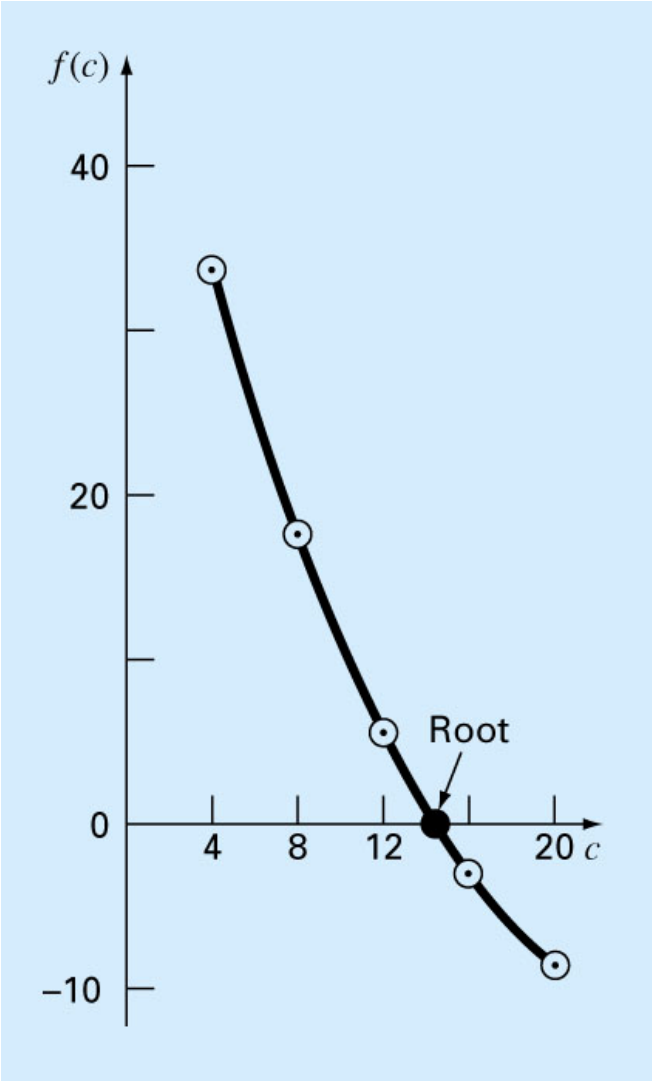
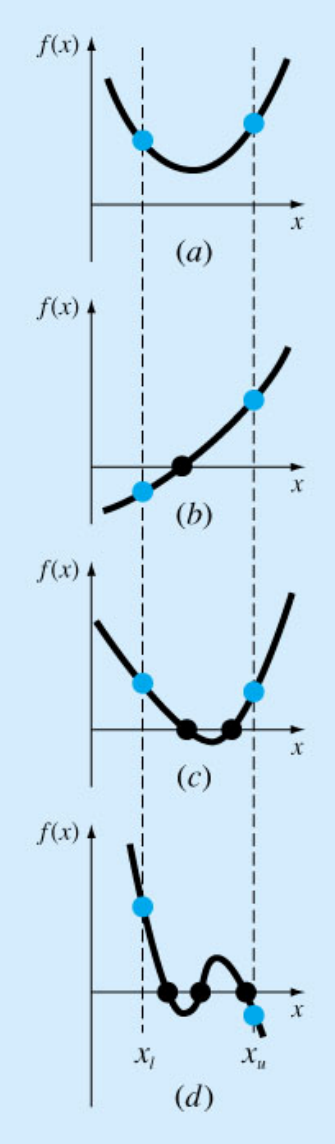
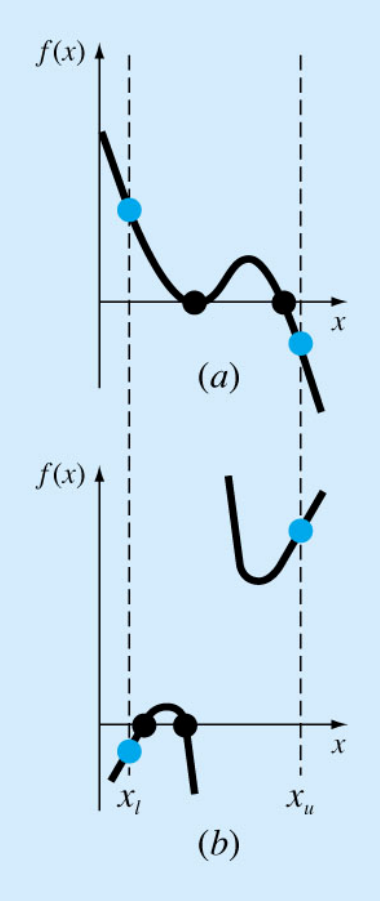
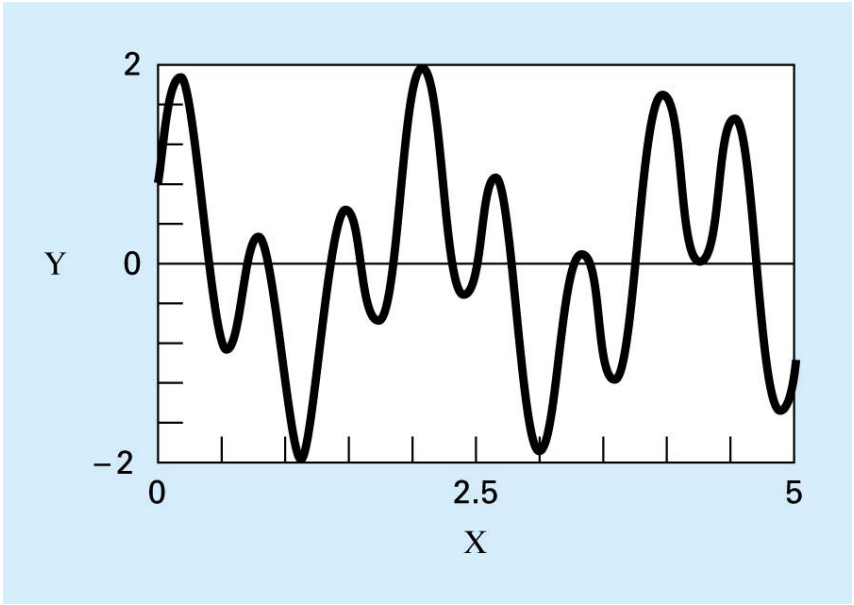
Root Finding

