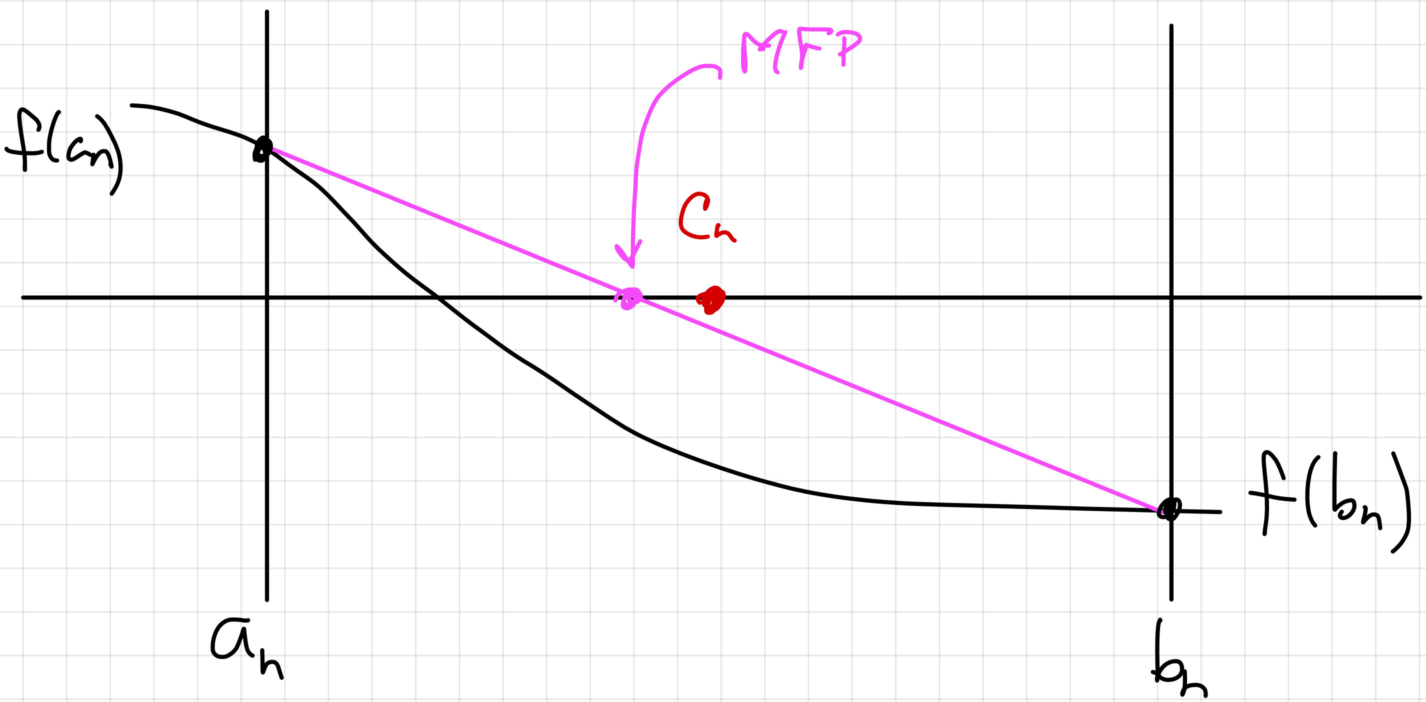
Method of False Position

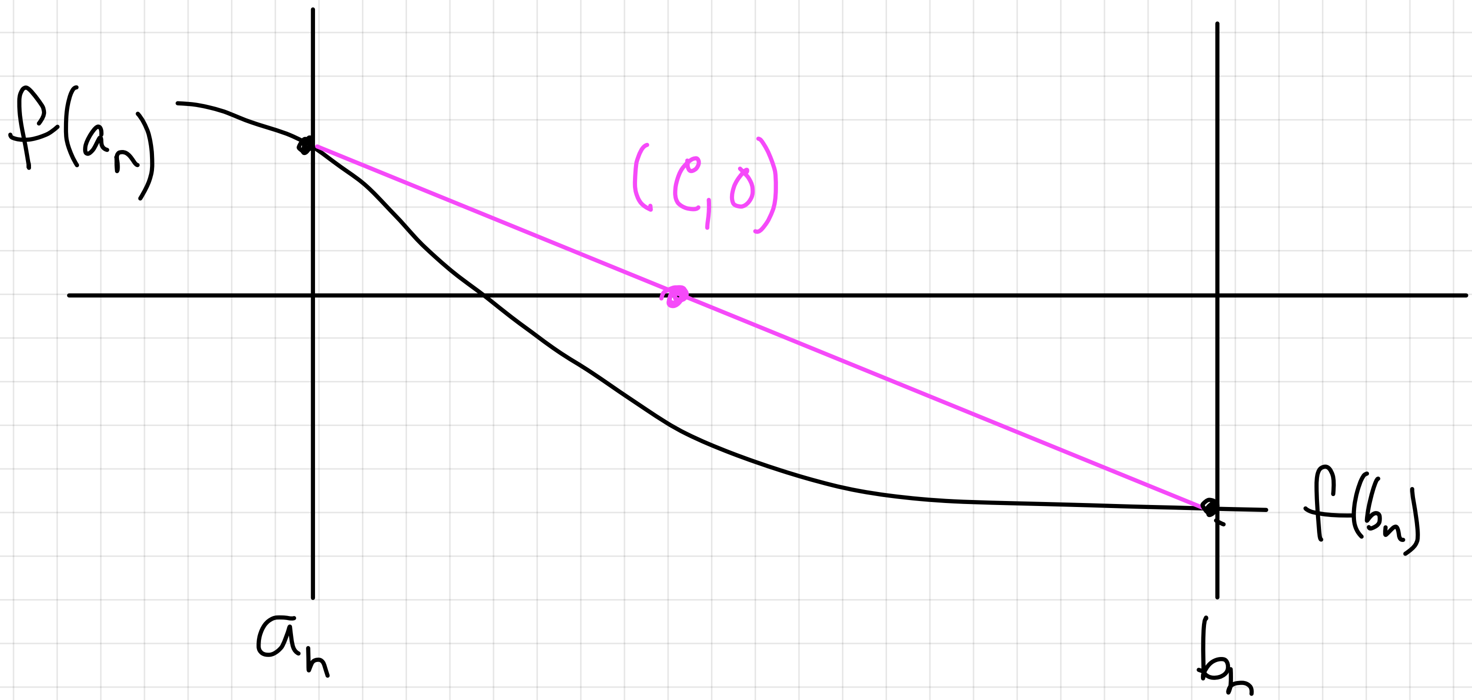
Method of False Position (MFP)

