1. Show that

$$\frac{x_{n+1} - x_n}{x_n - x_{n-1}} \approx \lambda$$

Suppose that $x_{n+1} - \bar{x}$, $x_n - \bar{x}$ and $x_{n-1} - \bar{x}$

take

$$e_n = x_n - \bar{x} \quad \text{——} ①$$

$$e_{n+1} = x_{n+1} - \bar{x} \quad \text{——} ②$$

e_{n+1} can be expressed interms of e_n using:

$$e_{n+1} = \lambda e_n \quad \text{——} ③$$

Substituting ① and ② into ③ we obtain:

$$x_{n+1} - \bar{x} = \lambda (x_n - \bar{x}) \quad \text{——} ④$$

Similarly: $e_n = \lambda e_{n-1}$

Obtaining $(x_n - \bar{x}) = \lambda (x_{n-1} - \bar{x})$ —— ⑤

Subtracting ④ from ⑤

$$x_n - x_{n+1} = \lambda x_{n-1} - \lambda x_n$$

Simplifying to

$$\frac{x_{n+1} - x_n}{x_n - x_{n-1}} \approx \lambda$$

2. For Superlinear Convergence $\alpha = 1$, $\lambda = 0$, hence

$$\lim_{n\to\infty} \frac{|x_{n+1} - \bar{x}|}{|x_n - \bar{x}|} = 0$$

Consider $\displaystyle\lim_{n\to\infty} \frac{|x_{n+1} - \bar{x}|}{|x_n - \bar{x}|^{\alpha}} = \lambda$, for $\lambda \neq 0$ and $\alpha > 1$

So

$$\lim_{n\to\infty} \frac{|x_{n+1} - \bar{x}|}{|x_n - \bar{x}|} = \lim_{n\to\infty} \frac{|x_{n+1} - \bar{x}|}{|x_n - \bar{x}|^{(1 - \alpha + \alpha)}}$$

$$= \lim_{n\to\infty} \frac{|x_{n+1} - \bar{x}|}{|x_n - \bar{x}|^{\alpha} \, |x_n - \bar{x}|^{1-\alpha}}$$

$$= \lim_{n\to\infty} \frac{|x_{n+1} - \bar{x}|}{|x_n - \bar{x}|^{\alpha}} |x_n - \bar{x}|^{\alpha - 1}$$

Since $\displaystyle\lim_{n\to\infty} \frac{|x_{n+1} - \bar{x}|}{|x_n - \bar{x}|^{\alpha}} = \lambda$, then,

$$\lim_{n\to\infty} \frac{|x_{n+1} - \bar{x}|}{|x_n - \bar{x}|} = \lambda \lim_{n\to\infty} |x_n - \bar{x}|^{\alpha - 1}$$

Since $\alpha > 1$, it means fast convergence therefore x_n tends quickly to $\bar{x}$, hence $\displaystyle\lim_{n\to\infty} |x_n - \bar{x}|^{\alpha - 1} = 0$ as α increases. thus,

$$\lim_{n\to\infty} \frac{|x_{n+1} - \bar{x}|}{|x_n - \bar{x}|} = \lambda \cdot 0 = 0$$

which is Superlinear convergence since $\alpha = 1$, and $\lambda = 0$

3. Suppose that $\{x_n\}$ Converges Superlinearly to $\bar{x}$. Show that

$$\lim_{n\to\infty} \frac{|x_{n+1} - x_n|}{|x_n - \bar{x}|} = 1$$

$$\lim_{n\to\infty} \frac{|x_{n+1} - x_n|}{|x_n - \bar{x}|} = \lim_{n\to\infty} \frac{|x_{n+1} - \bar{x} + \bar{x} - x_n|}{|x_n - \bar{x}|}$$

$$\leq \lim_{n\to\infty} \frac{|x_{n+1} - \bar{x}|}{|x_n - \bar{x}|} + \lim_{n\to\infty} \frac{|x_n - \bar{x}|}{|x_n - \bar{x}|}$$

$$\lim_{n\to\infty} \frac{|x_{n+1} - x_n|}{|x_n - \bar{x}|} \leq 1 + \lim_{n\to\infty} \frac{|x_{n+1} - \bar{x}|}{|x_n - \bar{x}|}$$

for Super linear Convergence, $\lim_{n\to\infty} \frac{|x_{n+1} - \bar{x}|}{|x_n - \bar{x}|} = 0$

