Bisection Algorithm

Bisection Algorithm -- Bracketing Method

Solve $f(x) = 0$

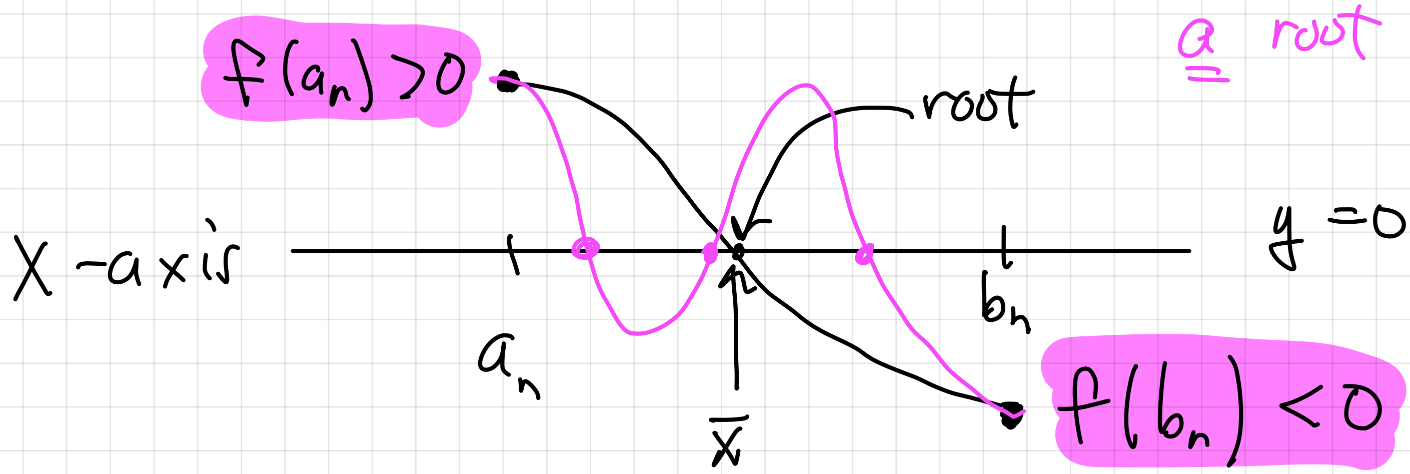
Idea: Find a sequence of intervals

$$I_n = [a_n, b_n], \quad a_n < b_n$$

where $\lim\limits_{n \to 0} b_n - a_n = 0$

and $f(a_n) f(b_n) < 0$

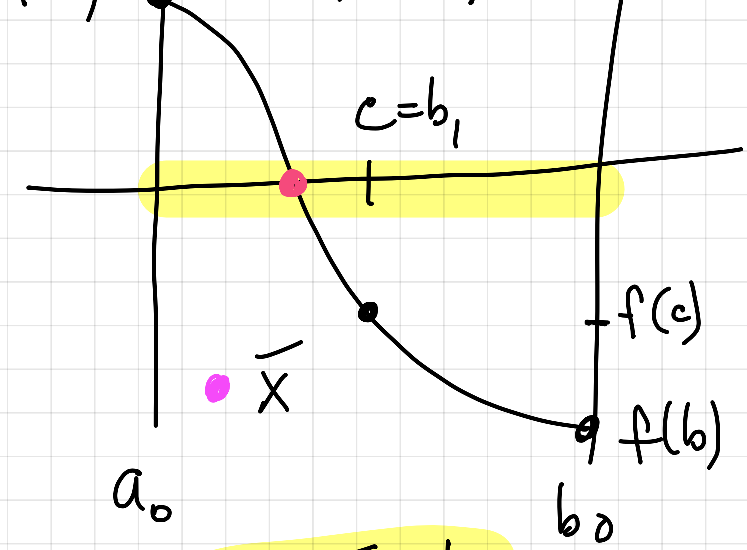
test for whether $[a_n, b_n]$ includes a root



Intermediate value theorem tells us that $f(x)$ has to cross the x-axis between a_n and b_n.