- Bracketing method similar to Bisection, but with better approximation to the root.



c_n - bisection $\qquad$ c_n - MFP

Idea: Approximate curve on $[a_n, b_n]$ using straight line approximation.



Line through the two points $(a_n, f(a_n))$ and $(b_n, f(b_n))$:

$$y = \frac{f(b_n)-f(a_n)}{b_n-a_n}(x-b_n)+f(b_n)$$

The line crosses the x-axis at the point $(c,0)$. We can solve for c to get

$$0 = \frac{f(b_n)-f(a_n)}{b_n-a_n}(c-b_n)+f(b_n)$$

$$\Rightarrow \quad c = b_n - f(b_n)\frac{b_n-a_n}{f(b_n)-f(a_n)}$$

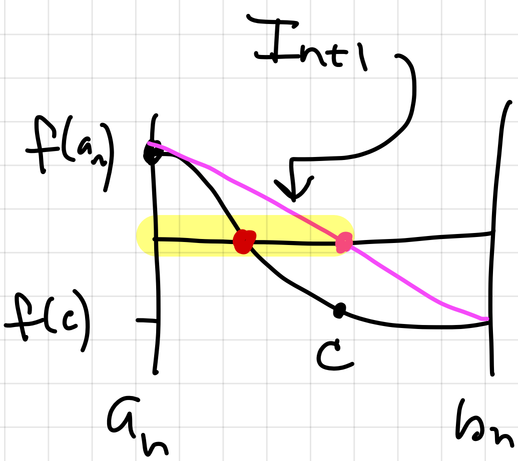
MFP Algorithm

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

1. Compute $\quad c = b_n - \dfrac{f(b_n) - f(a_n)}{b_n - a_n} f(b_n)$

2. Consider 2 cases

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

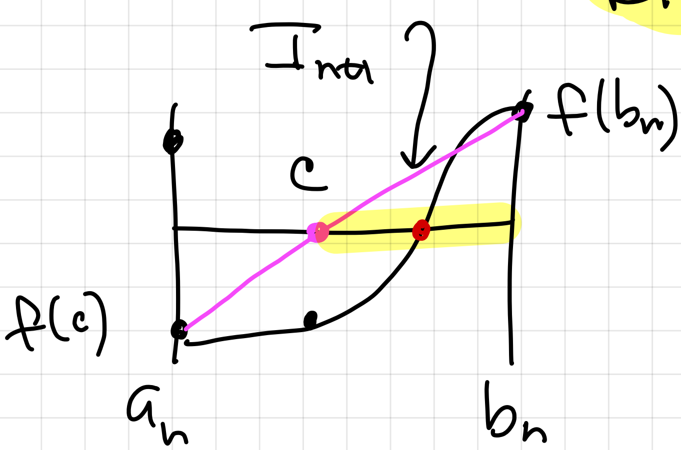


$\Rightarrow [a_n, c]$ contains a root

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

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

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. Stopping criteria (??)

Method of False Position

```python
beta = 0.1

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

def falseposition(f,a,b):
    fa = f(a)
    fb = f(b)
    tol = 1e-8
    kmax = 30
    for k in range(kmax):
        c = b - fb*(b-a)/(fb-fa)
        fc = f(c)
        if (fa*fc < 0):
            b = c
            fb = fc
        else:
            a = c
            fa = fc
        e = abs(b-a)
        if e < tol:
            print("Tolerance reached")
            break
    return c

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

