Continuacion de la practica 1

1 Se quiere aproximar la única solución real s de la ecuación x^2-3=0 en el intervalo [1, 2]. Para ello se consideran, sobre dicho intervalo, las siguientes funciones:

```
(%i58) g0(x):=x^2-3;g1(x):=x^2+x-3;g2(x):=3/x;g3(x):=(x+3/x)/2;
```

(%o58) $g0(x):=x^2-3$

(%o59) $g1(x):=x^2+x-3$

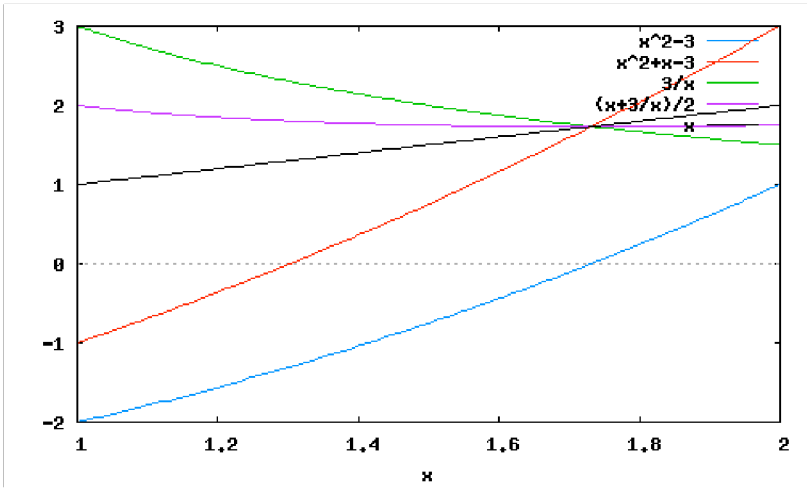
(%o60) $g2(x):=\frac{3}{x}$

(%o61) $g3(x):=\frac{x+\frac{3}{x}}{2}$

a) Justifica basandote en la representacion grafica, por que para cada i=1,2,3 gi(x) tiene a s como unico punto fijo en [1,

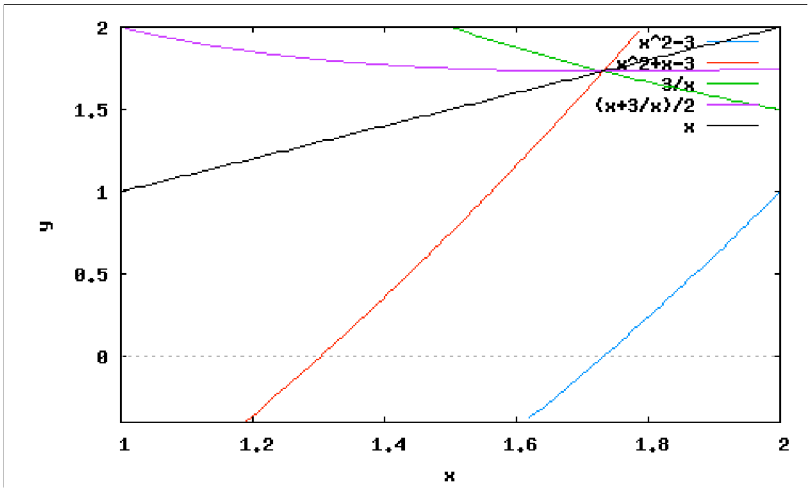
```
(%i62) wxplot2d([g0(x),g1(x),g2(x),g3(x),x], [x,1,2])$
```

(%t62)




```
(%i63) wxplot2d([g0(x),g1(x),g2(x),g3(x),x], [x,1,2], [y,-0.4,2])$
plot2d: some values were clipped.
plot2d: some values were clipped.
plot2d: some values were clipped.
```

(%t63)



