

## 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:

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

```
(%o104) g0(x):=x^2-3
```

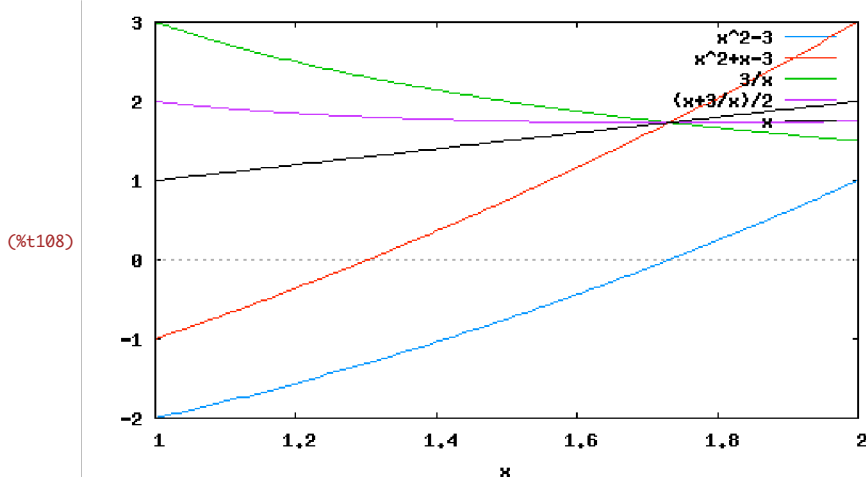
```
(%o105) g1(x):=x^2+x-3
```

```
(%o106) g2(x):=3/x
```

```
(%o107) g3(x):=(x+3/x)/2
```

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

```
--> wxplot2d([g0(x),g1(x),g2(x),g3(x),x], [x,1,2])$
```

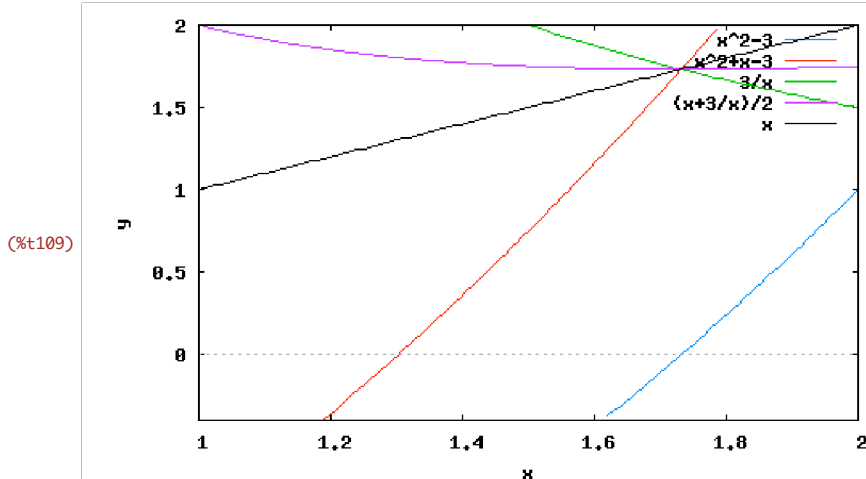


```
--> wxplot2d([g0(x),g1(x),g2(x),g3(x),x], [x,1,2], [y,-0.4,2])$
```

plot2d: some values were clipped.

plot2d: some values were clipped.

plot2d: some values were clipped.



Como observamos en la grafica en el punto  $s=\sqrt{3}$ , solucion de  $g_0$ , es el unico punto fijo de las  $g_i$ , 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  $g_i$  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  $g_i$  consideradas el método de iteración funcional asociado permite aproximar  $s$ ? Justifica la respuesta y da un valor aproximado para  $s$ .

```
--> gi(n):=block([x],
  x:2,
  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),
      fpprec:32,
      print("iteracion",i," x:",bfloat(x))
    )
  )
)$
```

```
--> 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: 1.1772822894755696299775747313903b31
iteracion 8 x: 1.3859935891128389263439308698366b62
iteracion 9 x: 1.9209782290618889780173954126039b124
iteracion 10 x: 3.690157356529751199776782587393b248
(%o111) done
```

```
--> 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
(%o112) done
```

```

--> gi(3);
iteracion 1 x: 1.75b0
iteracion 2 x: 1.7321428571428571428571428571429b0
iteracion 3 x: 1.732050810014727540500736377025b0
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
(%o113) done

```

Claramente 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  $[0, \pi/2]$ .