```
x          =         0.3916002113181833
f(x) =                 5.5511e-17
```

Differences between Bisection and MFP

- ~ Same number of function calls as bisection.

Method of False Position

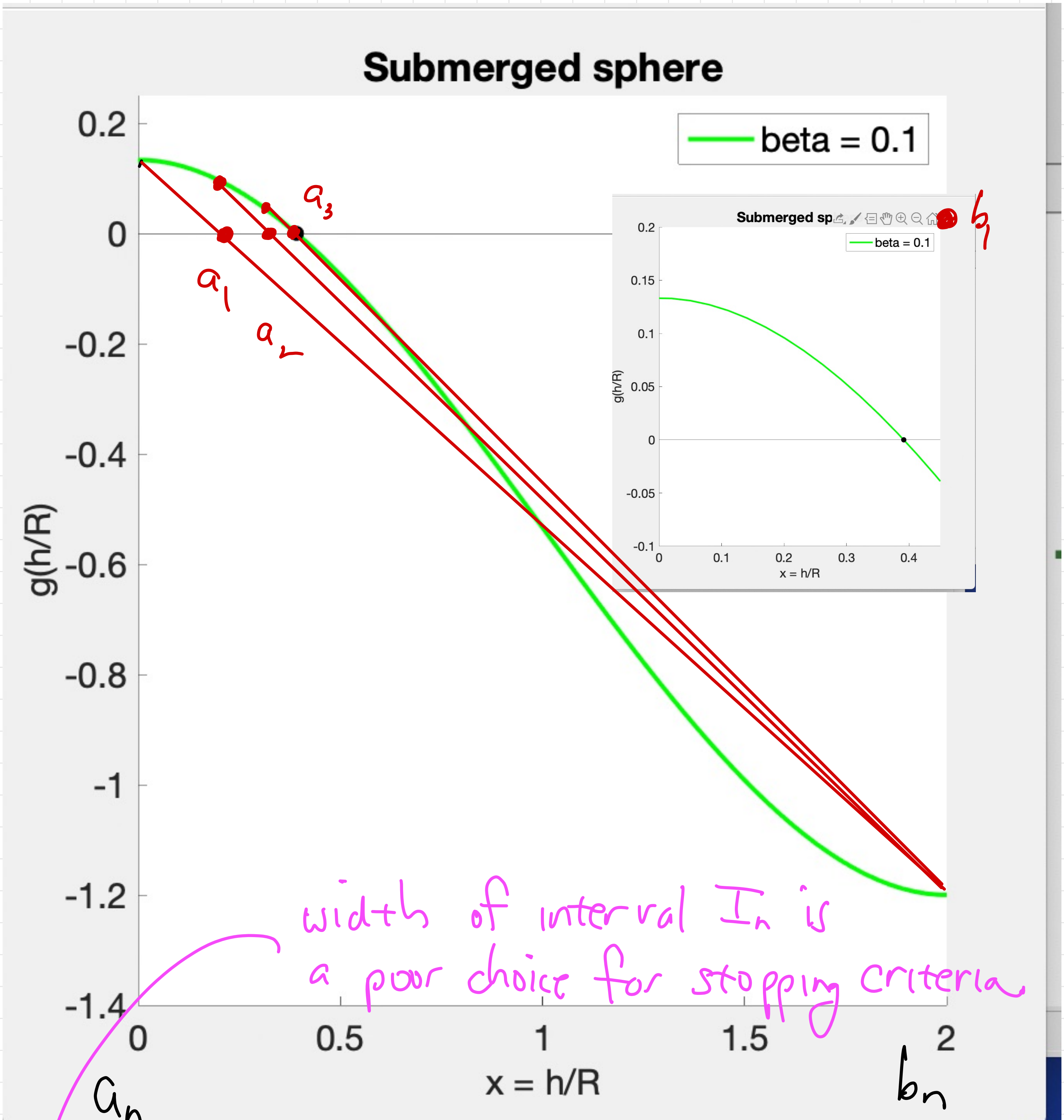
n	a_n	b_n	$\lvert b_n - a_n \rvert$
0	0.2000000000000000	2.0000000000000000	1.8000e+00
1	0.3333333333333333	2.0000000000000000	1.6667e+00
2	0.3799999999999999	2.0000000000000000	1.6200e+00
3	0.3896940418679551	2.0000000000000000	1.6103e+00
4	0.3913005793742759	2.0000000000000000	1.6087e+00
5	0.3915534653144359	2.0000000000000000	1.6084e+00
6	0.3915929270817553	2.0000000000000000	1.6084e+00
7	0.3915990764566473	2.0000000000000000	1.6084e+00
8	0.3916000345153816	2.0000000000000000	1.6084e+00
9	0.3916001837737750	2.0000000000000000	1.6084e+00
10	0.3916002070269964	2.0000000000000000	1.6084e+00
11	0.3916002106496526	2.0000000000000000	1.6084e+00
12	0.3916002112140322	2.0000000000000000	1.6084e+00
13	0.3916002113019574	2.0000000000000000	1.6084e+00
14	0.3916002113156558	2.0000000000000000	1.6084e+00
15	0.3916002113177897	2.0000000000000000	1.6084e+00
16	0.3916002113181221	2.0000000000000000	1.6084e+00
17	0.3916002113181738	2.0000000000000000	1.6084e+00
18	0.3916002113181818	2.0000000000000000	1.6084e+00
19	0.3916002113181833	2.0000000000000000	1.6084e+00
20	0.3916002113181833	2.0000000000000000	1.6084e+00
21	0.3916002113181833	2.0000000000000000	1.6084e+00
22	0.3916002113181833	2.0000000000000000	1.6084e+00
23	0.3916002113181833	2.0000000000000000	1.6084e+00
24	0.3916002113181833	2.0000000000000000	1.6084e+00
25	0.3916002113181833	2.0000000000000000	1.6084e+00
26	0.3916002113181833	2.0000000000000000	1.6084e+00
27	0.3916002113181833	2.0000000000000000	1.6084e+00
28	0.3916002113181833	2.0000000000000000	1.6084e+00
29	0.3916002113181833	2.0000000000000000	1.6084e+00