Como observamos en la grafica en el punto s=sqrt(3), solucion de g0, es el unico punto fijo de las gi, en [1,2]

b)Para cada i = 1, 2, 3 , elabora un programa que calcule las 10 primeras
iteraciones del método de iteración funcional
asociado a gi que considera
x
(0) = 2 . El programa debe mostrar en pantalla todas las iteraciones
calculadas con 32 dígitos
 significativos. ¿Para cuál/es de las funciones gi
consideradas el método de iteración funcional asociado permite aproximar
s? Justifica la respuesta y da un valor aproximado para s.

```
(%i64) gi(n):=block([x],
       x:2,
            fpprec:32,
       if n=1 then
           (
            for i:1 thru 10 do (
          x:g1(x),
            print("iteracion",i," x:",bfloat(x))
            )
           )
       else if n=2 then(
            for i:1 thru 10 do (
                x:g2(x),
            print("iteracion",i," x:",bfloat(x))
            )
           )
       else if n=3 then(
            for i:1 thru 10 do (
            x:g3(x),
            print("iteracion",i," x:",bfloat(x))
            )
       )
       )$
```

```
(%i65) gi(1);
iteracion 1  x: 3.0b0
iteracion 2  x: 9.0b0
iteracion 3  x: 8.7b1
iteracion 4  x: 7.653b3
iteracion 5  x: 5.8576059b7
iteracion 6  x: 3.431154746547537b15
iteracion 7  x: 1.17728228947556962997757473139O3b31
iteracion 8  x: 1.385993589112838926343930869836 6b62
iteracion 9  x: 1.92097822906188897801739541260 39b124
iteracion 10  x: 3.690157356529751199776782587393b248
(%o65) done
```

```
(%i66) gi(2);
iteracion 1  x: 1.5b0
iteracion 2  x: 2.0b0
iteracion 3  x: 1.5b0
iteracion 4  x: 2.0b0
iteracion 5  x: 1.5b0
iteracion 6  x: 2.0b0
iteracion 7  x: 1.5b0
iteracion 8  x: 2.0b0
iteracion 9  x: 1.5b0
iteracion 10  x: 2.0b0
(%o66) done
```

```
(%i67) gi(3);
iteracion 1  x: 1.75b0
iteracion 2  x: 1.7321428571428571428571428571429b0
iteracion 3  x: 1.7320508100147275405007363770250b0
iteracion 4  x: 1.7320508075688772952543539460722b0
iteracion 5  x: 1.7320508075688772935274463415059b0
iteracion 6  x: 1.7320508075688772935274463415059b0
iteracion 7  x: 1.7320508075688772935274463415059b0
iteracion 8  x: 1.7320508075688772935274463415059b0
iteracion 9  x: 1.7320508075688772935274463415059b0
iteracion 10  x: 1.7320508075688772935274463415059b0
(%o67) done
```

Claramante la que mejor se aproxima es la tercera, ya que desde la quinta iteracion da un valor muy proximo a s.

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

a) Razona por qué tiene una única solución s en el intervalo [0, π/2].

Este problema es equivalente a buscar los puntos fijos de b(x)=cos(x)/2, como vemos en la grafica solamente tiene 1 en
Y comprobamos que cumple las condiciones de convergencia segun el teorema 5
-b(x) Continua en el intervalo [0, π/2].
-La imagen de la funcion,b([0, π/2]) está contenida en el intervalo [0, π/2].
-|b´(x)|≤l≤0.5<1 luego es contractil
Ya podemos asegurar la convergencia del metodo de iteración funcional y la unicidad del punto fijo

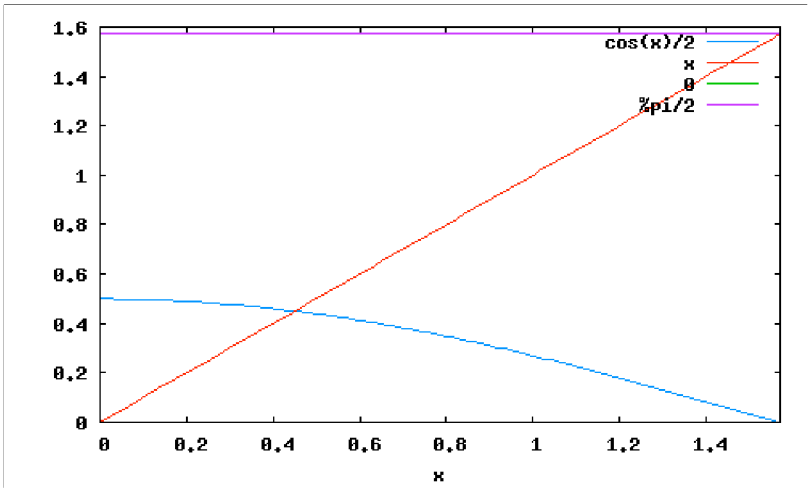
```
(%i68) b(x):=cos(x)/2;define(db(x),diff(b(x),x,1));
```