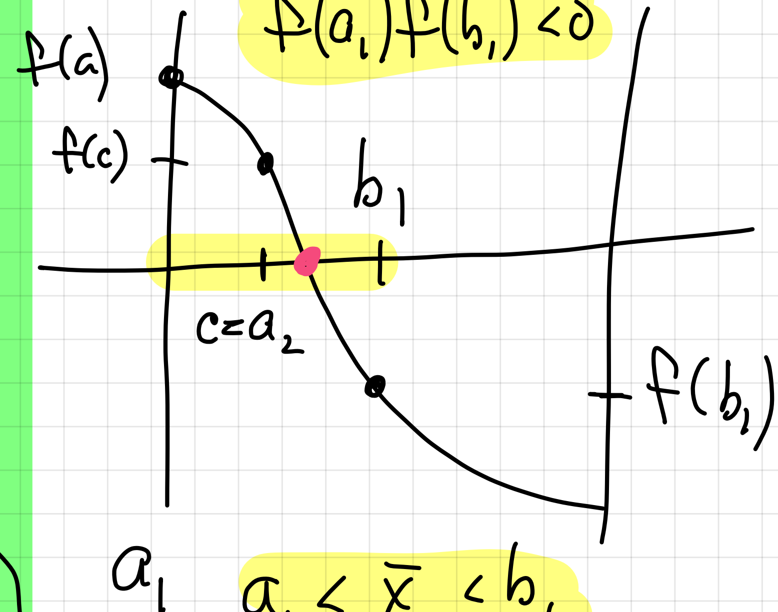
Bisection method

$f(a_0)f(b_0)<0$



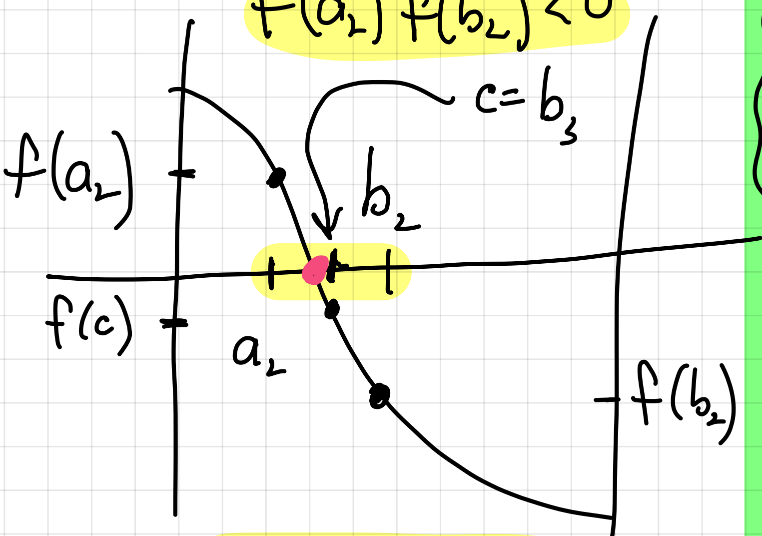
$a_0 < \bar{x} < b_0$

$f(a_1)f(b_1)<0$



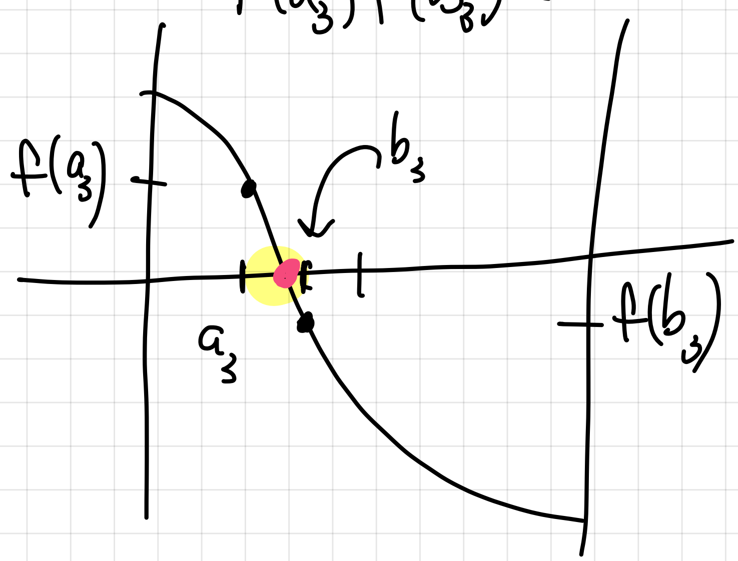
$a_1 < \bar{x} < b_1$

$f(a_2)f(b_2)<0$



$a_2 \le \bar{x} \le b_2$

$f(a_3)f(b_3)<0$



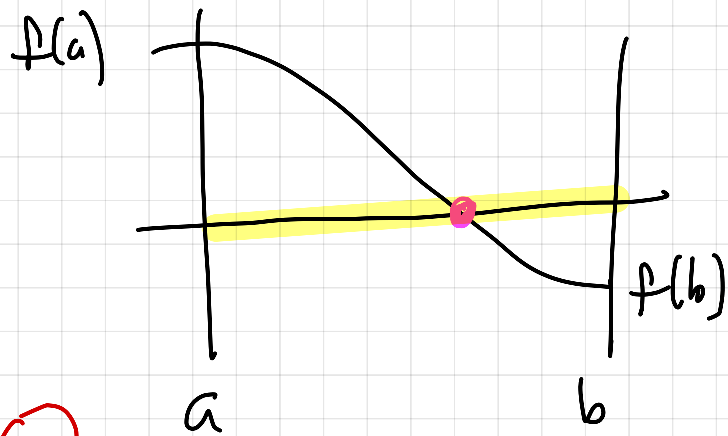
$a_3 \le \bar{x} \le b_3$

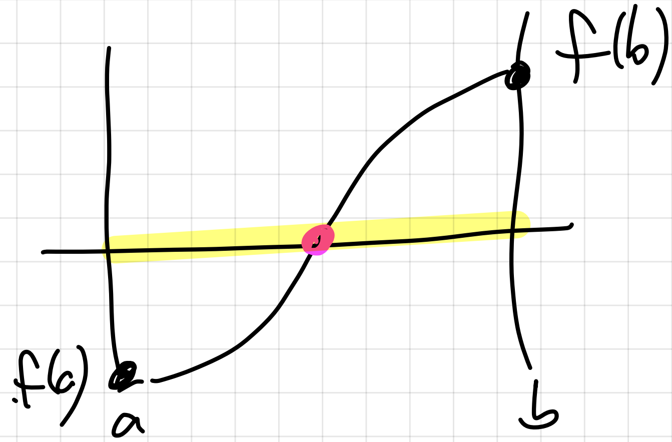
Key: $\bar{x} \in [a_n, b_n]$; $|b_n - a_n| \longrightarrow 0$