```
x            =       0.3916002113181833
x (bisect) =       0.3916002112964634
f(x) =                 5.5511e-17
```

convergence?

- $\lvert b_n - a_n \rvert$ does not go to zero?
- left endpoint appears to converge to the root.
- Right end point doesn't move

Convergence of MFP



width of interval I_n is a poor choice for stopping criteria

- right endpoint never moves
- left endpoint converges to the root
- $|b_n - a_n| \sim |\bar{x} - b_n| \approx 1.6$

Convergence Analysis

Iteration:

$$c_n = b_n - f(b_n)\frac{b_n - a_n}{f(b_n) - f(a_n)}$$

Let the root be given by $\bar{x}$. Then

$$c_n - \bar{x} = b_n - \bar{x} - f(b_n)\frac{b_n - a_n}{f(b_n) - f(a_n)}$$

Expand $f(a_n)$, $f(b_n)$ about $\bar{x}$:

$$f(a_n) \approx \underbrace{\cancel{f(\bar{x})}}_{=0} + f'(\bar{x})(a_n - \bar{x}) + \frac{1}{2}f''(\bar{x})(a_n - \bar{x})^2$$

$$f(b_n) \approx \underbrace{\cancel{f(\bar{x})}}_{=0} + f'(\bar{x})(b_n - \bar{x}) + \frac{1}{2}f''(\bar{x})(b_n - \bar{x})^2$$

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

$$= (b_n - a_n)\left[f'(\bar{x}) + \frac{1}{2}f''(\bar{x})(a_n + b_n - 2\bar{x})\right]$$

$$= (b_n - a_n)\left[f'(\bar{x}) + \frac{1}{2}f''(\bar{x})(a_n + b_n - 2\bar{x})\right]$$

$$c_n - \bar{x} = b_n - \bar{x} - f(b_n)\frac{b_n - a_n}{f(b_n) - f(a_n)}$$

$$\approx b_n - \bar{x} - \left[f'(\bar{x})(b_n - \bar{x}) + \frac{1}{2}f''(\bar{x})(b_n - \bar{x})^2\right]$$

$$\times \left[\frac{\cancel{b_n - a_n}}{\cancel{(b_n - a_n)}\left[f'(\bar{x}) + \frac{1}{2}f''(\bar{x})(a_n + b_n - 2\bar{x})\right]}\right]$$