$$\lim_{n\to\infty} \frac{|x_{n+1} - x_n|}{|x_n - \bar{x}|} \leq 1 + 0$$

hence

$$\lim_{n\to\infty} \frac{|x_{n+1} - x_n|}{|x_n - \bar{x}|} = 1$$

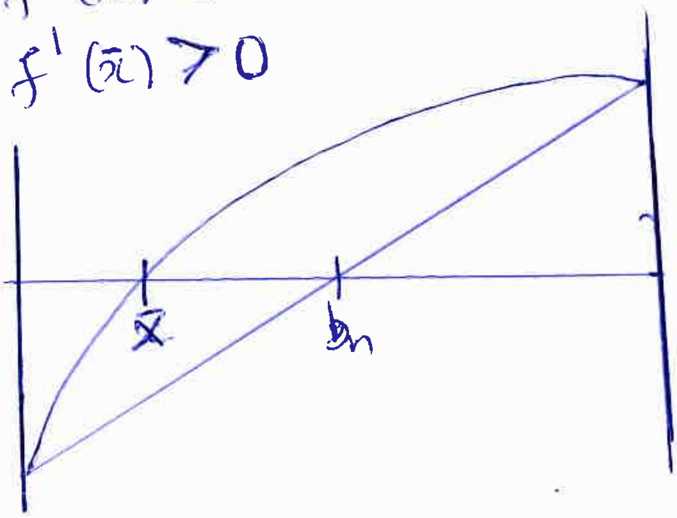
While developing root-finding Schemes we are Interested in how fast the Solution Converges to the root, So this makes us to be Interested in the error between the Solution and the root. So If the error be at all possible Solutions near the root are the same, giving 1 after division, this means that the scheme approximates well the root. Therefore $|x_n - \bar{x}| = |x_{n+1} - x_n|$ gives more Information on how fast the Scheme will Converge to the ~~solution~~ root.

1. Show that $\lambda \approx \dfrac{Lf''(\bar{x})}{2f'(\bar{x}) + Lf''(\bar{x})}$ Satisfies $|\lambda|<1$.

Using the four possible Cases.

$$\lambda \approx \frac{L f''(\bar{x})}{2f'(\bar{x}) + Lf''(\bar{x})}, \text{ where } L = \begin{cases} b_n - \bar{x}, & b_n \text{ fixed} \\ a_n - \bar{x}, & a_n \text{ fixed.} \end{cases}$$

Case 1: $f''(\bar{x}) < 0$
$f'(\bar{x}) > 0$



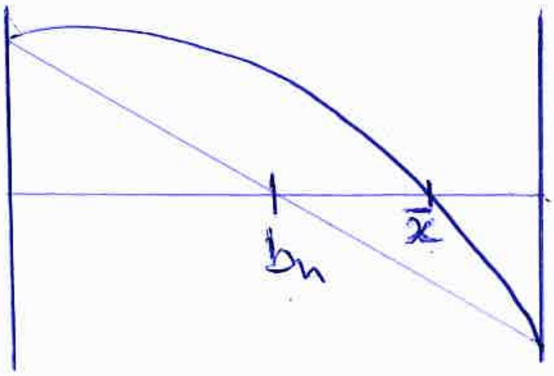
If b_n is fixed, $L = b_n - \bar{x}$.
from the diagram, $b_n - \bar{x} > 0$ and $f''(\bar{x}) < 0$
therefore $(b_n - \bar{x})(f''(\bar{x})) < 0$.
Since $f'(\bar{x}) > 0$, it follows that

$$2f'(\bar{x}) + (b_n - \bar{x}) f''(\bar{x}) > (b_n - \bar{x}) f''(\bar{x})$$

$$1 > \frac{(b_n - \bar{x}) f''(\bar{x})}{2f'(\bar{x}) + (b_n - \bar{x}) f''(\bar{x})}$$

$$1 > \left| \frac{(b_n - \bar{x}) f''(\bar{x})}{2f'(\bar{x}) + (b_n - \bar{x}) f''(\bar{x})} \right| = |\lambda|, \text{ hence } |\lambda| < 1$$

Case 2: $f''(\bar{x}) < 0$
$f'(\bar{x}) < 0$



If b_n is fixed, $L = b_n - \bar{x}$

from the diagram; $b_n - \bar{x} < 0$ and $f''(x) < 0$

$|(b_n - \bar{x}) f''(\bar{x})| > 0$

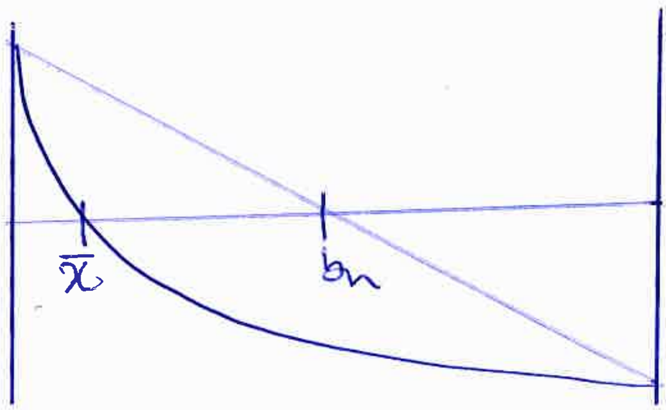
Since $f'(\bar{x}) < 0$, it follows that

$$|2f'(\bar{x}) + (b_n - \bar{x}) f''(\bar{x})| > |(b_n - \bar{x}) f''(\bar{x})|$$

$$1 > \left| \frac{(b_n - \bar{x}) f''(\bar{x})}{2f'(\bar{x}) + (b_n - \bar{x}) f''(\bar{x})} \right| = |\lambda|$$

hence $\underline{|\lambda| < 1}$

Case 3: $f''(\bar{x}) > 0$
$f'(\bar{x}) < 0$



fixing b_n, $(b_n - \bar{x}) > 0$

Since $f''(\bar{x}) > 0$

then $|(b_n - \bar{x}) f''(\bar{x})| > 0$

Since $f'(\bar{x}) < 0$, then it follows that

$$|2f'(\bar{x}) + (b_n - \bar{x}) f''(\bar{x})| > |(b_n - \bar{x}) f''(\bar{x})|$$