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  $[0, \pi/2]$ .

-La imagen de la funcion,  $b([0, \pi/2])$  está contenida en el intervalo  $[0, \pi/2]$ .

- $|b'(x)| \leq 0.5 < 1$  luego es contractil

Ya podemos asegurar la convergencia del metodo de iteración funcional y la unicidad del punto fijo

```

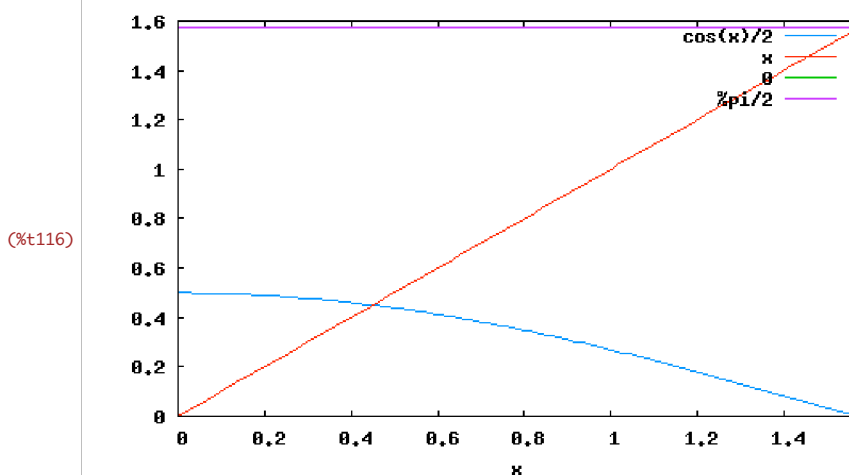
--> b(x):=cos(x)/2;define(db(x),diff(b(x),x,1));
(%o114) b(x):=cos(x)/2
(%o115) db(x):=-sin(x)/2

```

```

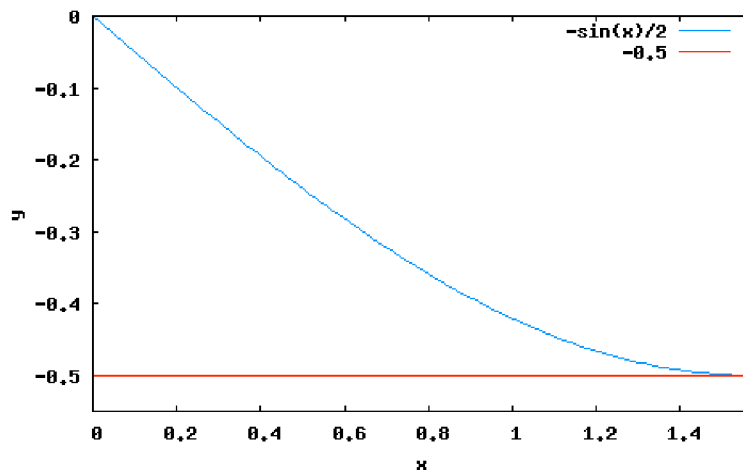
--> wxplot2d([b(x),x,0,%pi/2], [x,0,%pi/2])$

```



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

(%t117)



- 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 (Tº9)

- 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.