Bisection Method

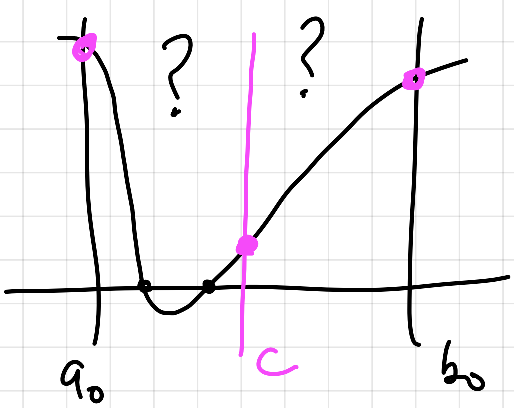
Required: an initial interval $[a,b]$ that brackets a root & $f(a)f(b) < 0$.

How do we know our interval brackets a root? Two cases:





$f(a) > 0 \quad f(b) < 0$ $\qquad$ $f(a) < 0 \quad f(b) > 0$



$$f(a)f(b) < 0$$

(or use 'sign')

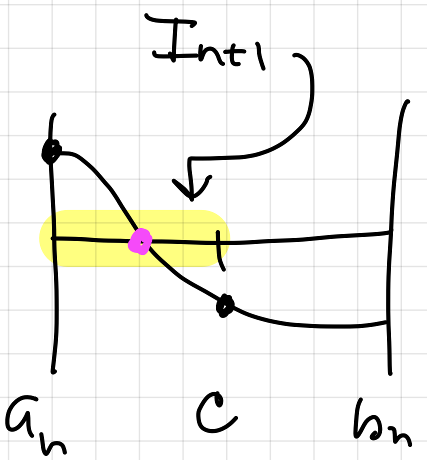
Initial interval: $I_0 = [a_0, b_0]$

Bisection Algorithm

Given $I_n = [a_n, b_n]$ containing a root:

1. Bisect the interval $c = \dfrac{a_n + b_n}{2}$

2. Consider 2 cases



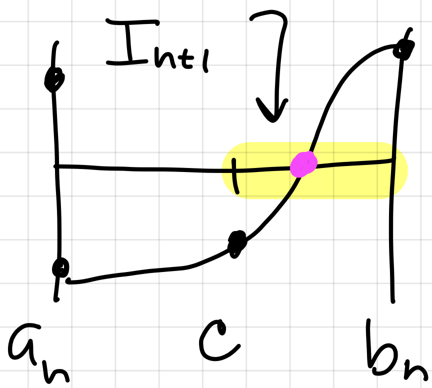
If $f(a_n) f(c) < 0$

$\Rightarrow [a_n, c]$ contains a root

$\Rightarrow a_{n+1} = a_n$

$\Rightarrow b_{n+1} = c$

Else % only other choice.



$[c, b_n]$ contains a root

$a_{n+1} = c$

$b_{n+1} = b_n$

3. $I_{n+1} = [a_{n+1}, b_{n+1}]$

4. If $|b_{n+1} - a_{n+1}| < \varepsilon$ Stop

$|c_{n+1} - \bar{x}| \leq |b_{n+1} - a_{n+1}|$

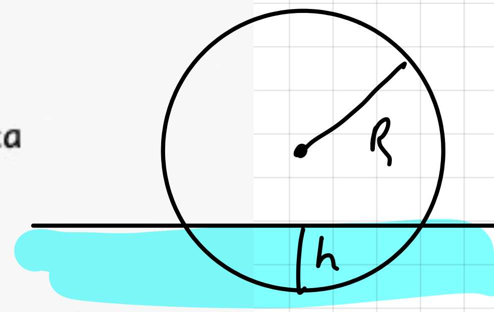
Bisection Algorithm

Solve $f(x) = \frac{1}{3}x^3 - x^2 + \frac{4}{3}\beta = 0$

$I_0 = [0,2]$

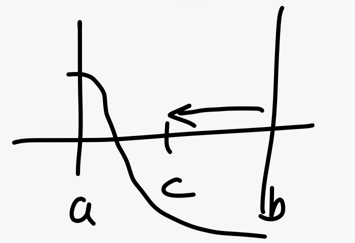
Version 1

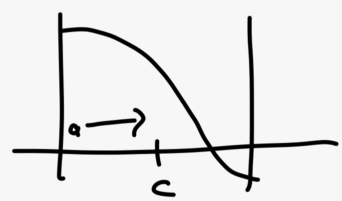
$X = .4 = \frac{h}{R}$



$h = .4R$

```python
beta = 0.1

def g(x):
    return (1/3)*x**3 - x**2 + 4/3*beta

def bisect(f,a,b):
    tol = 1e-10
    while (True):
        c = (a+b)/2;
        if (f(a)*f(c) < 0):
            b = c
        else:
            a = c
        if abs(b-a) < tol:
            return (a+b)/2

xroot = bisect(g,0,2)
print("x      = {:24.16f}".format(xroot))
print("f(x) = {:24.4e}".format(g(xroot)))
```






```
x    =       0.3916002112964634
f(x) =            1.3680e-11
```

$\Rightarrow$ We can reduce the number of function evaluations from 2 per iteration to one per iteration.

Bisection Algorithm

Version 2

```python
beta = 0.1

def g(x):
    return (1/3)*x**3 - x**2 + 4/3*beta

def bisect(f,a,b):
    fa = f(a)
    fb = f(b)
    tol = 1e-10
    while (True):
        c = (a+b)/2;
        fc = f(c)
        if (fa*fc < 0):
            fb = fc
            b = c
        else:
            a = c
            fa = fc
        if abs(b-a) < tol:
            return (a+b)/2

xroot = bisect(g,0,2)
print("x      = {:24.16f}".format(xroot))
print("f(x) = {:24.4e}".format(g(xroot)))
```

only one function eval / iteration

```
x    =        0.3916002112964634
f(x) =                1.3680e-11
```