Determine the value of x that makes f(x)=0





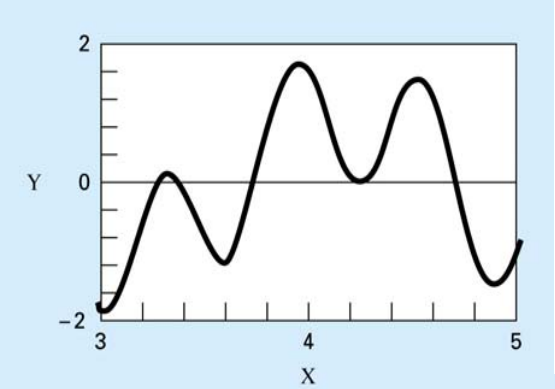


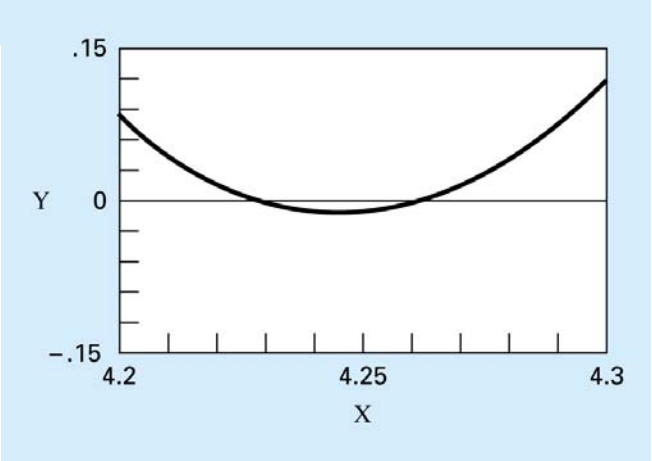


Plot Your Function and see the general behavior:

Use this plot to provide an initial guess orfor potential problems.

Refine your Interval



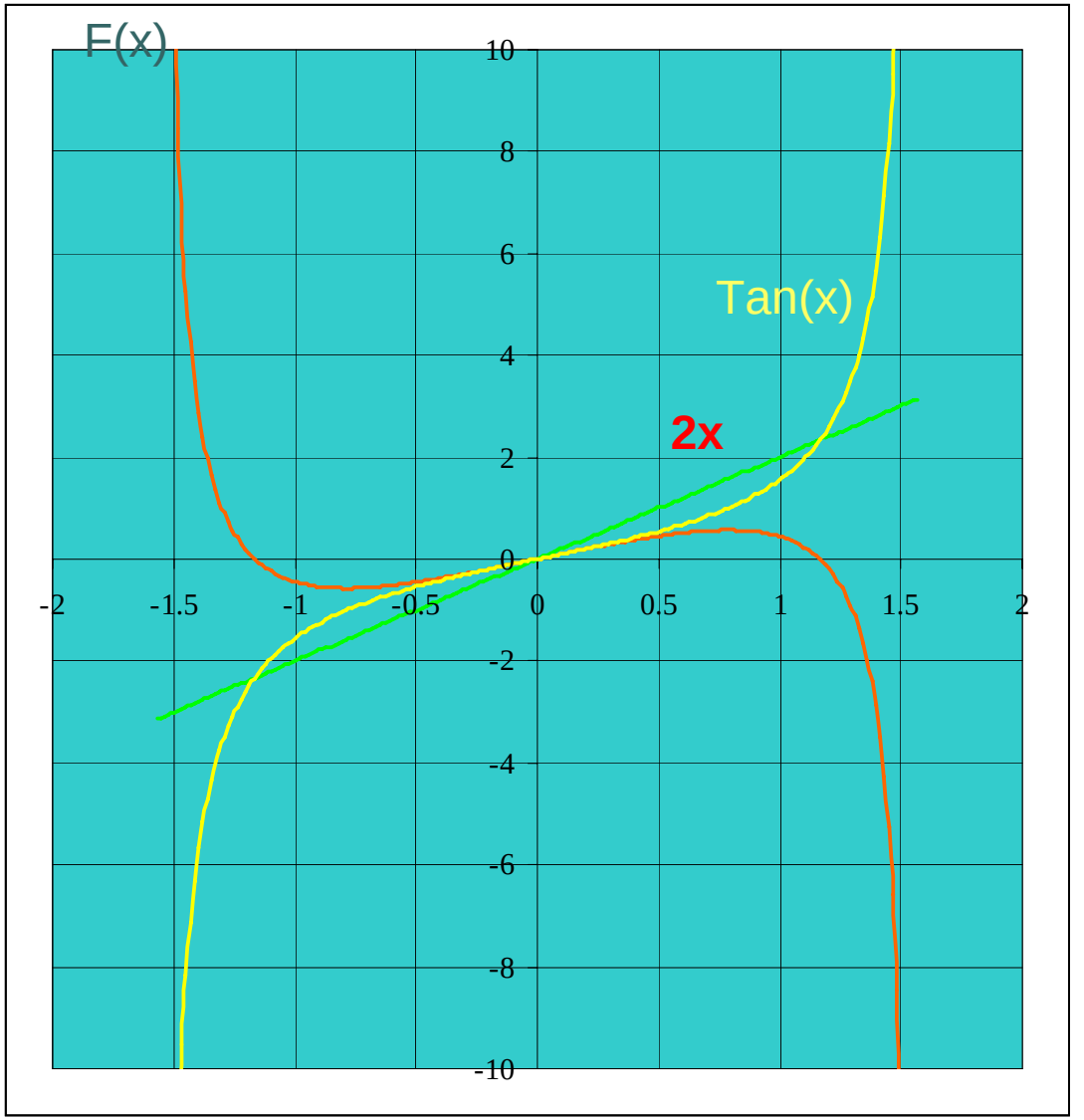


1. How often will this function be evaluated?
2. How much precission is needed?
3. How fast and robust must be the method?
4. Does the function have singularities?