$$(\%o68)\quad b(x):=\frac{\cos(x)}{2}$$

$$(\%o69)\quad db(x):=-\frac{\sin(x)}{2}$$

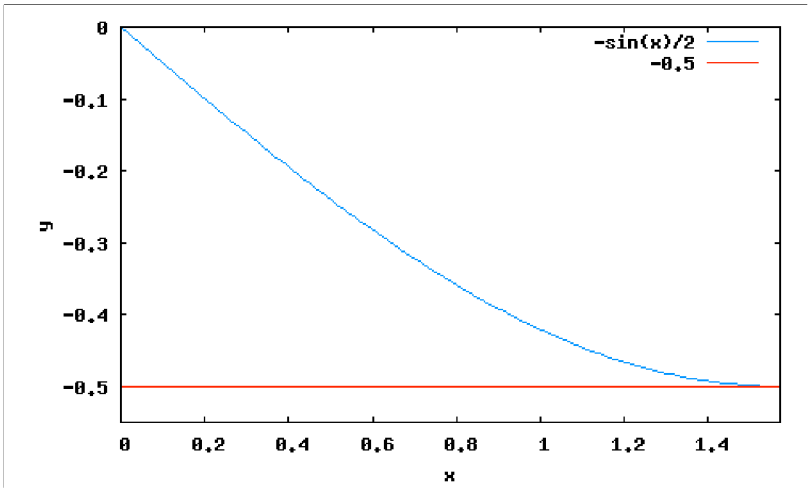
```
(%i70) wxplot2d([b(x),x,0,%pi/2], [x,0,%pi/2])$
```

(%t70)



```
(%i71) wxplot2d([db(x),-0.5], [x,0,%pi/2],[y,-0.55,0])$
```

(%t71)



b) Sin calcular iteraciones, describe un método de iteración funcional (distinto al de Newton-Raphson) que,considerando x^(0) = 1 como aproximación inicial, permita aproximar s. Razona la respuesta. ¿Qué orden de convergencia tiene la sucesión de iteraciones generada por dicho método?

Podemos usar el metodo funcional que tiene por función a b(x), y como aproximacion inicial x^0=1
La convergencia es lineal como consecuencia del teorema de convergencia global (T°9)

c) Elabora un programa que calcule las 12 primeras iteraciones del método de iteración funcional descrito en el apartado anterior. El programa sólo debe mostrar en pantalla, con 32 dígitos significativos, el valor aproximado de s proporcionado por la última de las iteraciones calculadas.

```
(%i72) pb():=block([x],
       fpprec:32,
       x:1.0b0,
       for i:1 thru 12 do(
           x:b(x)
           ),
       print ("solucion hallada x: ",x)
       )$
```

```
(%i73) pb();
solucion hallada x:  4.5018361902159086388302806542872b-1
(%o73) 4.5018361902159086388302806542872b-1
```