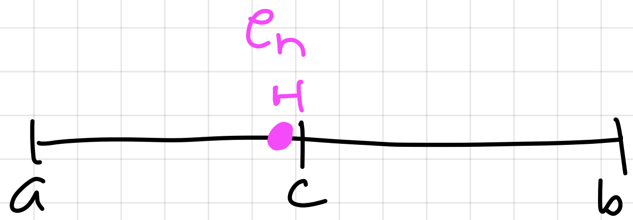
Convergence of bisection

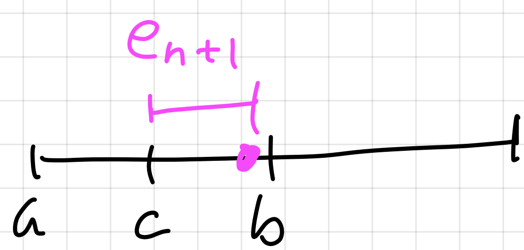
approx. root $\quad$ true root

K	a_n	b_n	$\lvert c_n - \bar{x} \rvert$	
1	0.00000000e+00	1.00000000e+00	6.08399786e-01	
2	0.00000000e+00	5.00000000e-01	1.08399786e-01	
3	2.50000000e-01	5.00000000e-01	1.41600214e-01	
4	3.75000000e-01	5.00000000e-01	1.66002142e-02	
5	3.75000000e-01	4.37500000e-01	4.58997858e-02	
6	3.75000000e-01	4.06250000e-01	1.46497858e-02	
7	3.90625000e-01	4.06250000e-01	9.75214243e-04	
8	3.90625000e-01	3.98437500e-01	6.83728576e-03	
9	3.90625000e-01	3.94531250e-01	2.93103576e-03	
10	3.90625000e-01	3.92578125e-01	9.77910757e-04	
11	3.90625000e-01	3.91601562e-01	1.34825706e-06	e_n
12	3.91113281e-01	3.91601562e-01	4.86932993e-04	e_{n+1}
13	3.91357422e-01	3.91601562e-01	2.42792368e-04	
14	3.91479492e-01	3.91601562e-01	1.20722055e-04	
15	3.91540527e-01	3.91601562e-01	5.96868992e-05	
16	3.91571045e-01	3.91601562e-01	2.91693211e-05	
17	3.91586304e-01	3.91601562e-01	1.39105320e-05	
18	3.91593933e-01	3.91601562e-01	6.28113747e-06	
19	3.91597748e-01	3.91601562e-01	2.46644020e-06	
20	3.91599655e-01	3.91601562e-01	5.59091568e-07	
21	3.91599655e-01	3.91600609e-01	3.94582748e-07	
22	3.91600132e-01	3.91600609e-01	8.22544098e-08	
23	3.91600132e-01	3.91600370e-01	1.56164169e-07	
24	3.91600132e-01	3.91600251e-01	3.69548798e-08	
25	3.91600192e-01	3.91600251e-01	2.26497650e-08	
26	3.91600192e-01	3.91600221e-01	7.15255738e-09	
27	3.91600206e-01	3.91600221e-01	7.74860381e-09	
28	3.91600206e-01	3.91600214e-01	2.98023217e-10	

```
Converged in 28 iterations
x    =       0.3916002102196217
f(x) =                6.9193e-10
```

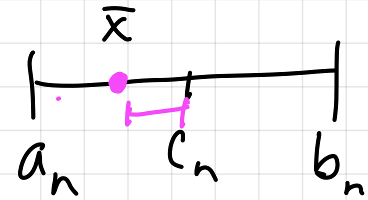
Convergence is not smooth





Convergence of the Bisection Method

Let $\bar{x}$ be the root. And let c_n be the approximation to the root in interval $I_n = [a_n, b_n]$.



Then:

$$|c_n - \bar{x}| \leq |b_n - a_n|$$

We have

$$|b_n - a_n| = \frac{1}{2}|b_{n-1} - a_{n-1}| = \frac{1}{2^2}|b_{n-2} - a_{n-2}|$$

$$\cdots = \frac{1}{2^n}|b_0 - a_0|$$

$|S_n - L|$

So $$|c_n - \bar{x}| < \underbrace{|b_0 - a_0|}_{\lambda} \cdot \underbrace{\frac{1}{2^n}}_{\beta_n}$$

The rate of convergence is then

$$\Theta\left(\frac{1}{2^n}\right)$$

Note: $\beta_n \longrightarrow 0$
$n \longrightarrow 0$

Convergence of the Bisection Method

To compute the order of convergence, we define

$$e_n = |c_n - \bar{x}| \le |b_n - a_n|$$

$$e_{n+1} = |c_{n+1} - \bar{x}| \le |b_{n+1} - a_{n+1}| < \frac{1}{2}|b_n - a_n|$$