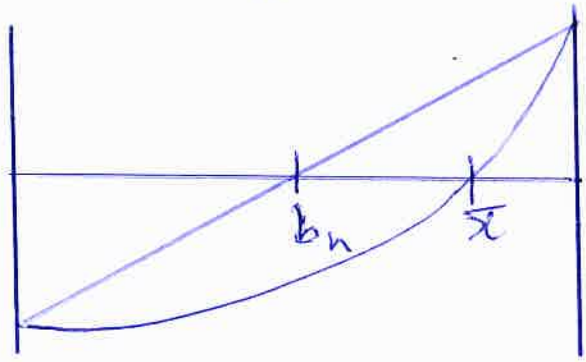
$$1 > \left| \frac{(b_n - \bar{x}) f''(\bar{x})}{2f'(\bar{x}) + (b_n - \bar{x}) f''(\bar{x})} \right| = |\lambda|$$

hence

$$|\lambda| < 1$$

Case 4: $f''(\bar{x}) > 0$

$f'(x) > 0$



fixing b_n, $(b_n - \bar{x}) < 0$ and $f''(\bar{x}) > 0$

$$(b_n - \bar{x}) f''(\bar{x}) < 0$$

Since $f'(x) > 0$, it follows that

$$2f'(\bar{x}) + (b_n - \bar{x}) f''(\bar{x}) > (b_n - \bar{x}) f''(\bar{x})$$

$$1 > \left| \frac{(b_n - \bar{x}) f''(\bar{x})}{2f'(\bar{x}) + (b_n - \bar{x}) f''(\bar{x})} \right| = |\lambda|$$

Hence $1 > |\lambda|$

In [17]:
```
%matplotlib notebook
%pylab
```

```
Using matplotlib backend: nbAgg
Populating the interactive namespace from numpy and matplotlib
```

N0.4

In [18]:
```
f=lambda x : (x**2) - (1e51)

def bisect(f,a,b):
    tol = 1e-10
    fa=f(a)
    kmax = int(log2((b-a)/tol)+1)
    print("Estimated Number of iterations =", kmax, "iterations")
    for k in range(kmax):
        c=(a+b)/2
        fc=f(c)
        if sign(f(a)) != sign(f(c)):
            b=c
        else:
            a=c
            fa=fc
        k+=1
        if abs(b-a) < tol:
            print("Converged in {:d} iterations".format(k))
            break
    return c

xroot = bisect(f,2e25,4e25)
#print(xroot)
#print(f(xroot))
print("x = {:24.16f}".format(xroot))
print("f(x) = {:24.4e}".format(f(xroot)))
```

```
Estimated Number of iterations = 118 iterations
x = 31622776601683794681921536.0000000000000000
f(x) =              1.6615e+35
```

Can we actually achieve this tolerance?

No

Why?

Because we know that the iterations in bisection method is always converging to the xroot, and f(xroot) must be zero which is not the case in our calculations, then we cant achieve this tolerance.

How would you choose a more appropriate tolerance?

Since the appropriate tolerance depends on the number of iterations N, and N depends only on the initial interval $[a_o, b_o]$ bracketing the root. Therefore the interval length after N iterations is $\frac{b_o - a_o}{2^N}$ and this must be less or equal to τ to obtain an accuracy of τ i.e.

$$\frac{b_o - a_o}{2^N} \leq \tau$$

No. 5

In [19]:
```
def fixed_point(f,xo,beta):
    x=xo #initial gues
    kmax=100
    tol=1e-10
    for k in range(kmax):
        x1=f(x)
        if abs(x1-x) < tol:
            print('Tolerance achieved\n')
            xroot=x1
            break
        x=x1
    print('The root = ',x1)
    print('Number of iterations = ',k)
```

In [20]:
```
f= lambda x:(1/3)*(x**3)-(x**2)+(4/3)*beta
beta=0.1
xo=0.1
fixed_point(f,xo,beta)
```

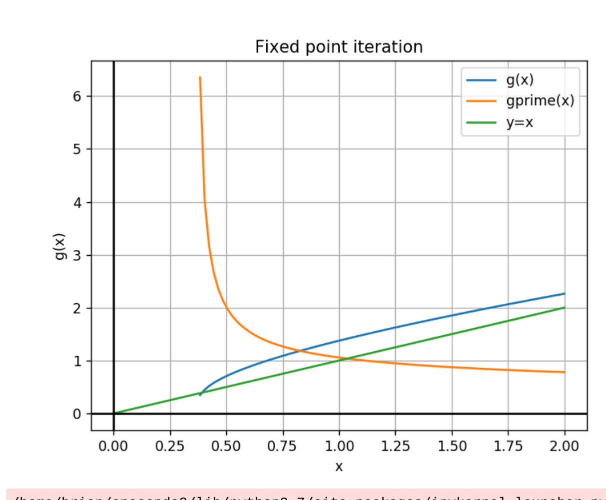
```
Tolerance achieved

The root =  0.1195995366894333
Number of iterations =  13
```

In [21]:
```
f= lambda x:(1/3)*(x**3)-(x**2)+(4/3)*beta
beta=0.95
xo=0.1
fixed_point(f,xo,beta)
```

```
The root =  0.7942962233400211
Number of iterations =  99
```