d) Haz un programa que calcule las iteraciones del método descrito en b) 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: “Se ha alcanzado la tolerancia”. El programa además debe mostrar en pantalla, con 32 dígitos significativos, todas las iteraciones calculadas. Ejecuta el programa para un número máximo de 50 iteraciones y tol=10−12: ¿Cuántas iteraciones se realizan?

```
(%i74) pbcontol(n,tol):=block([x],
           x:1.0b0,
           for i:1 thru n do(
               x:b(x),
               print("iteracion ",i," x: ",x),
               if(abs(x-b(x))<tol) then(
                   print("Se ha alcanzado la tolerancia en la iteracion ",i," x: ",x),
                   i:n
                   )
           )

       )$
```

```
(%i75) pbcontol(50,10^-12)$
iteracion 1 x: 2.701511529340698587004683037214 9b-1
iteracion 2 x: 4.818652840585210460654274600768 5b-1
iteracion 3 x: 4.430660154453104650885854300724b-1
iteracion 4 x: 4.517207378769961625191256396993 2b-1
iteracion 5 x: 4.498486539841154921002592097630 4b-1
iteracion 6 x: 4.502564611702062128785793865475 8b-1
iteracion 7 x: 4.501677604858847906967750440538 9b-1
iteracion 8 x: 4.501870598263853876984765805696 3b-1
iteracion 9 x: 4.501828610109571240415509579532 7b-1
iteracion 10 x: 4.501837745305894059404705969922b-1
iteracion 11 x: 4.501835757804189473413531758937 6b-1
iteracion 12 x: 4.501836190215908638830280654287 2b-1
iteracion 13 x: 4.501836096138069934481570194917 5b-1
iteracion 14 x: 4.501836116606155448375303789179 2b-1
iteracion 15 x: 4.501836112153007638186560599240 1b-1
iteracion 16 x: 4.501836113121858654505841855618 1b-1
iteracion 17 x: 4.501836112911070167014620722330 9b-1
iteracion 18 x: 4.501836112956930454833815292481 1b-1
Se ha alcanzado la tolerancia en la iteracion 18 x: 4.501836112956930454833815292481 1b-1
```

3 Comprueba que s = −2 es un punto fijo de la función g(x) = 1 +x−x^2/4 tal que |g′(s)| > 1. Elabora un programa que calcule las 15 primeras iteraciones del método de iteración funcional asociado a g que considera x^(0) = −2.05 como aproximación inicial.El programa debe mostrar en pantalla, con 32 digitos s todas las iteraciones calculadas. ¿Permite dicho método aproximar s?¿Por qué? ¿Era de esperar la respuesta anterior?¿Por qué? Dibuja juntas en el intervalo [−4, 4] las gráficas de g(x) y de y(x) = x y recue la interpretación geométrica del cálculo de las iteraciones en los métodos de it

```
(%i76) c(x):=1+x-(x^2/4)$  define(dc(x),diff(c(x),x,1))$
```

Comprobamos que -2 es punto fijo y que la derivada en dicho punto es mayor a 1

```
(%i78) is(c(-2)=-2);  is(abs(dc(-2))>1);
(%o78) true
(%o79) true
```

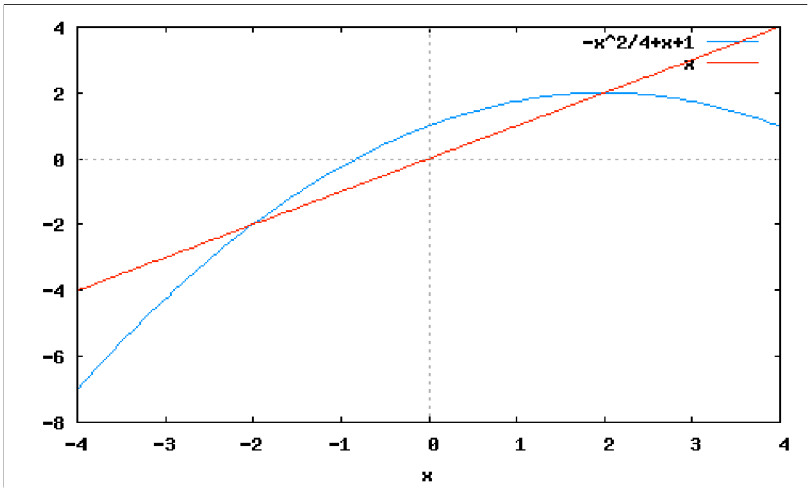
```
(%i80) pc():=block([x],
       x:-2.05,
       fpprec:32,
       for i:1 thru 15 do (
           x:c(x),
           print("iteracion: ",i," x:",bfloat(x))
           )
       )$
```

```
(%i81) pc()$
iteracion:  1  x: -2.10062499999999996447286321199 5b0
iteracion:  2  x: -2.20378134765624977831066644284 87b0
iteracion:  3  x: -2.41794440472564886590589594561 61b0
iteracion:  4  x: -2.87955819081166719541897691669 87b0
iteracion:  5  x: -3.95252203437930749174711309024 13b0
iteracion:  6  x: -6.85812964244279221759370557265 36b0
iteracion:  7  x: -1.76166151905809158506599487736 82b1
iteracion:  8  x: -9.42028978838324917433055816218 26b1
iteracion:  9  x: -2.31174939031177564174868166446 69b3
iteracion:  10  x: -1.33835706029202835634350776672 36b6
iteracion:  11  x: -4.47801243564440307617187 5b11
iteracion:  12  x: -5.01314884349126026199 04b22
iteracion:  13  x: -6.28291533174943992728480663976 79b44
iteracion:  14  x: -9.86875626648304379570962374445 88b88
iteracion:  15  x: -2.43480875618120853289568411700 81b177
```