$$= (b_n - \bar{x})\left[1 - \frac{f'(\bar{x}) + \frac{1}{2}f''(\bar{x})(b_n - \bar{x})}{f'(\bar{x}) + \frac{1}{2}f''(\bar{x})(a_n + b_n - 2\bar{x})}\right]$$

... some algebra

$$= (b_n - \bar{x})(a_n - \bar{x})\left[\frac{f''(\bar{x})}{2f'(\bar{x}) + f''(\bar{x})(a_n + b_n - 2\bar{x})}\right]$$

let b_n be the fixed endpoint. Then

$$e_{n-1} = |a_n - \bar{x}|, \quad e_n = |c_n - \bar{x}|$$

a_{n+1}

$$\underbrace{c_n - \bar{x}}_{e_n} = (b_n - \bar{x})\underbrace{(a_n - \bar{x})}_{e_{n-1}} \left[\frac{f''(\bar{x})}{2f'(\bar{x}) + f''(\bar{x})(a_n + b_n - 2\bar{x})} \right]$$

≈ 1.6 in example

$$e_n = \left[\frac{(b_n - \bar{x}) f''(\bar{x})}{2f'(\bar{x}) + f''(\bar{x})(a_n - \bar{x} + b_n - \bar{x})} \right] e_{n-1}$$

Let $\ell = b_n - \bar{x} \Rightarrow$ remains fixed > 0

Note: $a_n - \bar{x} \ll 1$, so $(\underbrace{b_n - \bar{x}}_{\ell \sim \Theta(1)} + \underbrace{a_n - \bar{x}}_{\varepsilon}) \approx \ell$

Define*

$$\boxed{\lambda \equiv \frac{\ell f''(\bar{x})}{2f'(\bar{x}) + f''(\bar{x})\ell}} \sim \text{constant (ind. of } n)$$

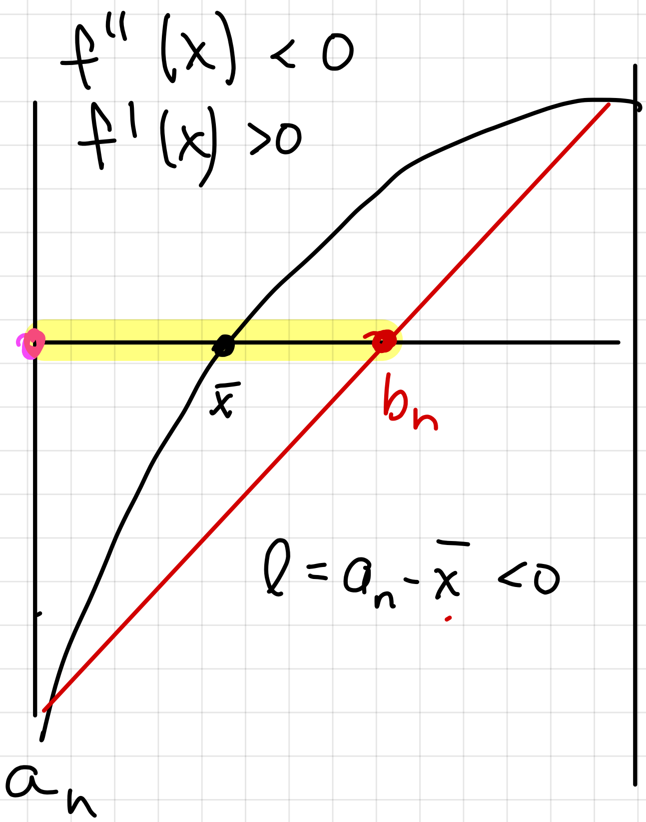
We have $e_n \approx \lambda e_{n-1}$, which converges if $|\lambda| < 1$. **Show that $|\lambda| < 1$.**

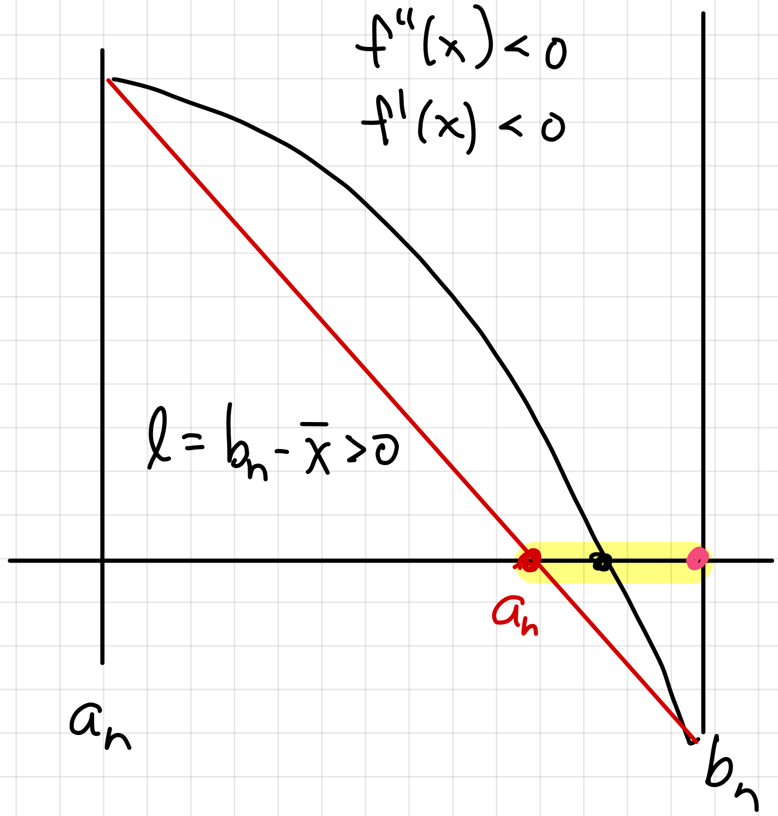
Consider four possible configurations.

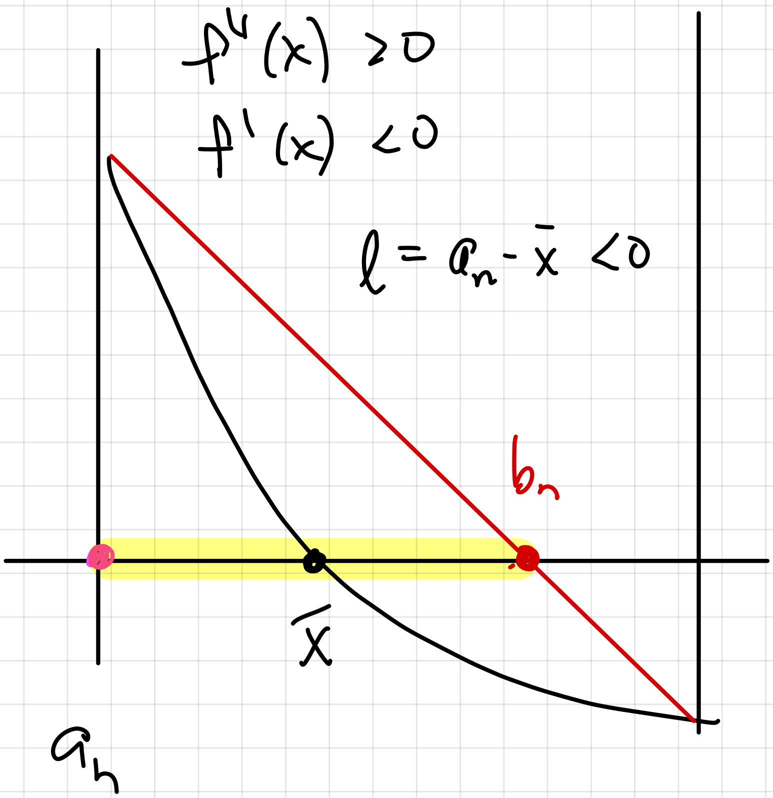
* Actual λ may be smaller
will depend on initial interval.

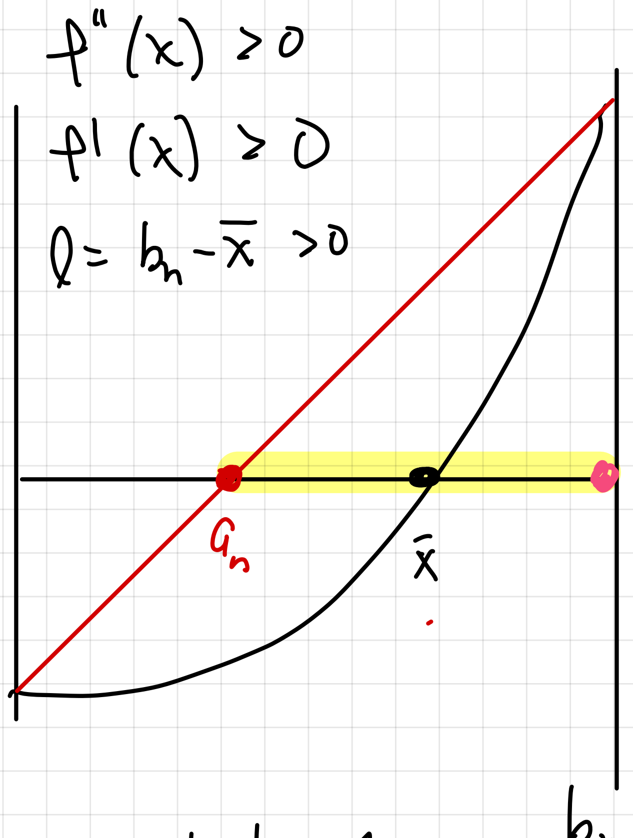
→

Four possible configurations









Homework: Show $|\lambda| < 1$

Convergence Analysis

Method of False Position