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

```
--> pb();
solucion hallada x: 4.5018361902159086388302806542872b-1
(%o119) 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  $\text{tol}=10^{-12}$ : ¿Cuántas iteraciones se realizan?

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

```
--> pbcontol(50,10^-12)$
```

```
iteracion 1 x: 2.7015115293406985870046830372149b-1
iteracion 2 x: 4.8186528405852104606542746007685b-1
iteracion 3 x: 4.430660154453104650885854300724b-1
iteracion 4 x: 4.5172073787699616251912563969932b-1
iteracion 5 x: 4.4984865398411549210025920976304b-1
iteracion 6 x: 4.5025646117020621287857938654758b-1
iteracion 7 x: 4.5016776048588479069677504405389b-1
iteracion 8 x: 4.5018705982638538769847658056963b-1
iteracion 9 x: 4.5018286101095712404155095795327b-1
iteracion 10 x: 4.501837745305894059404705969922b-1
iteracion 11 x: 4.5018357578041894734135317589376b-1
iteracion 12 x: 4.5018361902159086388302806542872b-1
iteracion 13 x: 4.5018360961380699344815701949175b-1
iteracion 14 x: 4.5018361166061554483753037891792b-1
iteracion 15 x: 4.50183611215300763818656059924b-1
iteracion 16 x: 4.5018361131218586545058418556181b-1
iteracion 17 x: 4.5018361129110701670146207223309b-1
iteracion 18 x: 4.5018361129569304548338152924811b-1
Se ha alcanzado la tolerancia en la iteracion 19 x: 4.5018361129569304548338152924811b-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 dígitos 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

```
--> 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

```
--> is(c(-2)=-2); is(abs(dc(-2))>1);
(%o77) true
(%o78) true
```

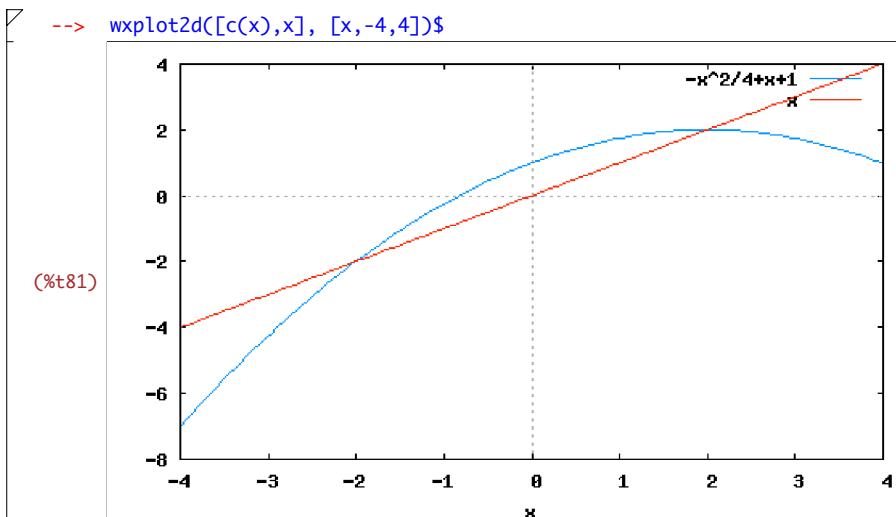
```
--> pc():=block([x],
  x:-2.05,
  fpprec:32,
  for i:1 thru 15 do (
    x:c(x),
    print("iteracion: ",i," x:",bfloat(x))
  )
)$
```

```

--> pc()$
iteracion: 1 x: -2.10062499999999964472863211995b0
iteracion: 2 x: -2.2037813476562497783106664428487b0
iteracion: 3 x: -2.4179444047256488659058959456161b0
iteracion: 4 x: -2.8795581908116671954189769166987b0
iteracion: 5 x: -3.9525220343793074917471130902413b0
iteracion: 6 x: -6.8581296424427922175937055726536b0
iteracion: 7 x: -1.7616615190580915850659948773682b1
iteracion: 8 x: -9.4202897883832491743305581621826b1
iteracion: 9 x: -2.3117493903117756417486816644669b3
iteracion: 10 x: -1.3383570602920283563435077667236b6
iteracion: 11 x: -4.478012435644403076171875b11
iteracion: 12 x: -5.0131488434912602619904b22
iteracion: 13 x: -6.2829153317494399272848066397679b44
iteracion: 14 x: -9.8687562664830437957096237444588b88
iteracion: 15 x: -2.4348087561812085328956841170081b177

```

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.



- 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  $x^{\wedge}(0) = -0.2$  y  $x^{\wedge}(0) = 0.2$ , 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.