$\beta = 0.1$

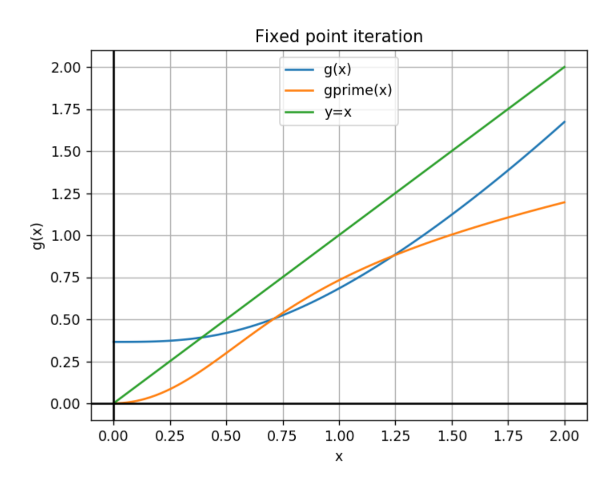
In [22]:
```
beta=0.1
#let f(x)=0 be written in the form x=g(x)
#first function g(x)
x=linspace(0,2,100)
g=lambda x:(3*(x**2)-(4*beta))**(1/3)
gprime=lambda x:(2*x)/(((3*(x**2))-(4*beta))**(2/3))
y=lambda x:x

figure(1)
plot(x,g(x),label='g(x)')
plot(x,gprime(x),label='gprime(x)')
plot(x,y(x),label='y=x')
xlabel('x')
ylabel('g(x)')
title('Fixed point iteration')
axhline(y=0, color='k')
axvline(x=0, color='k')
grid()
legend()
show()
```




```
/home/brian/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5: RuntimeWarning: invalid va
lue encountered in power
  """
/home/brian/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:6: RuntimeWarning: invalid va
lue encountered in power
```

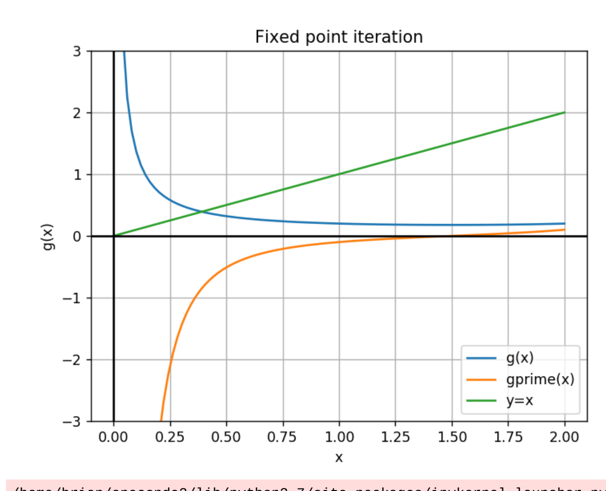
From the graph above, it is easily seen that for all x $\in$ $[0, 2]$, g(x) does not take every value in the interval $[0, 2]$, even though it intersects with y=x. And also $g^{'}(x)$ exists but doesn't satisfy $|g^{'}(x)| \leq r$ for r between $(0, 1)$. therefore x=g(x) doesnot have a unique solution, hence doesn't converge for g(x) $\in$ $[0, 2]$

In [23]:
```
beta=0.1
#let f(x)=0 be written in the form x=g(x)
#second function g(x)
x=linspace(0,2,100)
g=lambda x:((1/3)*(x**3)+((4/3)*beta))**(1/2)
gprime=lambda x:((sqrt(3))*(x**2))/(2*sqrt((4*beta)+(x**3)))
y=lambda x:x

figure(2)
plot(x,g(x),label='g(x)')
plot(x,gprime(x),label='gprime(x)')
plot(x,y(x),label='y=x')
xlabel('x')
ylabel('g(x)')
title('Fixed point iteration')
axhline(y=0, color='k')
axvline(x=0, color='k')
grid()
legend()
show()
```



From the graph above, it is easily seen that for all x $\in$ $[0, 2]$, g(x) takes on every value in the interval $[0, 2]$. Since also $g^{'}(x)$ exists and satisfies $|g^{'}(x)| \leq r$ for r between $(0, 1)$ and it intersects with y=x with in the domain where $|g^{'}(x)| \leq r$. Therefore x=g(x) has a unique solution between $[0, 2]$, hence x coverges to a fixed point α for g(x) $\in$ $[0, 2]$, at a point where g(x) intersects y=x.

In [24]:
```
beta=0.1
#let f(x)=0 be written in the form x=g(x)
#third function g(x)
x=linspace(0,2,100)
g=lambda x:(-4*beta)/((x**2)-(3*x))
gprime=lambda x:((4*beta)*((2*x)-3))/(((x-3)**2)*(x**2))
y=lambda x:x

figure(3)
plot(x,g(x),label='g(x)')
plot(x,gprime(x),label='gprime(x)')
plot(x,y(x),label='y=x')
axhline(y=0, color='k')
axvline(x=0, color='k')
grid()
xlabel('x')
ylabel('g(x)')
ylim(-3,3)
title('Fixed point iteration')
legend()
show()
```