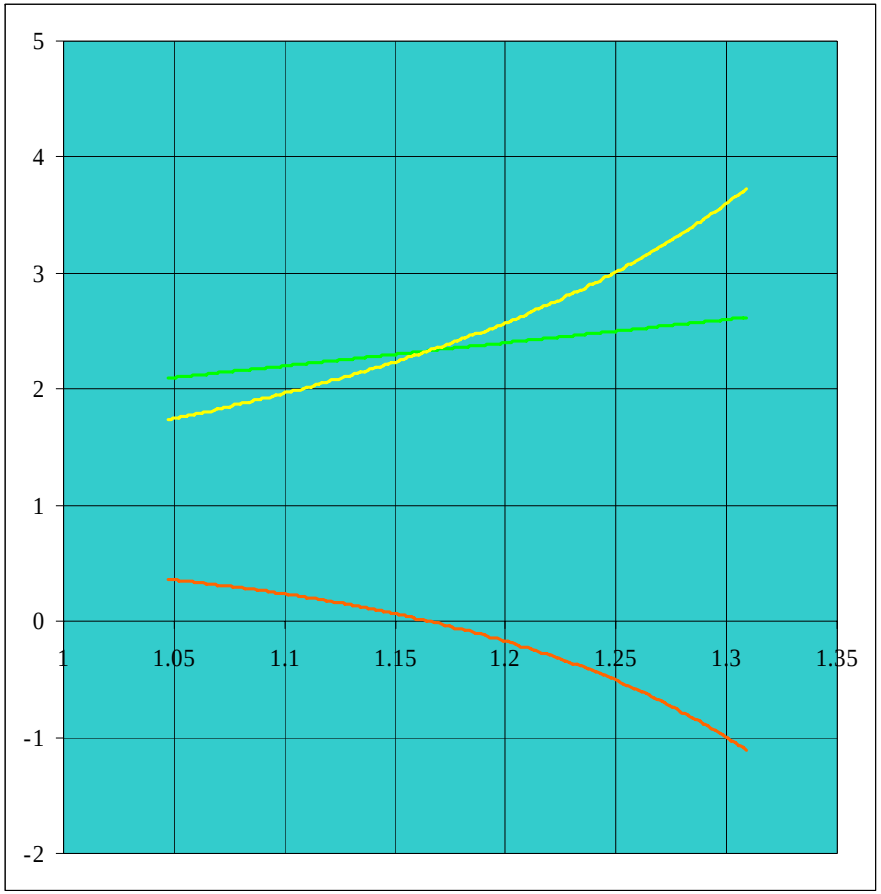
Example

$F(x) = 2x - \tan(x)$

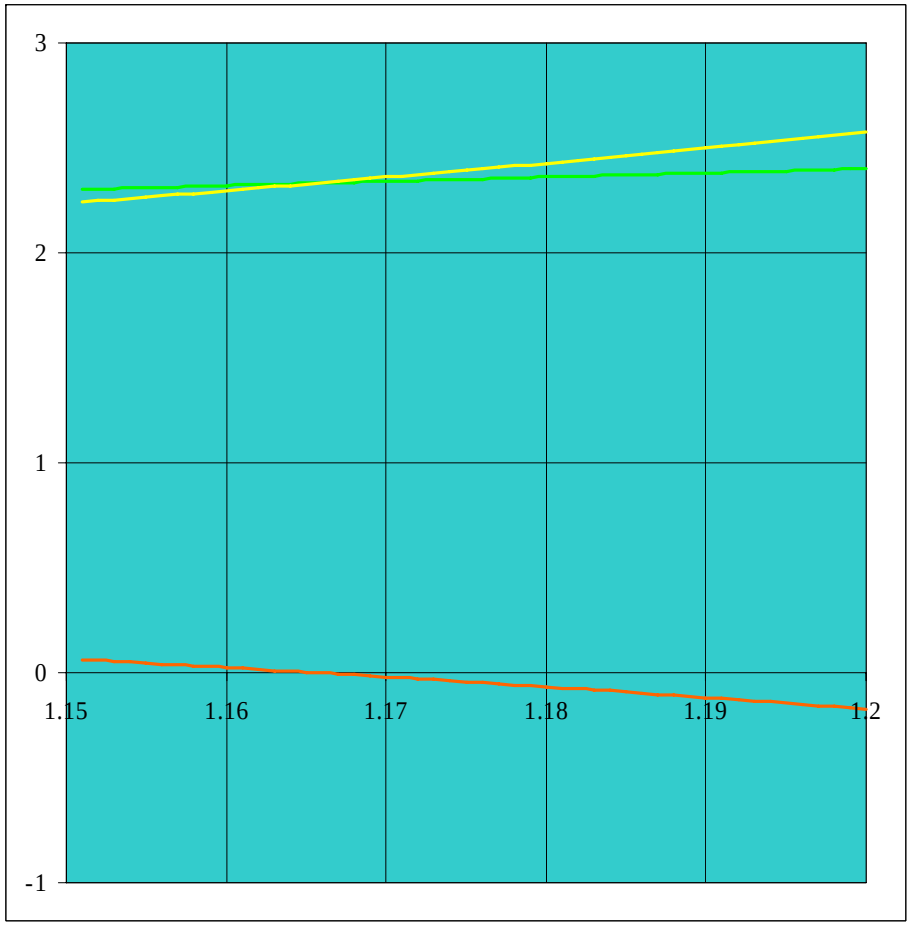
Root $1 < x_r < 1.5$

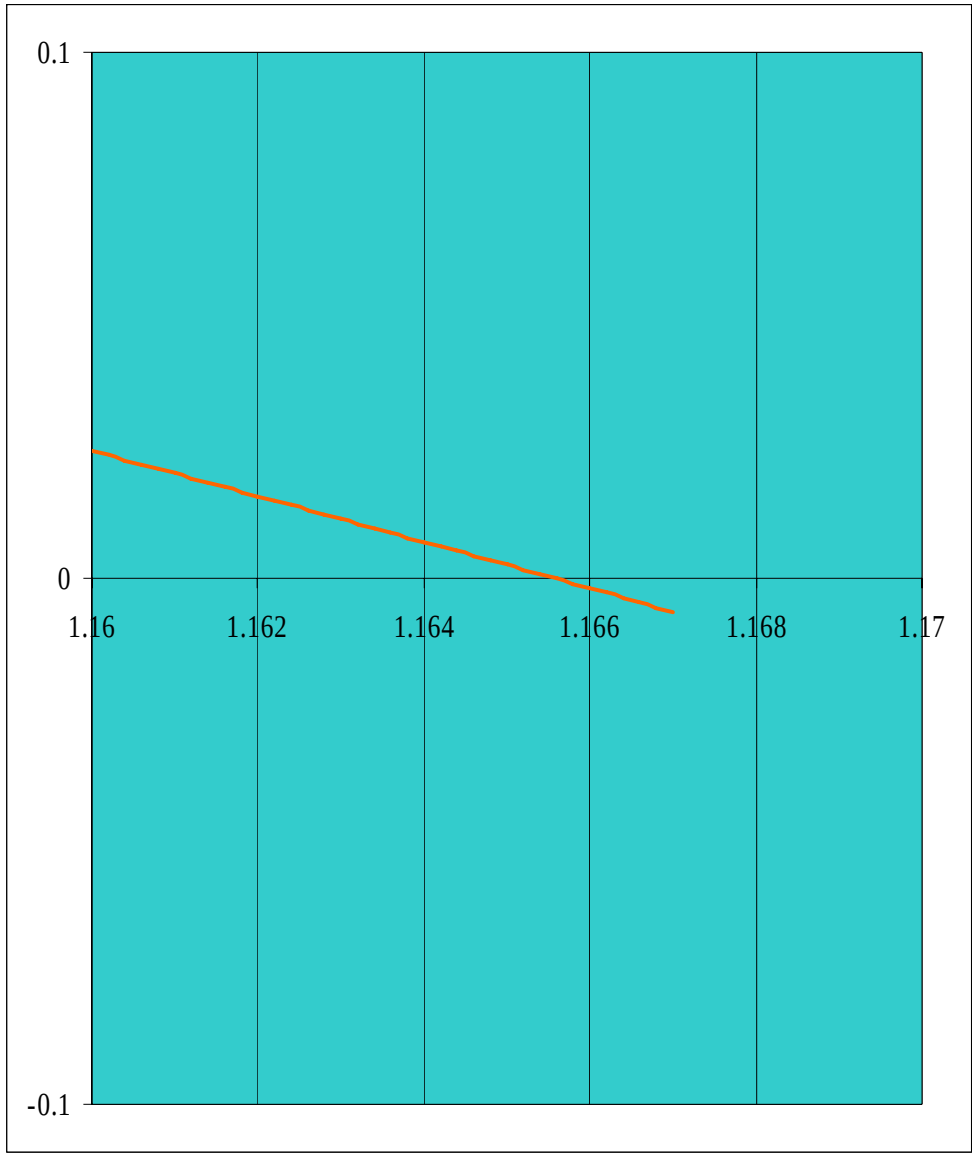


Root 1.15 < xr <1.20



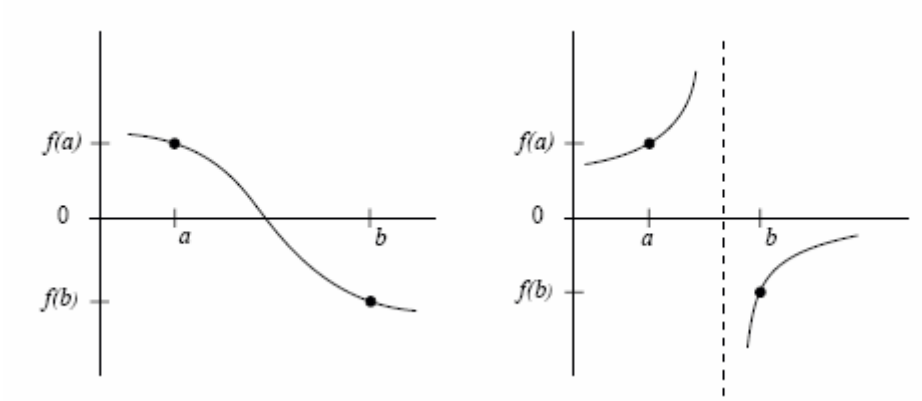
Root 1.16 < x_r <1.17





Xr=1.1655 ±0.0001

- Bracketing Methods
  - The root is bracketed on the interval $[x_l, x_u]$, if $f(x_l)$ and $f(x_u)$ have opposite sign. (be careful for the singularities)