Como observamos el metodo diverge ,ademas |c´(-2)| es mayor que 1 luego era de esperar este resultado segun la teoria, y viendo la grafica observamos que segun la interpretacion geometrica, la sucesion que obtenemos es repulsiva ya que la funcion se queda por debajo de la identidad, estando el valo inicial a la izquierda de la solucion, luego la sucesion diverge negativamente como tambien hemos comprobado empiricamente.

```
(%i82) wxplot2d([c(x),x], [x,-4,4])$
```

(%t82)



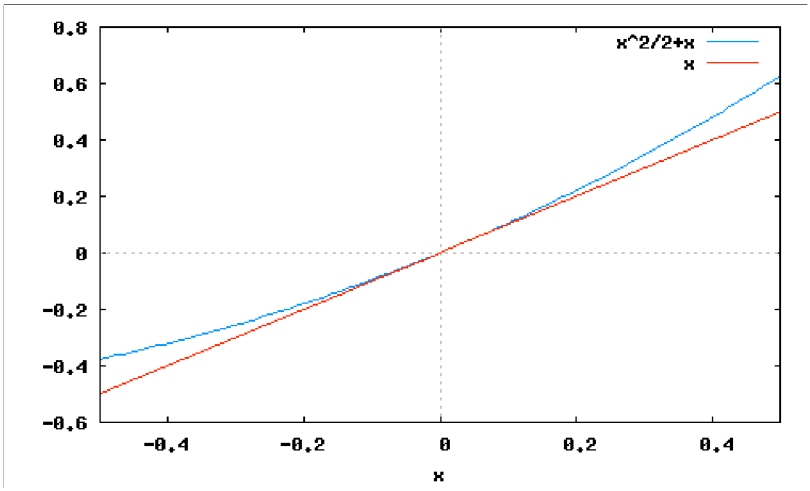
4 Comprueba que s = 0 es un punto fijo de la función g(x) = x+x^2/2 tal que |g'(s)|=1.Dibuja juntas en el intervalo [-0.5,0.5] las gráficas de g(x) de y(x) = x, y deduce si los métodos de iteración funcional asociados a g que consideran x^(0) = -0.2 y x^(0) = 0.2 , permiten aproximar s. Elabora un programa que calcule y muestre en pantalla, con 32 dígitos significativos, las 20 primeras iteraciones de cada uno de estos dos métodos.

```
(%i83) d(x):=x+x^2/2$ define(dd(x),diff(d(x),x,1))$
```

```
(%i85) is(abs(dd(0))=1);
(%o85) true
```

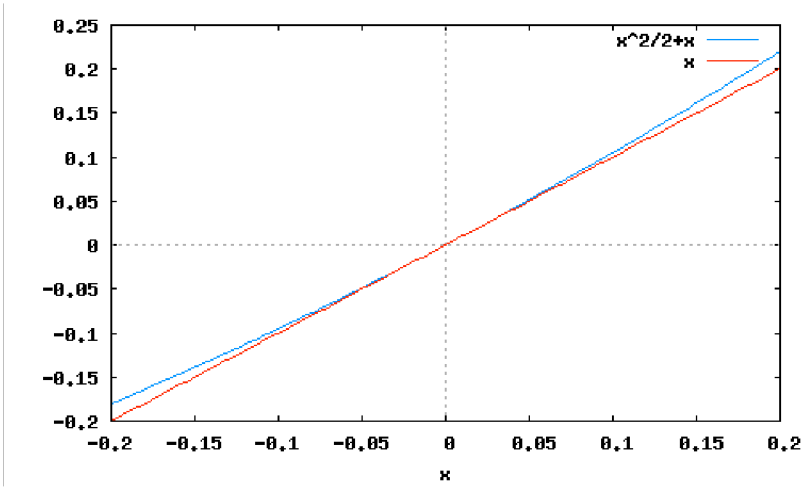
```
(%i86) wxplot2d([d(x),x], [x,-0.5,0.5])$
```