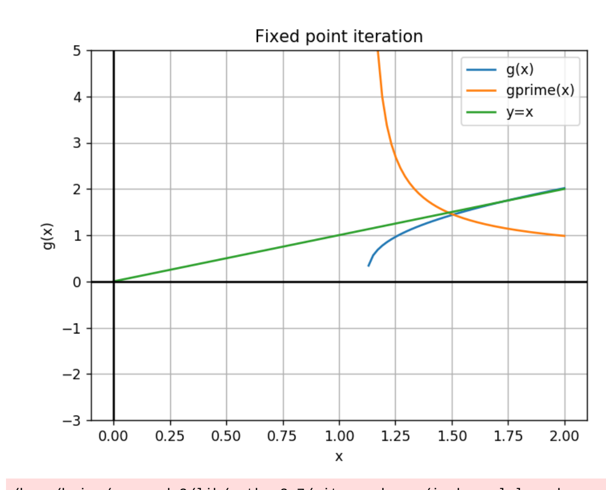



```
/home/brian/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5: RuntimeWarning: divide by
zero encountered in true_divide
  """
/home/brian/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:6: RuntimeWarning: divide by
zero encountered in true_divide
```

From the graph above, it is easily seen that for all x $\in$ $[0, 2]$, g(x) does not take every value in the interval $[0, 2]$, since the last point is $-\infty$ hence diverges, even though it intersects with y=x. and that $g^{'}(x)$ exists and satisfis $|g^{'}(x)| \leq r$ for r between $(0, 1)$. Therefore x=g(x) doesnot have a unique solution, hence doesn't converge for g(x) $\in$ $[0, 2]$

$\beta = 0.95$

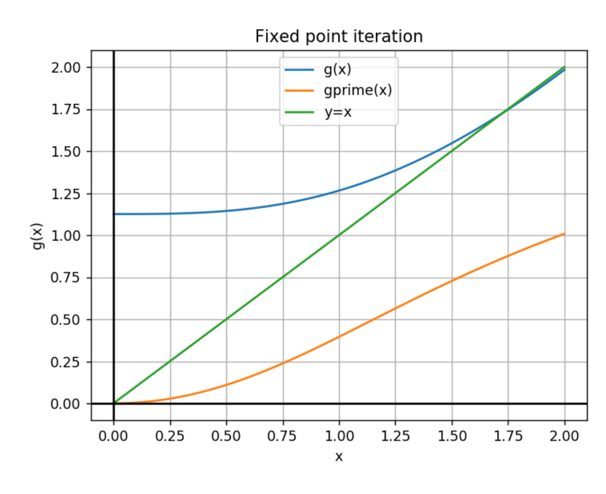
In [25]:
```
beta=0.95
#let f(x)=0 be written in the form x=g(x)
#first function g(x)
x=linspace(0,2,100)
g=lambda x:(3*(x**2)-(4*beta))**(1/3)
gprime=lambda x:(2*x)/(((3*(x**2))-(4*beta))**(2/3))
y=lambda x:x

figure(4)
plot(x,g(x),label='g(x)')
plot(x,gprime(x),label='gprime(x)')
plot(x,y(x),label='y=x')
axhline(y=0, color='k')
axvline(x=0, color='k')
grid()
xlabel('x')
ylabel('g(x)')
ylim(-3,5)
title('Fixed point iteration')
legend()
show()
```




```
/home/brian/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5: RuntimeWarning: invalid va
lue encountered in power
  """
/home/brian/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:6: RuntimeWarning: invalid va
lue encountered in power
```

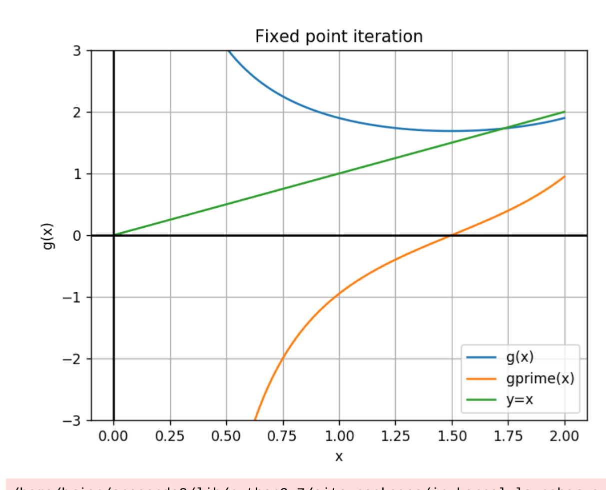
From the graph above, it is easily seen that for all x $\in$ $[0, 2]$, g(x) does not take every value in the interval $[0, 2]$, even though it intersects with y=x. And also $g^{'}(x)$ exists but doesn't satisfy $|g^{'}(x)| \leq r$ for r between $(0, 1)$, since it doesnot even exist in that range. therefore x=g(x) doesnot have a unique solution, hence doesn't converge for g(x) $\in$ $[0, 2]$

In [26]:
```
beta=0.95
#let f(x)=0 be written in the form x=g(x)
#second function g(x)
x=linspace(0,2,100)
g=lambda x:((1/3)*(x**3)+((4/3)*beta))**(1/2)
gprime=lambda x:((sqrt(3))*(x**2))/(2*sqrt((4*beta)+(x**3)))
y=lambda x:x

figure(5)
plot(x,g(x),label='g(x)')
plot(x,gprime(x),label='gprime(x)')
plot(x,y(x),label='y=x')
axhline(y=0, color='k')
axvline(x=0, color='k')
grid()
xlabel('x')
ylabel('g(x)')
title('Fixed point iteration')
legend()
show()
```



From the graph above, it is easily seen that for all x $\in$ $[0, 2]$, g(x) takes on every value in the interval $[0, 2]$. Since also $g^{'}(x)$ exists and satisfies $|g^{'}(x)| \leq r$ for r between $(0, 1)$ and it intersects with y=x with in the domain where $|g^{'}(x)| \leq r$. Therefore x=g(x) has a unique solution between $[0, 2]$, hence x coverges to a fixed point α for g(x) $\in$ $[0, 2]$, at a point where g(x) intersects y=x.