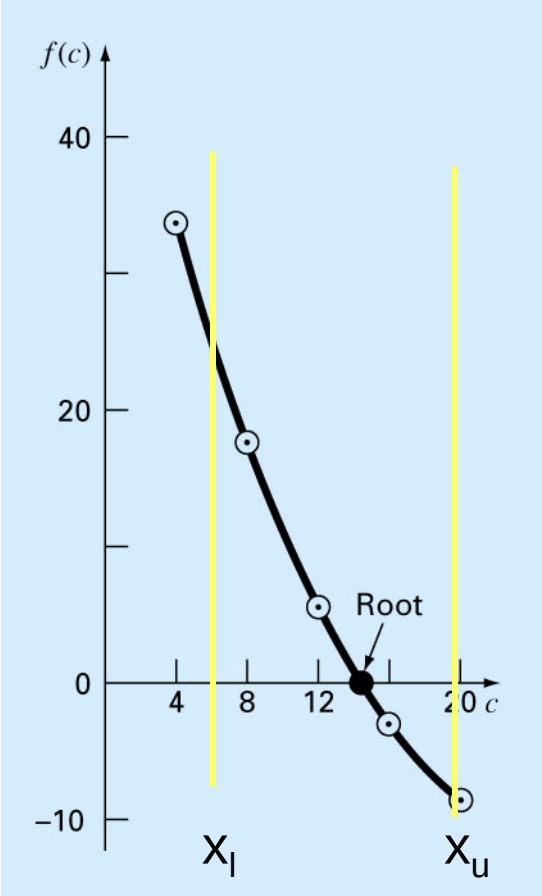


- Open Methods
  - Using an initial guess, refine your guess to reach for the root

Bracketing Methods

Test for sign change:

$f(x_l)*f(x_u)<0$



Bisection Method

Binary Chopping, Interval Halving, Bolzano Method

Step1: Chose lower x_l and upper x_u guesses for the root such that the function changes sign over the interval. THis can be checked by ensuring $f(x_l)f(x_u)<0$

Step2: An estimate of the root x_r is determined by:

$$1. \quad x_r = \frac{(x_u + x_l)}{2}$$

Step 3: Make the following evaluations to determine in which subinterval the root lies:

a) if $f(x_l)f(x_r)<0$, the root lies in the lower subinterval. Therefore set $x_u=x_r$, $f(x_u)=f(x_r)$ and return to step 2.

b) if $f(x_l)f(x_r)>0$, the root lies in the upper subinterval. Therefore set $x_u=x_r$, $f(x_l)=f(x_r)$ and return to step 2.

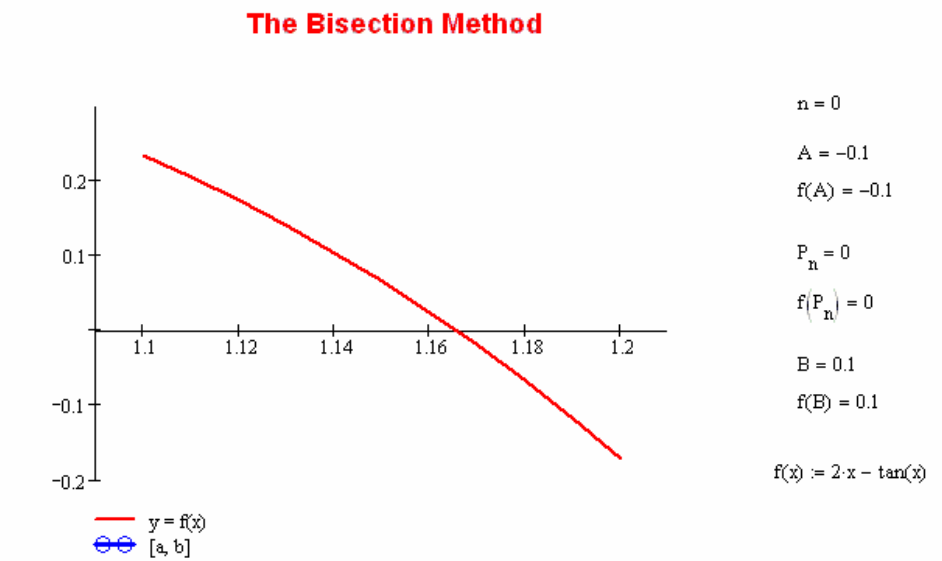
c) if $f(x_l)f(x_r)=0$ the root is x_r; terminate the computation

Example F(x)=2x-tan(x)