$$e_n = \lambda e_{n-1}, \quad |\lambda| < 1$$

Convergence is linear with asymptotic error constant

$$\lambda = \frac{\ell f''(\bar{x})}{2f'(\bar{x}) + \ell f''(\bar{x})}$$

Question : error constant depends on $f(x)$. What happens if $f'(\bar{x}) = 0$?

Question : We still don't have good stopping criteria.

- $|b_n - a_n|$ is no good — (might not go to 0)
- Can't use $|c_n - \bar{x}|$ (don't know $\bar{x}$)

Stopping Criteria
Method of False Position

- Need an error estimate:

$$e_n = c_n - \bar{x}$$
$$= c_n - c_{n-1} + c_{n-1} - \bar{x}$$
$$= c_n - c_{n-1} + e_{n-1}$$
$$= c_n - c_{n-1} + \frac{1}{\lambda} e_n$$

$e_n \approx \lambda e_{n-1}$

$$\left(1 - \frac{1}{\lambda}\right) e_n = c_n - c_{n-1}$$
$$e_n = \frac{\lambda}{\lambda - 1}(c_n - c_{n-1})$$
$$\underbrace{|e_n| = \left|\frac{\lambda}{\lambda - 1}\right| |c_n - c_{n-1}|}$$

interesting, but not useful. What is λ?

$$|e_n| = \left|\frac{\lambda}{\lambda-1}\right| (c_n - c_{n-1})$$

Estimate λ:

$$\lambda = \frac{c_n - c_{n-1}}{c_{n-1} - c_{n-2}} \approx \frac{(c_n - \bar{x}) - (c_{n-1} - \bar{x})}{(c_{n-1} - \bar{x}) - (c_{n-2} - \bar{x})}$$

$$= \frac{e_n - e_{n-1}}{e_{n-1} - e_{n-2}}$$

$e_n \approx \lambda e_{n-1}$

$$\frac{e_n - e_{n-1}}{e_{n-1} - e_{n-2}} = \frac{(\lambda - 1) e_{n-1}}{(1 - \frac{1}{\lambda}) e_{n-1}} = \lambda$$

Use approximation for λ. Then stopping criteria:

$$|e_n| = \left|\frac{\lambda}{\lambda-1}\right| |c_n - c_{n-1}| < \varepsilon$$

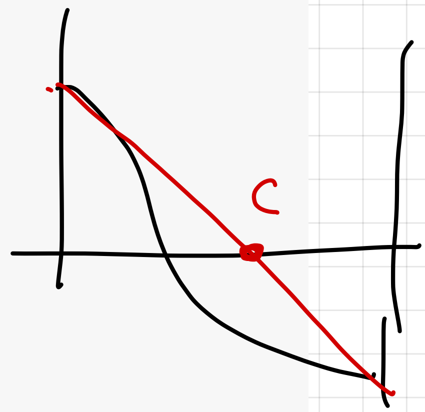
Requires 3 successive iterates

c_n, c_{n-1}, c_{n-2}

Complete Algorithm - MFP