We have that $|b_{n+1} - a_{n+1}| = \frac{1}{2}|b_n - a_n|$

So

(Assume $\alpha = 1$)

$$\frac{e_{n+1}}{e_n} \simeq \frac{\frac{1}{2}|b_n - a_n|}{|b_n - a_n|} = \frac{1}{2}$$

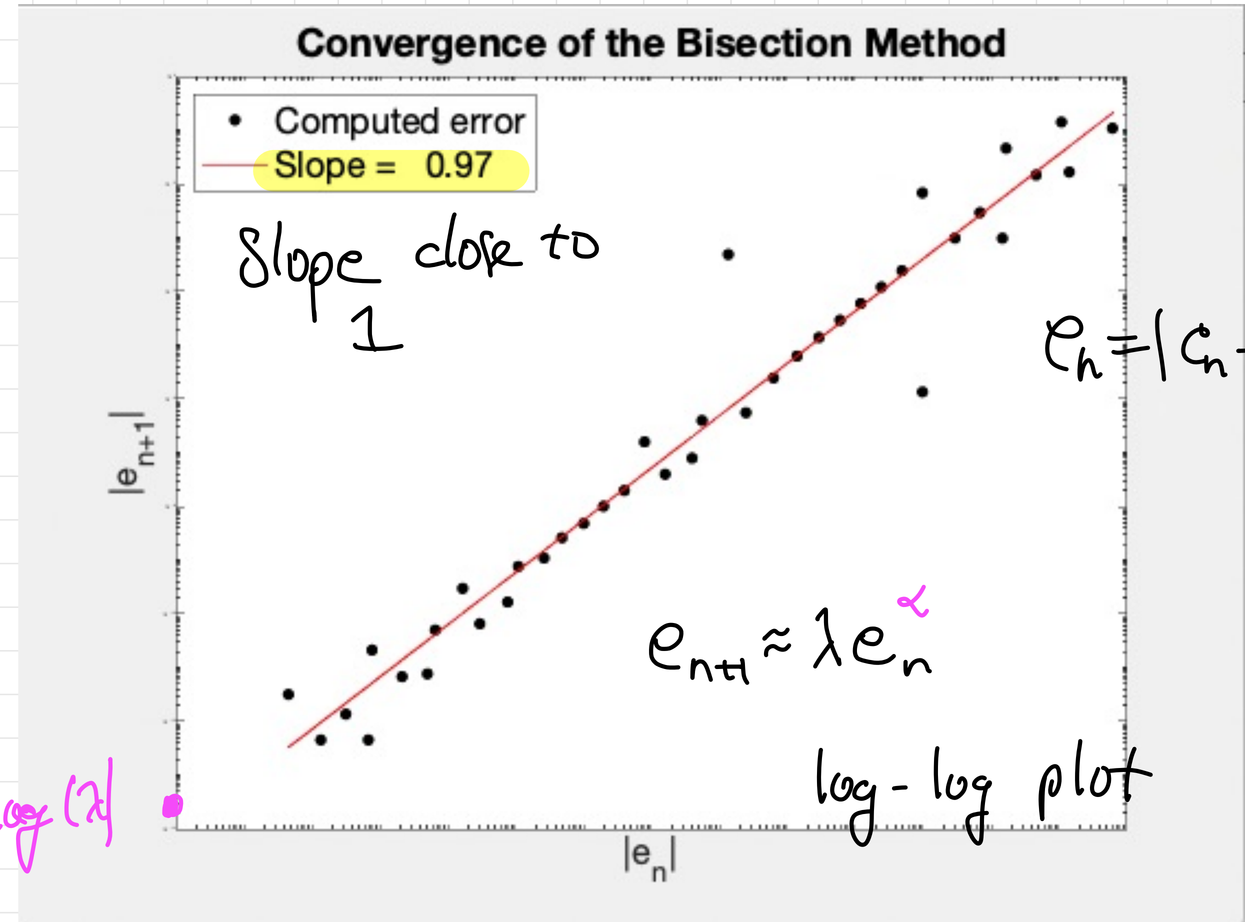
$\lambda =$ constant

So the convergence is linear with asymptotic error constant $\frac{1}{2}$.