```

--> d(x):=x+x^2/2$ define(dd(x),diff(d(x),x,1))$

```

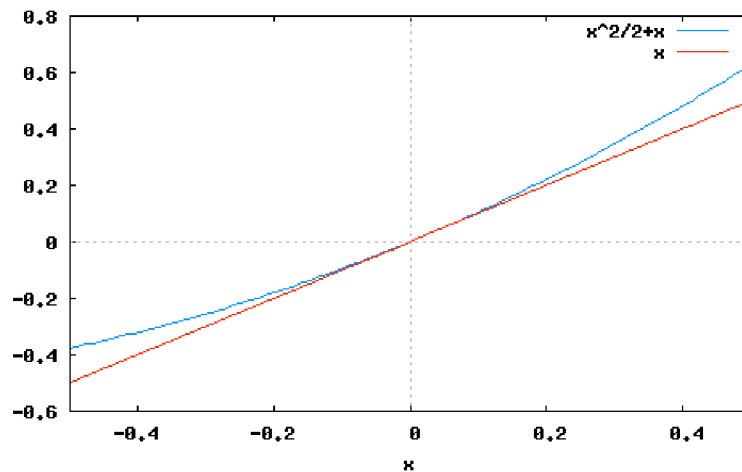
```

--> is(abs(dd(0))=1);
(%o67) true

```

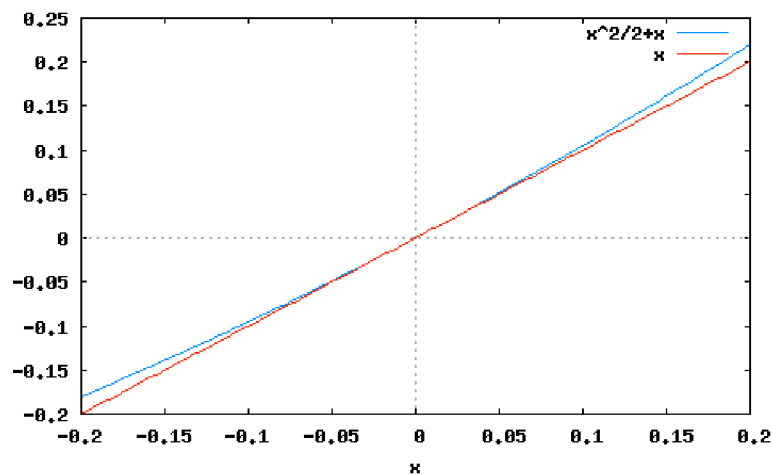
```
--> wxplot2d([d(x),x], [x,-0.5,0.5])$
```

(%t68)



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

(%t69)



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 converge 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

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

```
--> pd(0.2);
```

```

--> pd(-0.2);
iteracion: 1 y: -1.799999999999999333866185224906b-1
iteracion: 2 y: -1.638000000000000115463194561016b-1
iteracion: 3 y: -1.503847799999999568103703495581b-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
(%o72) done

```

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

```

--> p(x):=2*x^4-3*x^2+3*x-4;define(dp(x),diff(p(x),x,1));
(%o1) p(x):=2 x^4-3 x^2+3 x-4
(%o2) 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.

```

--> 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)
);
(%o26) horner(xi):=
block([res,n],res:coeff(p(x),x,4),n:4,for i thru n do res:coeff(p(x),x,4-i)+res xi,return(res))

--> is(horner(-2)=p(-2));
(%o27) 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.