In [27]:
```
beta=0.95
#let f(x)=0 be written in the form x=g(x)
#third function g(x)
x=linspace(0,2,100)
g=lambda x:(-4*beta)/((x**2)-(3*x))
gprime=lambda x:((4*beta)*((2*x)-3))/(((x-3)**2)*(x**2))
y=lambda x:x

figure(6)
plot(x,g(x),label='g(x)')
plot(x,gprime(x),label='gprime(x)')
plot(x,y(x),label='y=x')
xlabel('x')
ylabel('g(x)')
ylim(-3,3)
title('Fixed point iteration')
axhline(y=0, color='k')
axvline(x=0, color='k')
grid()
legend()
show()
```

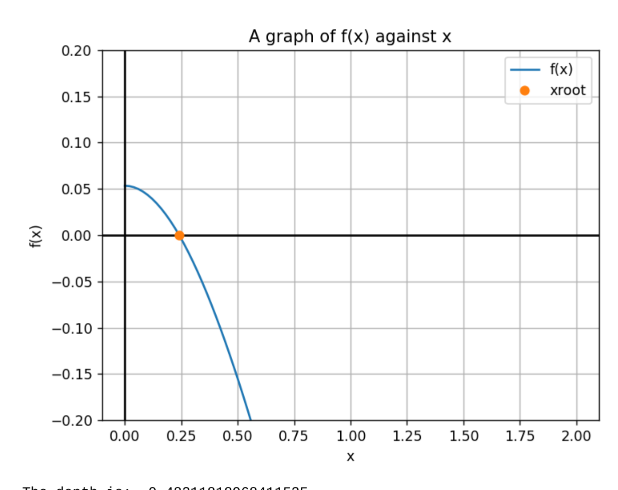



```
/home/brian/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5: RuntimeWarning: divide by
zero encountered in true_divide
  """
/home/brian/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:6: RuntimeWarning: divide by
zero encountered in true_divide
```

From the graph above, it is easily seen that for all x $\in$ $[0, 2]$, g(x) does not take every value in the interval $[0, 2]$, since the last point is $-\infty$ hence diverges, even though it intersects with y=x. and that $g^{'}(x)$ exists and satisfis $|g^{'}(x)| \leq r$ for r between $(0, 1)$. Therefore x=g(x) doesnot have a unique solution, hence doesn't converge for g(x) $\in$ $[0, 2]$

No.6

In [28]:
```
r=2 #radius
dm=0.04 #density of marble
dw=0.998 #density of water

beta=dm/dw

f=lambda x:(1/3)*(x**3)-(x**2)+(4/3)*beta
h=lambda x:r*x
ao,bo=0,2
n=100
x=linspace(ao,bo,n)

#location of t_g
root=zeros(len(x))
mark=zeros(len(x))
mark[0]=0



figure(7)
plot(x,f(x),label='f(x)')
ylim(-0.2,0.2)
grid()
xlabel('x')
ylabel('f(x)')
title('A graph of f(x) against x')
axhline(y=0, color='k')
axvline(x=0, color='k')

#bisection method
def bisection(f,a,b):
    tol=1e-10
    fa=f(a)
    kmax=int(log2((b-a)/tol)+1)
    k=0
    while k<kmax:
        c=(a+b)/2
        fc=f(c)
        if fa*fc<0:
            b=c
        else:
            a=c
            fa=fc
        k+=1
        if abs(b-a)<tol:
            break
    return c

xroot=bisection(f,ao,bo)
root[0]=xroot
plot(root,mark,markevery=xroot, ls="", marker="o", label="xroot")
legend()
show()
print('The depth is: ',h(xroot))
```

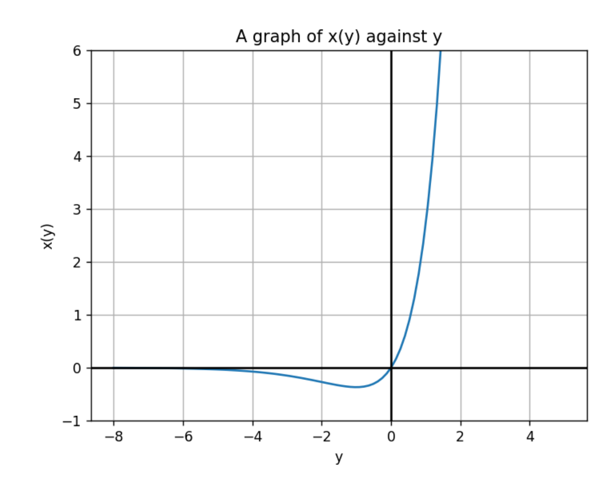



```
The depth is:  0.48211818968411535
```

N0.8

a)

In [29]:
```
x=lambda y:y*exp(y)
#xprime=lambda y:(y+1)*exp(y)
y=linspace(-8,5,100)
figure(8)
plot(y,x(y))
ylim(-1,6)
grid()
xlabel('y')
ylabel('x(y)')
title('A graph of x(y) against y')
axhline(y=0, color='k')
axvline(x=0, color='k')
show()
```



Acording to the graph above we have a turning point between 0 and 2, therefore the curve $x = y\exp(y)$ fails the horizontal test. So since $L(x) = \frac{dx}{dy} = (y + 1)\exp(y)$, at turning point:

$$\frac{dx}{dy} = 0 \implies (y + 1)\exp(y) = 0$$

Thus,

either

$$\exp(y) = 0 \implies y = \infty$$

or

$$y + 1 = 0 \implies y = -1$$

For the turning points,using $x = y\exp(y)$:

for $y = -1 \implies x = -1/e$ and ∞ for $y = \infty$ respectively.