$\alpha \neq 1$?

$$\frac{e_{n+1}}{e_n^{\alpha}} = \frac{\frac{1}{2}|b_n - a_n|}{|b_n - a_n|^{\alpha}} = \frac{1}{2}|b_n - a_n|^{1-\alpha}$$

Should be constant! choose $\alpha = 1$



Slope close to 1

$e_n = |c_n - \bar{x}|$

$e_{n+1} \approx \lambda e_n^{\alpha}$

log - log plot

$\log(\lambda)$

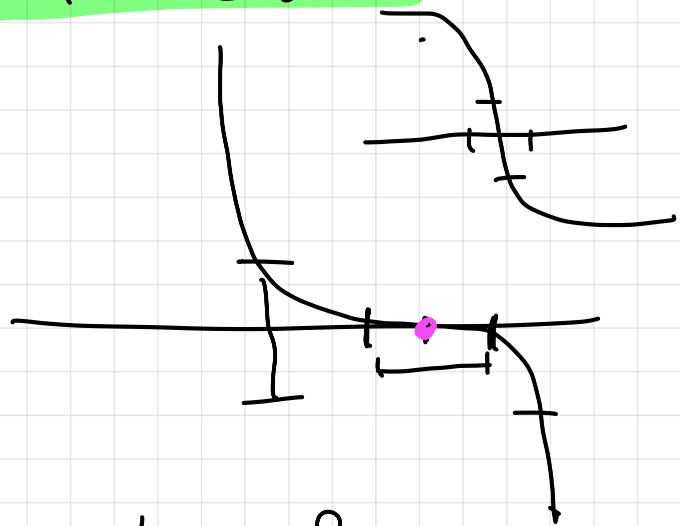
$$\log(e_{n+1}) = \alpha \log(e_n) + \log(\lambda)$$

slope

intercept

```
figure(1);
clf;
e1 = en(1:end-1);
e2 = en(2:end);

p(1) = loglog(e1,e2,'k.','markersize',15);
hold on;
ps = polyfit(log(e1),log(e2),1);
p(2) = loglog(e1,exp(polyval(ps,log(e1))),'r');

lstr{1} = 'Computed error';
lstr{2} = sprintf('Slope = %6.2f',ps(1));

set(gca,'fontsize',2);
title('Convergence of the Bisection Method','fontsize',18);
xlabel('|e_n|','fontsize',16);
ylabel('|e_{n+1}|','fontsize',16);
set(gca,'xtick',0:5:30)

lh = legend(p,lstr,'location','northwest');
set(lh,'fontsize',16);
shg
```

Stopping criteria for Bisection



- $|b_n - a_n| < \varepsilon$
- $|f(c)| < \varepsilon$
- Or, we can fix the number of iterations.

Suppose we want $|c_n - \bar{x}| \le \varepsilon$

$$|c_n - \bar{x}| \le \frac{1}{2^n} |b_0 - a_0| \le \varepsilon$$

So we need to find n so that

$$\frac{1}{2^n} |b_0 - a_0| \le \varepsilon$$

solve for n!
$\varepsilon \ne$ machine eps!

or

$$-n + \log_2(|b_0 - a_0|) \le \log_2(\varepsilon)$$

$$-n \le \log_2\left(\frac{\varepsilon}{|b_0 - a_0|}\right)$$

or

$$n \ge \log_2\left(\frac{|b_0 - a_0|}{\varepsilon}\right)$$

use to get K_{max}

Complete algorithm

```python
beta = 0.1

def g(x):
    return (1/3)*x**3 - x**2 + 4/3*beta

def bisect(f,a,b):
    fa = f(a)
    tol = 1e-8
    kmax = int(log2((b-a)/tol)+1)
    for k in range(kmax):
        c = (a+b)/2;
        fc = f(c)
        if (sign(fa) != sign(fc)):
            b = c
        else:
            a = c
            fa = fc
        k += 1
        if abs(b-a) < tol:
            print("Converged in {:d} iterations".format(k))
            break
    return (a+b)/2

xroot = bisect(g,0,2)
print("x    = {:24.16f}".format(xroot))
print("f(x) = {:24.4e}".format(g(xroot)))
```

f(b) not ever used.

Advantages of Bisection

- very robust
- guaranteed to converge to a root of f(x)
- inexpensive - only one function evaluation per iteration

the end