```
--> 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)
);
```

```
(%o25) hornerdef(xi):=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*xi, L:append(L,[res])), res:L[1], for i from 2 thru n do res:L[i]+res*xi, return(res))
```

```
--> is(hornerdef(-2)=dp(-2));
```

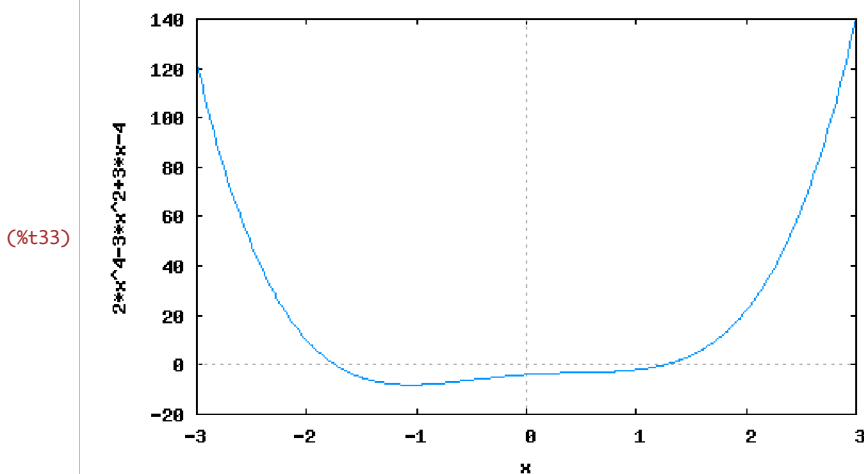
```
(%o24) true
```

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 solución  $z$ , del polinomio  $p(x)=2x^4-3x^2+3x-4$ , cumple que para  $h=\max\{|a_i/a_n|, \text{ para } i=1, \dots, n\}$ ,  $|z|<h+1$ , para este caso concreto  $|z|<3$ , luego podemos dibujar la gráfica en el intervalo de  $x$  perteneciente a  $[-3,3]$  en el cual visualizaremos todas las soluciones.

Como podemos observar contando el número de cortes del eje de la  $x$ , hay 2 soluciones reales, luego como todo polinomio de grado  $n$  tiene exactamente  $n$  soluciones en  $\mathbb{C}$ , y el número de soluciones complejas siempre es par, podemos afirmar que existen 2 soluciones reales y 2 complejas, todas ellas contenidas en el intervalo  $[-3,3]$

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



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.

```
--> (horner(-2)/hornerdef(-2));
```

```
(%o35) -10/49
```

```
--> 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))
);
```

```
(%o54) NewtonRapson(x0):=block(fprrec:32, for i thru 5 do (x0: x0 -  $\frac{\text{horner}(x0)}{\text{hornerdef}(x0)}$ , print(x, i, : , bfloat(x0))),
print(resultado : , bfloat(x0)))
```

```
--> NewtonRapson(-2)$
```

```
x 1 : -1.795918367346939b0
x 2 : -1.742432916750542b0
x 3 : -1.738970235336167b0
x 4 : -1.738956256679049b0
x 5 : -1.738956256451892b0
resultado : -1.738956256451892b0
```

- 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 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 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<sup>-15</sup>:

¿Cuántas iteraciones se han calculado?

¿Qué proporciona la última de las iteraciones calculada? ¿Por qué?

```
--> NewtonRapsonTol(x0,n,tol):=block([x,j],
  fprrec:32,
  for i:1 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)
);
```

```
(%o62) NewtonRapsonTol(x0, n, tol):=block([x, j], fprrec:32, for i thru n do (j: i, xn: x0 -  $\frac{\text{horner}(x0)}{\text{hornerdef}(x0)}$ ,
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))
```

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

```
x 1 : -1.795918367346939b0
x 2 : -1.742432916750542b0
x 3 : -1.738970235336167b0
x 4 : -1.738956256679049b0
x 5 : -1.738956256451892b0
x 6 : -1.738956256451892b0
Se a alcanzado la tolerancia
resultado : -1.738956256451892b0 En la iteracion 6
```