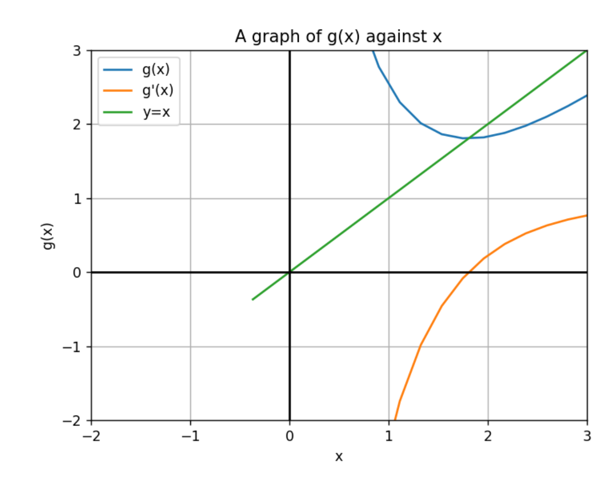
Therefore concluding that the range of $x = y\exp(y)$ is $[-1/e, \infty)$, since L(x)is the inverse of $y\exp(y)$, and its known that range of $y\exp(y)$ is the domain of its inverse, L(x).

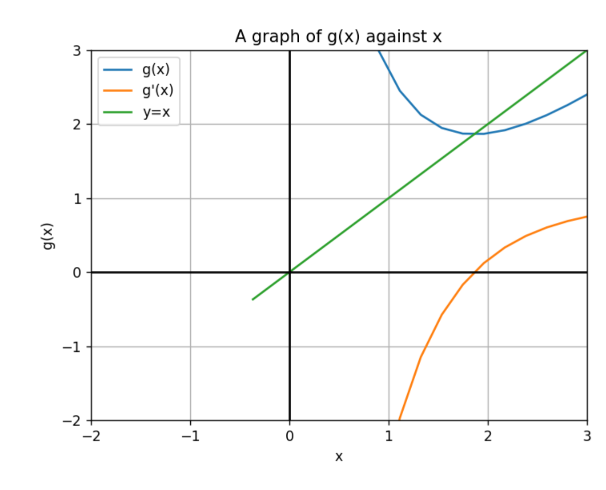
b)

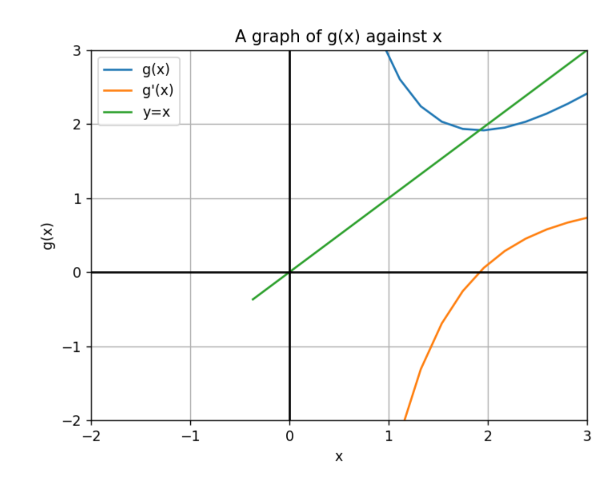
In [30]:
```
def L(x):
    f=lambda y:y*exp(y)
    fprime=lambda y:(y+1)*exp(y)

    tol=1e-10
    n=100

    #intial guess yo
    if x<1:
        yo=-0.01
    else:
        yo=log(x)

    i=0
    while i<n:
        fo=f(yo)
        fpo=fprime(yo)
        yn=yo-(fo-x)/fpo
        if abs(yn-yo)<tol:
            break
        yo=yn
        i+=1
    return yn
```

c)

In [60]:
```
def g(x, y):
    return (x/(x+1))*(x + y/(x*exp(x)))

def gprime(x, y):
    return ((x+2)/((x+1)**2))*(x - (y/exp(x)))

x = linspace(-1/e ,10)

for y in range(11,14): #sample values of y
    figure(y)
    plot(x,g(x,y),label='g(x)')
    plot(x,gprime(x,y),label="g'(x)")
    plot(x,x,label='y=x')
    grid()
    axhline(y=0, color='k')
    axvline(x=0, color='k')
    ylabel('g(x)')
    xlabel('x')
    ylim(-2,3)
    xlim(-2,3)
    title('A graph of g(x) against x')
    legend()
    show()
```