```python
def falseposition(f,a,b):
    fa = f(a)
    fb = f(b)
    tol = 1e-12
    kmax = 50
    ckm1 = ckm2 = 0
    error = [0]*kmax
    for k in range(kmax):
        ck = b - fb*(b-a)/(fb-fa)
        fc = f(ck)
        if (fa*fc < 0):
            fb = fc
            b = ck
        else:
            fa = fc
            a = ck
        if k > 2:
            l = (ck-ckm1)/(ckm1-ckm2)
            error[k] = abs(l/(l-1)*(ck-ckm1))
        else:
            error[k] = abs(b-a);
        print('{:5d} {:20.8e}'.format(k,error[k]))
        if error[k] < tol:
            print("Tolerance reached")
            break
        ckm2 = ckm1
        ckm1 = ck
    return ck, error

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

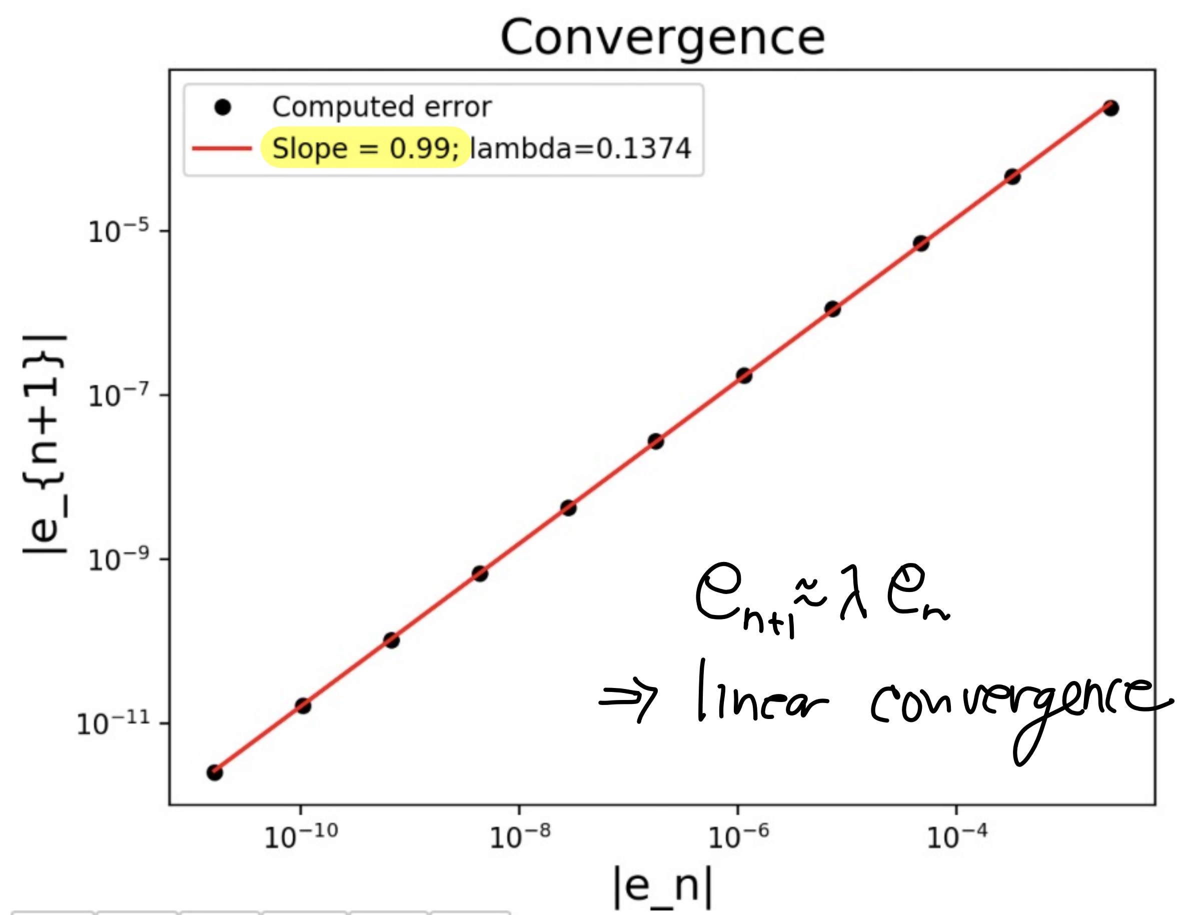


estimated error.

```
    0      1.80000000e+00
    1      1.66666667e+00
    2      1.62000000e+00
    3      2.54173049e-03
    4      3.19129690e-04
    5      4.72435450e-05
    6      7.29641380e-06
    7      1.13515748e-06
    8      1.76809986e-07
    9      2.75445829e-08
   10      4.29119134e-09
   11      6.68530816e-10
   12      1.04151600e-10
   13      1.62257800e-11
   14      2.52800040e-12
   15      3.93730762e-13
Tolerance reached
x    =       0.3916002113177897
f(x) =               2.4802e-13
```

estimated error

MFP – Convergence



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

$\Rightarrow$ linear convergence

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

```
%matplotlib notebook
%pylab
```

```
figure(1)
clf()

e1 = error[3:kfinal-1]
e2 = error[4:kfinal]

loglog(e1,e2,'k.',markersize=10,label='Computed error')
ps = polyfit(log(e1),log(e2),1)
loglog(e1,exp(polyval(ps,log(e1))),'r', \
        label='Slope = {:.2f}; lambda={:.4f}'.format(ps[0],exp(ps[1])))
title('Convergence',fontsize=18)
xlabel('|e_n|',fontsize=16)
ylabel('|e_{n+1}|',fontsize=16)
legend()

```

Method of False Position vs. Bisection

Similarities

- Both are "bracketing methods"
- Both converge linearly
- Both require one function call per iteration.