(%t86)




```
(%i87) wxplot2d([d(x),x], [x,-0.2,0.2])$
```

(%t87)



Observando la grafica vemos que el punto fijo en el intervalo es solamente x=0, ademas comprobado que su derivada es igual
teoricamente no podemos asegurar convergencia para las iteraciones funcionales, solo nos queda dos soluciones:
1° Geometricamente podemos ver en la segunda grafica que la sucesion convergue a 0 si el x^0=-0.2 pero diverge si x^0=0
2° Empiricamente mediante el programa , que como vemos nos demuestra lo visto en la solucion geometrica

```
(%i88) pd(x):=block([y],
       fpprec:32,
       y:x,
       for i:1 thru 20 do (
           y:d(y),
           print("iteracion: ",i," y:",bfloat(y))
           )
       )$
```

```
(%i89) pd(0.2);
iteracion:  1  y: 2.2000000000000002886579864025407b-1
iteracion:  2  y: 2.4420000000000002815525590449397b-1
iteracion:  3  y: 2.7401682000000004979511913916213b-1
iteracion:  4  y: 3.1155942882145626349199574178783b-1
iteracion:  5  y: 3.6009406766523227583221000071382b-1
iteracion:  6  y: 4.2492793644907872474547616548088-1
iteracion:  7  y: 5.1520981203651483593120019577327b-1
iteracion:  8  y: 6.4793038724586526910798056633212b-1
iteracion:  9  y: 8.5783728060415376237557438798831b-1
iteracion:  10  y: 1.2257796806013185886286009917967b0
iteracion:  11  y: 1.9770475932888538217468976654345b0
iteracion:  12  y: 3.9314061863534783825002705270890b0
iteracion:  13  y: 1.1659383487402678269972966518253b1
iteracion:  14  y: 7.9629995140561803168566257227212b1
iteracion:  15  y: 3.2500980581835101475007832050323b3
iteracion:  16  y: 5.2848187919622957706451416015625b6
iteracion:  17  y: 1.3964660116757701171875b13
iteracion:  18  y: 9.7505866088295566510391296b25
iteracion:  19  y: 4.7536969608143139926222879116928b51
iteracion:  20  y: 1.1298817397627622360368074146223b103
(%o89) done
```

```
(%i90) pd(-0.2);
iteracion:  1  y: -1.799999999999999933386618522490 6b-1
iteracion:  2  y: -1.6380000000000000011546319456101 6b-1
iteracion:  3  y: -1.5038477999999999568103703495581b-1
iteracion:  4  y: -1.3907698897217579681040433570161b-1
iteracion:  5  y: -1.2940578454139245456744333750976b-1
iteracion:  6  y: -1.2103285600500580942995298983078b-1
iteracion:  7  y: -1.1370837988864157530599641177105b-1
iteracion:  8  y: -1.0724358206019175832057754860216b-1
iteracion:  9  y: -1.0149298911364121322797871016519b-1
iteracion:  10  y: -9.6342575694030360700104154147994b-2
iteracion:  11  y: -9.1701629748350374504717308354884b-2
iteracion:  12  y: -8.7497035299098607774581637386291b-2
iteracion:  13  y: -8.3669169706032747724222531360283b-2
iteracion:  14  y: -8.0168904726384299097574626102869b-2
iteracion:  15  y: -7.6955378083870262795329608707107b-2
iteracion:  16  y: -7.3994312975854556468924272394361b-2
iteracion:  17  y: -7.1256733799470201029535587622377b-2
iteracion:  18  y: -6.8717972743585917227981951782567b-2
iteracion:  19  y: -6.6356892854591814634446222953557b-2
iteracion:  20  y: -6.4155274239933932078905343132647b-2
(%o90) done
```