From the graphs above, it is easily seen that for all x $\in$ $[-1/e, \infty)$, g(x) takes on every value in the interval $[-1/e, \infty)$. Since also $g^{'}(x)$ exists and satisfies $|g^{'}(x)| \leq r$ for r between $(0, 1)$. therefore x=g(x) has a unique solution between $[-1/e, \infty)$, and it intersects with y=x with in the domain where $|g^{'}(x)| \leq r$. Hence x coverges to a fixed point α for g(x) $\in$ $[-1/e, \infty)$, at a point where g(x) intersects y=x for x $\in$ $[-1/e, \infty)$. Hence since x=g(x) converges then also Newton's method will also converge in all graphs, therefore we can suppose that it will converge for other values of y for any x in the domain $[-1/e, \infty)$.

d)

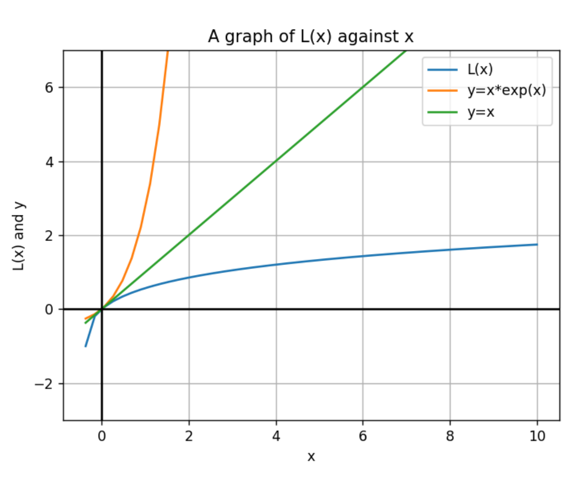
In [33]:
```
#check that L(xe^x)=x
x=[3,10,101]
for i in x:
    print('x=',i,'=',L(i*exp(i)))
```

```
x= 3 = 3.0
x= 10 = 10.0
x= 101 = 101.0
```

Hence we are accurately computing $L(x\exp(x)) = x$

e)

In [34]:
```
x=linspace(-1/e,10) #for aprinciple branch
L1=vectorize(L)

y=lambda x:x*exp(x)

figure(9)
plot(x,L1(x),label='L(x)')
plot(x,y(x),label='y=x*exp(x)')
plot(x,x,label='y=x')
axhline(y=0, color='k')
axvline(x=0, color='k')
ylabel('L(x) and y')
xlabel('x')
title('A graph of L(x) against x')
legend()
ylim(-3,7)
grid()
show()
```



Graphically we can see that one is the opposite of the other hence one is the inverse of the other

No.9

a)

Following the steps above, at h(t)=0, we have

$$\frac{33}{784}t + \exp(-0.3t) = 1 \quad (1)$$

Comparing it with the general form:

$$At + \exp(Bt) = 1$$

we clearly see that $A = \frac{33}{784}$ and $B = -0.3$.

Using the transformation

$$t = -\frac{1}{B}u + \frac{1}{A}$$

and substituting for t in equation (1) we obtain

$$u\exp(u) = \frac{B}{A}\exp(\frac{B}{A})$$

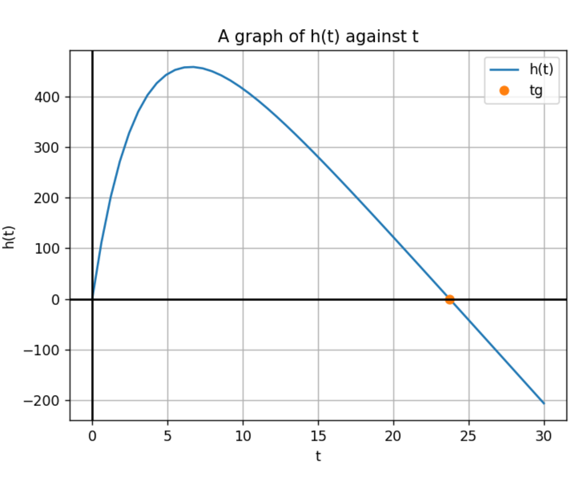
which implicitly defines the Lambert W function.

In [35]:
```
A=33/784; B=-0.3
C=B/A

#calling the Lambert W function L(x) to use it to calculate u
u=L(C*exp(C))

#time t_g for the rocket to hit the ground
t_g=lambda u:-(1/B)*u+(1/A)
print("t_g = {:8f}".format(t_g(u)),'Seconds')
```

```
t_g = 23.738391 Seconds
```

b)

In [36]:
```
h=lambda t:-33*t + 784*(1-exp(-0.3*t))
t=linspace(0,30)
root=zeros(len(t))
root[0]=t_g(u)
mark=zeros(len(t))
mark[0]=0

figure(10)
plot(t,h(t),label='h(t)')
plot(root,mark,markevery=t_g(u), ls="", marker="o", label="tg")
axhline(y=0, color='k')
axvline(x=0, color='k')
ylabel('h(t)')
xlabel('t')
title('A graph of h(t) against t')
legend()
grid()
show()
```



The value of t_g is correct, since we cosidered that at $h = 0$ is the ground, and looking on the graph, this is the exact time at which $h(t)$ is zero.

c)

Since in our equation h(t), on the right hand side t appears both inside and outside in the exponential function. So this means solving this equation directly will give us only on solution, which is not right, since h(t) is a multivalued function, thus generally it has morethan on solution. And also since we are dealing with rocket height, we expect large positive values of both t and h, with these large values t, $\exp(t)$, and $t\exp(t)$ grow similarly implying that also their inverse functions will have similar asymptotes. So $t\exp(t)$ will have two real branches. Giving better results than solving it directly.

In [ ]: