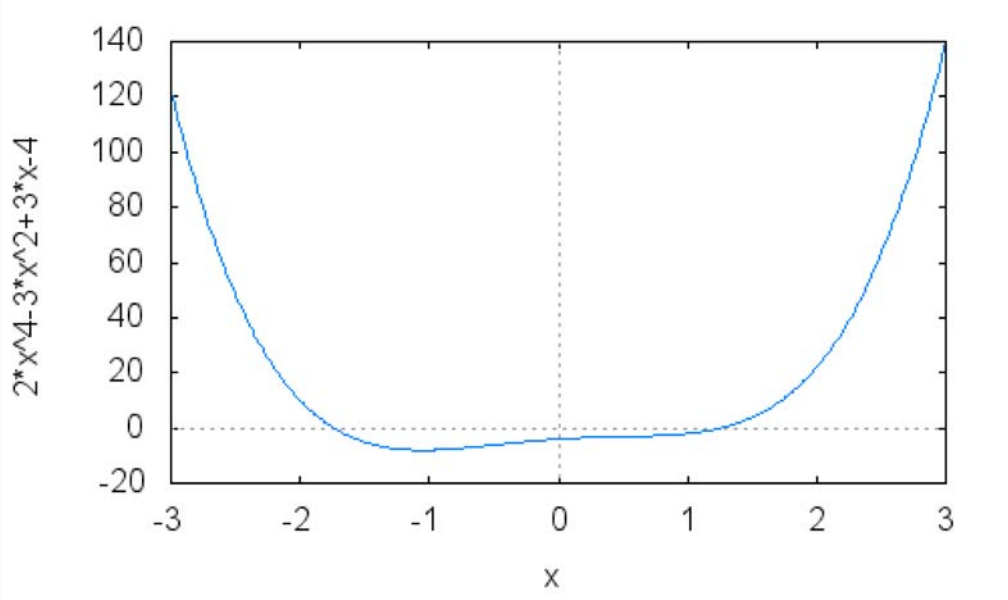
Tenemos un teorema que nos dice que el módulo de las soluciones de un polinomio es menor o igual que el máximo de los cocientes de los coeficientes del polinomio por el coeficiente lider mas 1.
Teorema 12: ||z|| <= lambda +1

lambda seria: 3/2, 4/2 claramente es 4/2=2

Por tanto ||z|| <= 3 , z solucion

El intervalo es el [-3,3]

```
(%i60) wxplot2d(p(x),[x,-3,3]);
```




```
(%o60)
```

El polinomio tiene 4 raices por que es de grado 4, y como vemos solo tiene dos reales por tanto las otras son complejas
2 reales y dos complejas. Además sabemos que los ceros complejos de un polinomio se presentan en número par.

1.3 Elabora un programa que calcule las 5 primeras iteraciones de Newton-Raphson con x(0)=-2, usa Hórner para calcular las iteraciones .Debe mostrar todas las iteraciones con 32 de precision

```
(%i70) fpprec:32$

(%i71) Raphson_d(polinomio,valor,niter):=block(
           [v,n,it],

           define(h(x),polinomio),

           n:hipow(expand(h(x)),x),
           b[n]:coeff(h(x),x,n),
           x0:valor,

           for j:1 thru niter step +1 do(
               for k:n-1 thru 0 step -1 do(
               b[k]:coeff(h(x),x,k)+x0*b[k+1]),

               define(f(x),b[0]),
               d[n]:b[n],

               for i:n-1 thru 1 step -1 do(
                   d[i]:b[i]+x0*d[i+1]),

               define(df(x),d[1]),

               x1:x0-f(x)/df(x),
               print("Iteracion",j,"=", bfloat(x1)),
            x0:x1
           )
       )$

(%i72) Raphson_d(p(x),-2,5);
Iteracion 1 = -1.795918367346938775510204081632 7b0
Iteracion 2 = -1.742432916750542041921353359128 4b0
Iteracion 3 = -1.738970235336166731984712803172 6b0
Iteracion 4 = -1.738956256679049247280396967453 6b0
Iteracion 5 = -1.738956256451891899033412342117 7b0
(%o71) done
```

1.4 Calcular iteraciones hasta un máximo establecido, cuando la diferencia en valor absoluto < que tol se detenga e informe "Se ha alcanzado la tolerancia Ademas numero de iteraciones y 32 digitos de las iteraciones calculadas.Nº max= 30 y tol 10^(-15) ¿Cuántas iteraciones han calculado?¿Qué proporciona la última iteración calculada?¿Por qué?

```
--> Raphson_e(polinomio,valor,niter,tol):=block(
        [v,n,it],

        define(h(x),polinomio),

        n:hipow(expand(h(x)),x),
        b[n]:coeff(h(x),x,n),
        x0:valor,

        for j:1 thru niter step +1 do(
            for k:n-1 thru 0 step -1 do(
            b[k]:coeff(h(x),x,k)+x0*b[k+1]),

            define(f(x),b[0]),
            d[n]:b[n],

            for i:n-1 thru 1 step -1 do(
                d[i]:b[i]+x0*d[i+1]),

            define(df(x),d[1]),

            x1:x0-f(x)/df(x),
            print("Iteracion",j,"=", bfloat(x1)),

            if(abs(x0-x1))< tol then
                return(false),
         x0:x1
        )
    )$
```

```
--> Raphson_e(p(x),-2,30,10^(-15));
Iteracion 1 = -1.795918367346938775510204081632 7b0
Iteracion 2 = -1.742432916750542041921353359128 4b0
Iteracion 3 = -1.738970235336166731984712803172 6b0
Iteracion 4 = -1.738956256679049247280396967453 6b0
Iteracion 5 = -1.738956256451891899033412342117 7b0
Iteracion 6 = -1.738956256451891898973427103301 1b0
(%o75) false
```

La última iteración es una aproximación a la solución p(x)=0
con un error menor que 10^(-15). Además podemos asegurar que es una aproximación de la solución por el Teorema Global de Newton-Raphson. (Comprobar las 4 condiciones)
f(a)*f(b)<0
f'(x) =!0
f''(x) no cambia de signo en [a,b]
max {|f(a)/f'(a)|,|f(b)/f'(b)|} <= b-a