5 Se considera el polinomio p(x)=2x^4-3x^2+3x-4

```
(%i91) p(x):=2*x^4-3*x^2+3*x-4;define(dp(x),diff(p(x),x,1));
(%o91) p(x):=2 x^4-3 x^2+3 x-4
(%o92) dp(x):=8 x^3-6 x+3
```

5.1 Haz un programa que utilice el algoritmo de Hörner para hallar el valor del polinomio en x = −2 y lo muestre en pantalla. Comprueba que dicho valor es correcto.

```
(%i93) horner(xi):=block([res,n],
       res:coeff(p(x),x,4),
       n:4,
       for i:1 thru n do(
           res:coeff(p(x),x,4-i)+res*xi
           ),
       return (res)
       );
```

(%o93) horner(ξ):=
block([res, n], res: coeff(p(x), x, 4), n: 4, for i thru n do res: coeff(p(x), x, 4 − i) + res ξ, return(res))

```
(%i94) is(horner(-2)=p(-2));
```

(%o94) true

5.2 Modifica el programa del apartado anterior para que éste calcule, además, el valor de la primera derivada del polinomio en x = −2 y lo muestre también en pantalla. Comprueba que dicho valor es correcto.

```
(%i95) hornerdef(xi):=block([res,n,L],
       n:4,
       L:append(L:[],[res:coeff(p(x),x,n)]),
       for i:1 thru n do(
           res:coeff(p(x),x,n-i)+res*xi,
           L:append(L,[res])
           ),
       res:L[1],           /*L es una lista con los coeficientes bn, de forma que l[1]=bn,..L[n]=b1, L[n+1]=b0 */
       for i:2 thru n do(    /*Por ello vamos desde i:2 hasta n*/
           res:L[i]+res*xi
           ),
       return (res)
       );
```

(%o95) hornerdef(ξ):=block([res, n, L], n: 4, L: append(L: [], [res: coeff(p(x), x, n)]), for i thru n do (res: coeff(p(x), x, n − i) + res ξ, L: append(L, [res])), res: L_1, for i from 2 thru n do res: L_i + res ξ, return(res))

```
(%i96) is(hornerdef(-2)=dp(-2));
```

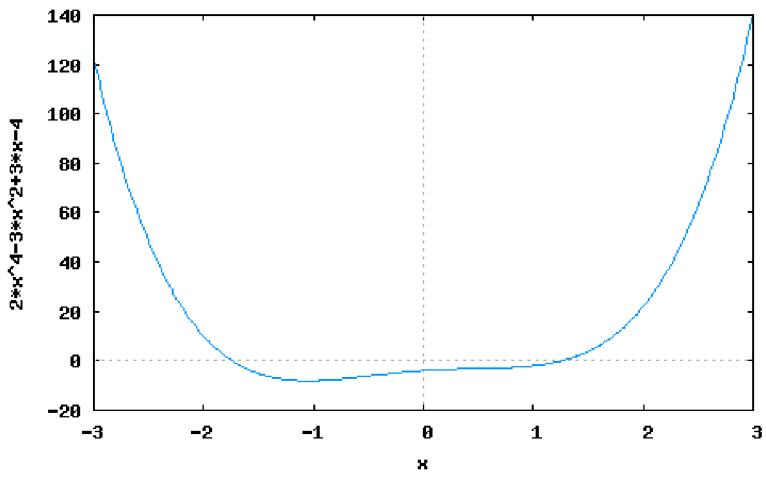
(%o96) true

5.3 Dibuja la gráfica del polinomio en un intervalo [a, b] en el que se encuentren todas las soluciones reales de la ecuación p(x) = 0 , justificando la elección del intervalo considerado. ¿Cuántas soluciones reales tiene dicha ecuación? ¿Por qué? ¿Y cuántas complejas? ¿Por qué?

```
Por el teorema 12, sabemos que toda solucion ,z,  del polinomio p(x)=2x^4-3x^2+3x-4, cumple que
para h=max{|ai/an|, para i=1,..n}, |z|<h+1, para este caso concreto |z|<3, luego podemos dibujar la grafica
en el intervalo de x perteneciente a [-3,3] en el cual visualizaremos todas las soluciones.
Como podemos observar contando el numero de cortes del eje de la x, hay 2 soluciones reales, luego como todo polinomio
de grado n tiene exactamente n soluciones en C, y el numero de soluciones complejas siempre es par, podemos afirmar que exi
2 soluciones reales y 2 complejas, todas ellas contenidas en el intervalo [-3,3]
```

```
(%i97) wxplot2d([p(x)], [x,-3,3])$
```

(%t97)



5.4 Elabora un programa que calcule las 5 primeras iteraciones del método de Newton-Raphson que considera x^(0) = −2 como aproximación inicial, usando el algoritmo de Hörner para calcular cada una de las iteraciones de dicho método. El programa debe mostrar en pantalla, con 32 dígitos significativos, todas las iteraciones calculadas.

```
(%i98) (horner(-2)/hornerdef(-2));
```

$$(\%o98)\ -\frac{10}{49}$$

```
(%i99) NewtonRapson(x0):=block(
       fprrec:32,
       for i:1 thru 5 do(
           x0:x0-(horner(x0)/hornerdef(x0)),
           print("x",i,": ",bfloat(x0))
       ),
       print("resultado : ",bfloat(x0))
       );
```

$$(\%o99)\ \mathrm{NewtonRapson}(x0):=\mathrm{block}\left(fprrec:32,\ \mathrm{for}\ \ i\ \ \mathrm{thru}\ \ 5\ \ \mathrm{do}\ \left(x0:x0-\frac{\mathrm{horner}(x0)}{\mathrm{hornerdef}(x0)},\ \mathrm{print}(x,i,:\ ,\mathrm{bfloat}(x0))\right),\ \mathrm{print}(resultado\ :\ ,\mathrm{bfloat}(x0))\right)$$

```
(%i100) NewtonRapson(-2)$
x 1 :  -1.795918367346938775510204081632 7b0
x 2 :  -1.742432916750542041921353359128 4b0
x 3 :  -1.738970235336166731984712803172 6b0
x 4 :  -1.738956256679049247280396967453 6b0
x 5 :  -1.738956256451891899033412342117 7b0
resultado :  -1.738956256451891899033412342117 7b0
```

5.5 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 consecutiv<as sea menor que una cierta tolerancia (tol) se detenga el cálculo de las iteraciones, e informe de ello mediante el siguiente mensaje:
"Se ha alcanzado la tolerancia".
El programa además debe mostrar en pantalla el número de iteraciones que ha calculado y el valor con 32 dígitos significativos de (únicamente) la última de las iteraciones calculadas.
Ejecuta el programa para un número máximo de 30 iteraciones y tol=10-15:
¿Cuántas iteraciones se han calculado?
¿Qué proporciona la última de las iteraciones calculada?¿Por qué?

```
(%i101) NewtonRapsonTol(x0,n,tol):=block([xn,j],
        fprrec:32,
        for i:1 thru n do(
            j:i,
            xn:x0-(horner(x0)/hornerdef(x0)),
            print("x",i,": ",bfloat(xn)),
             if(abs(xn-x0)<tol) then(i:n,print("Se a alcanzado la tolerancia")),
            x0:xn
        ),
        print("resultado : ",bfloat(xn)," En la iteracion ",j)
        );
```

(%o101) NewtonRapsonTol(x0, n, tol):=block([xn, j], fprrec:32, for i thru n do (j:i, xn:x0−horner(x0)/hornerdef(x0), print(x, i, : , bfloat(xn)), if |xn−x0|<tol then (i:n, print(Se a alcanzado la tolerancia)), x0:xn), print(resultado : , bfloat(xn), En la iteracion , j))

```
(%i102) NewtonRapsonTol(-2,30,10^-15)$
```

x 1 : −1.795918367346938775510204081632 7b0
x 2 : −1.742432916750542041921353359128 4b0
x 3 : −1.738970235336166731984712803172 6b0
x 4 : −1.738956256679049247280396967453 6b0
x 5 : −1.738956256451891899033412342117 7b0
x 6 : −1.738956256451891898973427103301 1b0
Se a alcanzado la tolerancia
resultado : −1.738956256451891898973427103301 1b0 En la iteracion 6