Differences:

- asymptotic error constants

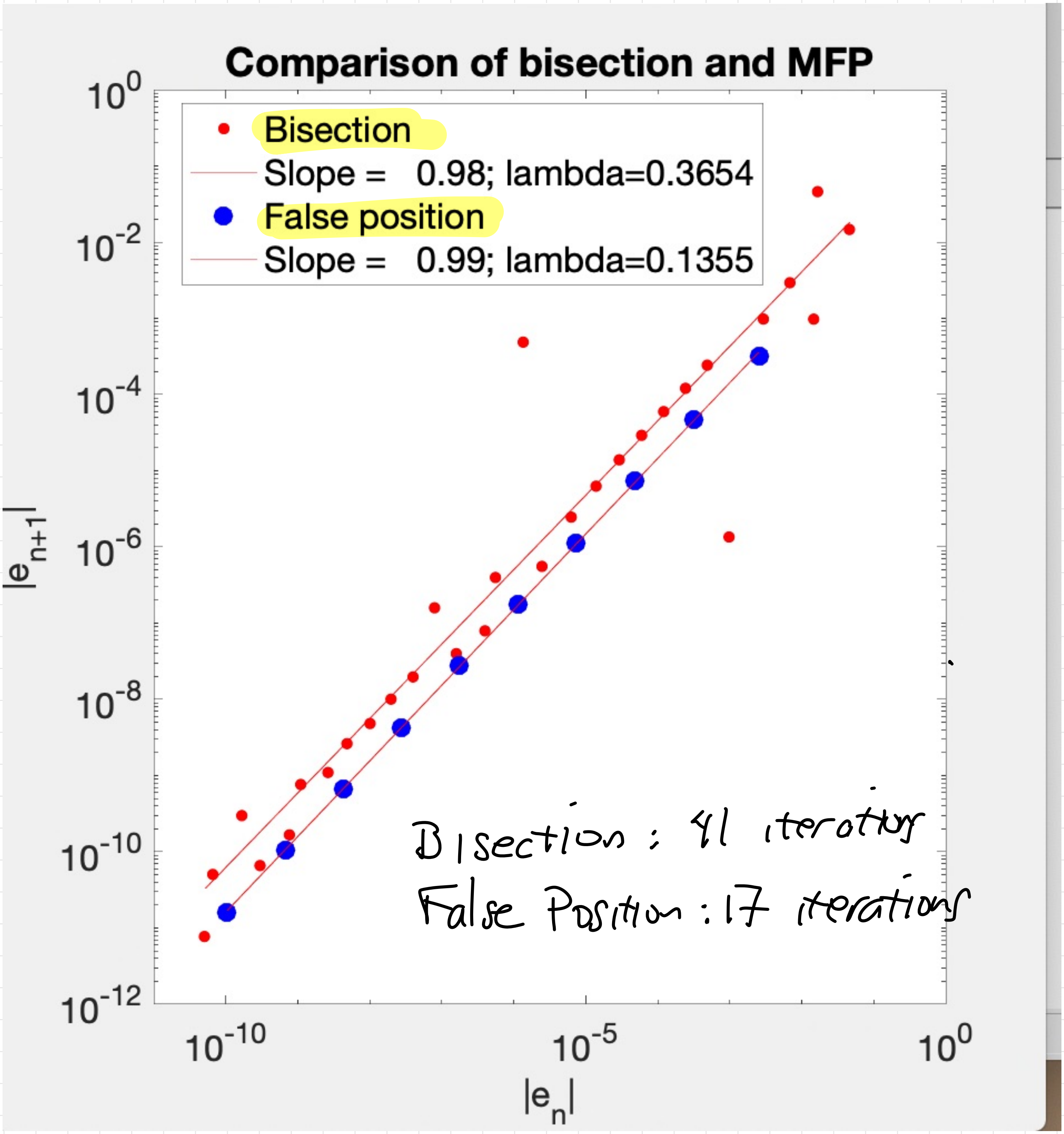
  Bisection : $\lambda = \frac{1}{2}$

  MFP : $\lambda \sim \dfrac{\ell f''(\bar{x})}{2f'(\bar{x}) + \ell f''(\bar{x})}$

  $$\ell \sim \begin{cases} a_n - c_n, & a_n \text{ is fixed} \\ b_n - c_n, & b_n \text{ is fixed} \end{cases}$$

  MFP could be slower or faster
- MFP has smoother convergence

Comparison



- MFP shows much more relidble convergence
- Bisection requires more iterations.

the end