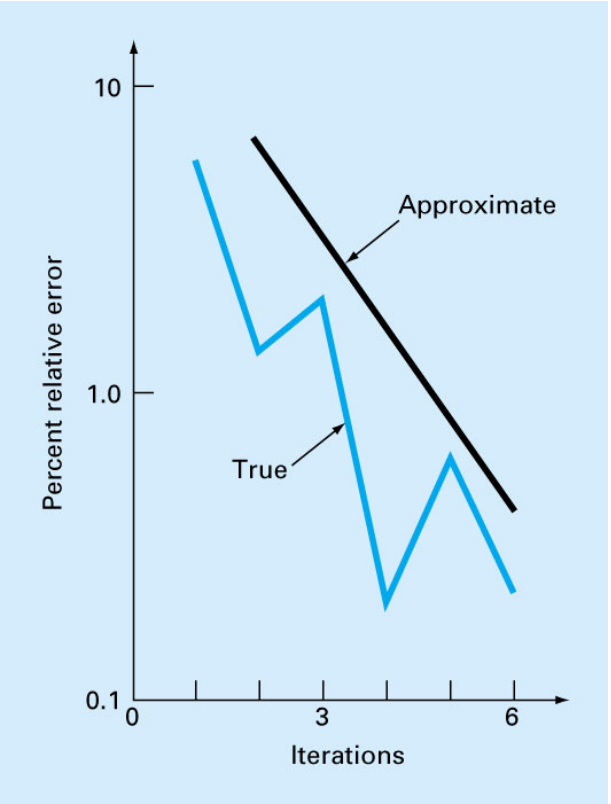
F(1.1)= 0.23524

F(1.2)= -0.17215

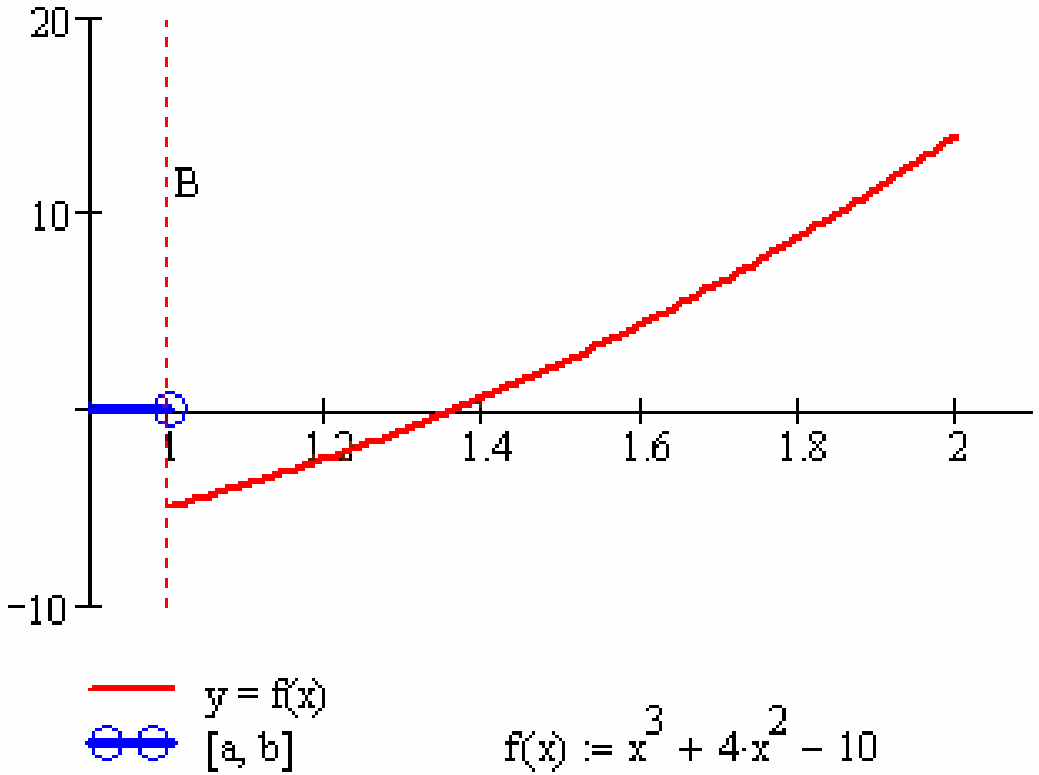
Using Bisection Method



	F(x)	$\%\varepsilon_a$
F(1.15)	0.065503051	4.347826
F(1.175)	-0.043221152	2.12766
F(1.1625)	0.013434203	1.075269
F(1.16875)	-0.014293352	0.534759
F(1.165625)	-0.000283024	0.268097
F(1.1640625)	0.006611804	0.134228
F(1.16484375)	0.003173496	0.067069
F(1.165234375)	0.001447519	0.033523
F(1.165429688)	0.000582819	0.016759
F(1.1655273344)	0.00015004	0.008379
F(1.165576172)	-6.64562E-05	0.004189



The Bisection Method



$n = 0$

$A = -1$

$f(A) = -7$

$P_n = 0$

$f(P_n) = -10$

$B = 1$

$f(B) = -5$

For a given bound of Error, $E_{ad,}$ one can estimate the number of steps , n, required to reach a solution whose error is less than the given bound.

Let $E_{ad} = E^n_a$ and determine n

Convergence does not depend on the function. Error on x after n iterations is known.

$$x_r^{new} - x_r^{old} = \frac{x_u - x_l}{2}$$

$$x_r^{new} = \frac{x_u + x_l}{2}$$

$$\varepsilon_a = \left| \frac{x_u - x_l}{x_u + x_l} \right|$$

$$E_a^0 = x_u^0 - x_l^0 = \Delta x^0$$

$$E_a^1 = \frac{\Delta x^0}{2}$$

$$E_a^n = \frac{\Delta x^0}{2^n}$$

$$2^n = \frac{\Delta x^0}{E_a^n}$$

$$n = \log_2 \left(\frac{\Delta x^0}{E_a^n} \right)$$

Pseudo Code for Bisection Method

```
Function Bisect(xl, xu, eps, imax, xr, iter, ea)
    xr = xu
    iter = 0#
    fl = Fun(xl)
    Do
        xrold = xr
        xr = (xl + xu) / 2
        fr = Fun(xr)
        iter = iter + 1
        If xr <> 0 Then
            ea = Abs((xr - xrold) / xr) * 100
        End If
        Test = fl * fr
        If Test < 0 Then
            xu = xr: fu = fr
            Else
            xl = xr: fl = fr
        End If
        if ea>eps or iter>imax Exit
    End Do
Bisect = xr
End Bisect
```

i.e. For the root of
f(x)=2x+tanx

```
Function Fun(x)
Fun = 2 * x - Tan(x)
End Fun
```

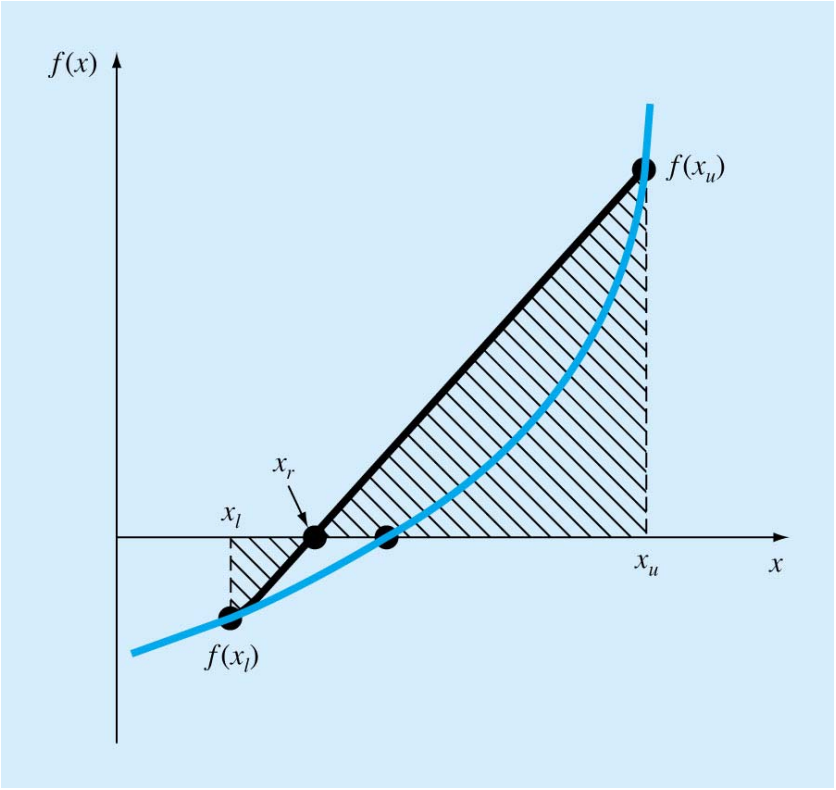
Convergence Critieria

1. $\left|x_k - x_{k-1}\right| < E_a$
2. $\frac{\left|x_k - x_{k-1}\right|}{x_k} * 100 < \%\varepsilon_a$
3. $f(x_k) < \varepsilon$

1. $\left|x_k - x_{k-1}\right| < E_a$

False Position Method

(linear interpolation, Regula-falsi)



Using Similar triangles:

$$\frac{f(x_l)}{x_r - x_l} = \frac{f(x_u)}{x_r - x_u}$$

Solve for x_r

$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)}$$

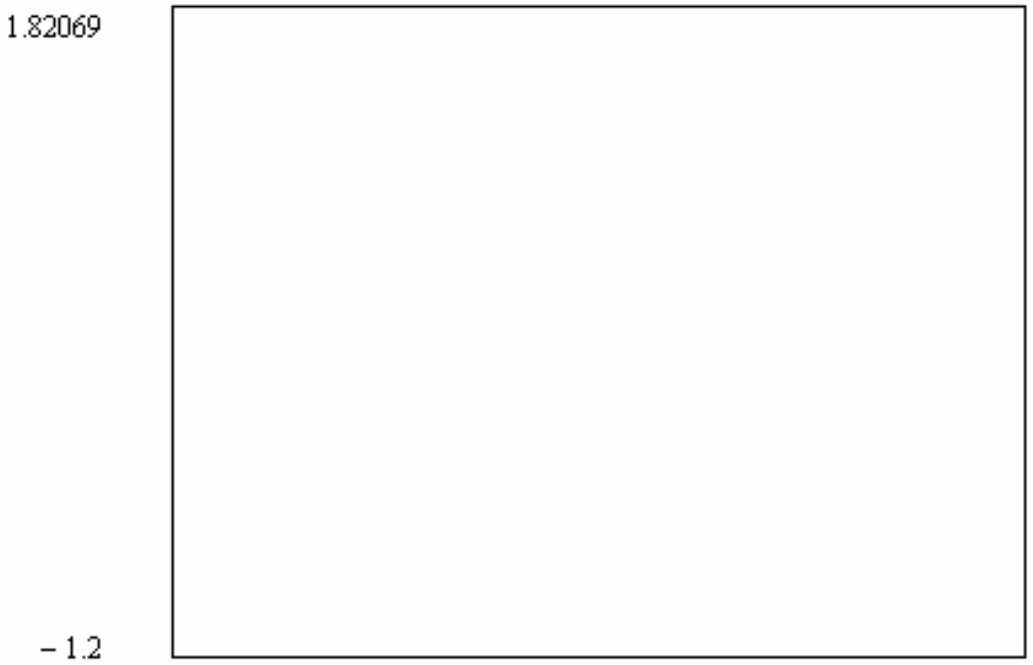
Linear convergence.

Faster than bisection method in most cases.

However rate of convergence depends on the function and it may be slower then bisection in some cases.

FALSE POSITION

r =	P_r =
0	8.2
1	9.9

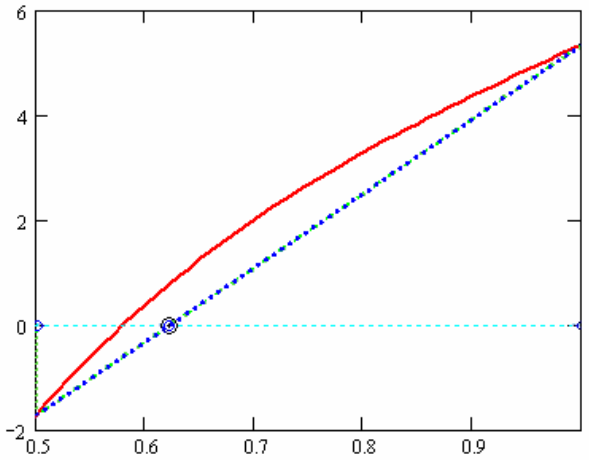


$f(x) := \sin(x) - \cos(k \cdot x)$

Ex: (pr.5.3) Determine the root of:

$-26+82.3x-88x^2+45.4x^3-9x^4+.65x^5=0$

$0.5<x<1$



$r =$

0
1
2

$P_r =$

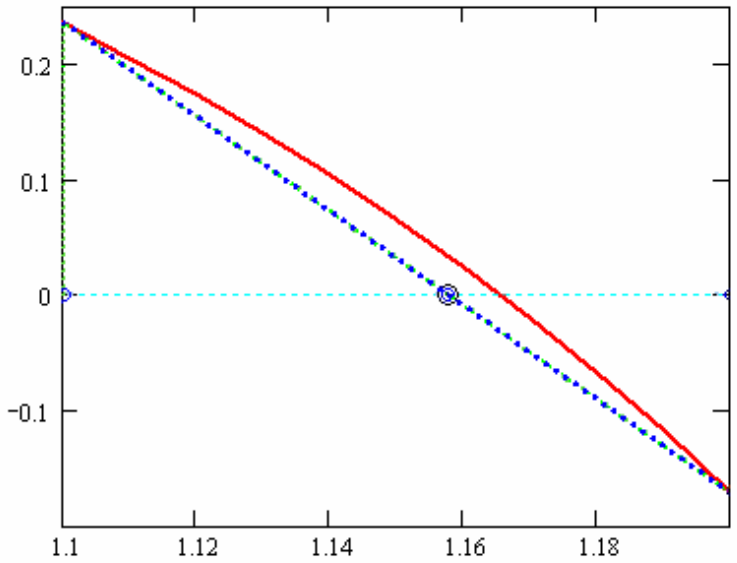
0.5
1
0.621490161397

	xl	xu	xr	f(xr)	ε_a
0	0.5	1	0.6214902	0.77450114	
1	0.5	0.62149016	0.5837269	0.084937795	6.46933565
2	0.5	0.58372691	0.5797807	0.008811573	0.68063975
3	0.5	0.57978069	0.5793734	0.000908652	0.07029948
4	0.5	0.57937339	0.5793314	9.36423E-05	0.007246
5	0.5	0.57933142	0.5793271	9.6498E-06	0.00074671
6	0.5	0.57932709	0.5793266	9.94402E-07	7.6948E-05
7	0.5	0.57932664	0.5793266	1.02472E-07	7.9294E-06
8	0.5	0.5793266	0.5793266	1.05596E-08	8.1711E-07
9	0.5	0.57932659	0.5793266	1.08816E-09	8.4203E-08
10	0.5	0.57932659	0.5793266	1.12137E-10	8.677E-09
11	0.5	0.57932659	0.5793266	1.15523E-11	8.9419E-10
20	0.5	0.57932659	0.5793266	3.20577E-15	1.9164E-14

Example F(x)=2x-tan(x)

F(1.1)= 0.23524
F(1.2)= -0.17215

Using False Position



r =

0
1
2

P_r =

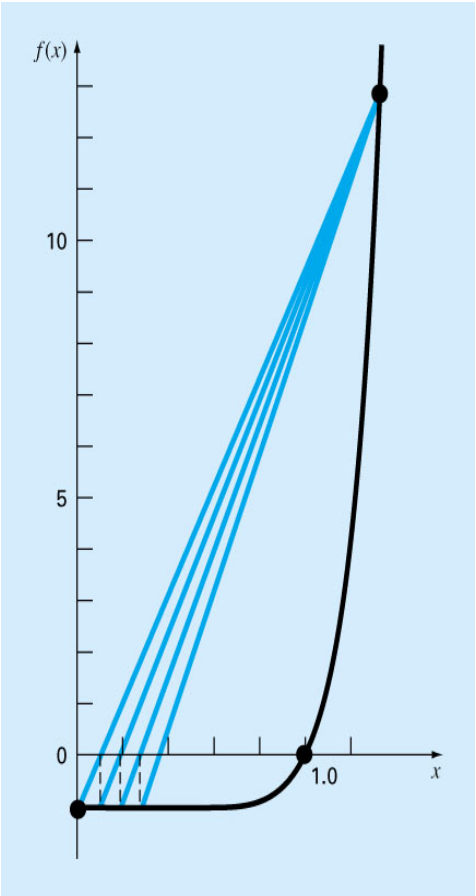
1.1
1.2
1.157743000116

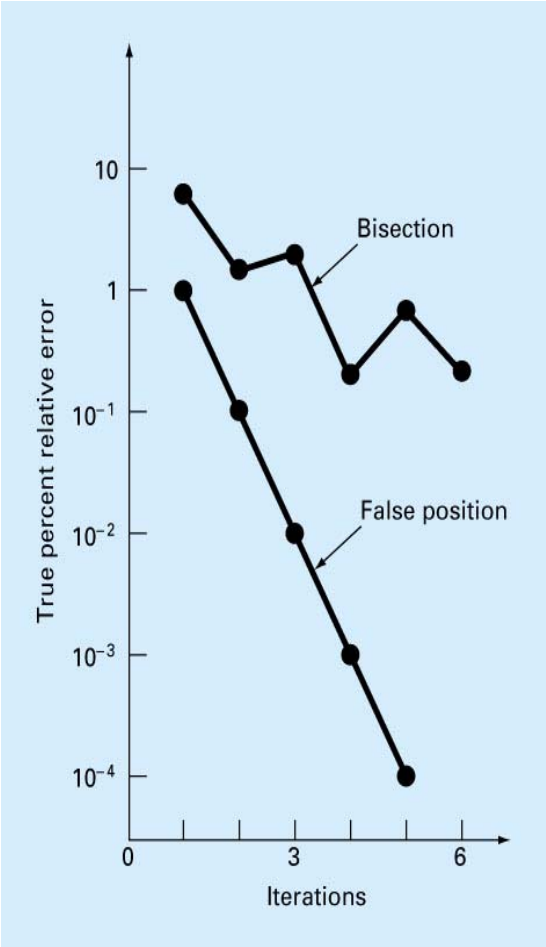
	x_l	x_u	x_r	$F(x_r)$	ε_a
0	1.1	1.2	1.1577430001	3.37675E-02	3.64994648
1	1.157743	1.2	1.1646724798	3.92882E-03	0.594972386
2	1.164672	1.2	1.1654607295	4.45283E-04	0.067634173
3	1.165461	1.2	1.1655498373	5.03161E-05	0.007645133
4	1.16555	1.2	1.1655599034	5.68371E-06	0.000863626
5	1.16556	1.2	1.1655610404	6.42006E-07	9.75518E-05
6	1.165561	1.2	1.1655611689	7.25179E-08	1.1019E-05
7	1.165561	1.2	1.1655611834	8.19126E-09	1.24465E-06
8	1.165561	1.2	1.1655611850	9.25243E-10	1.40589E-07
9	1.165561	1.2	1.1655611852	1.04511E-10	1.58803E-08
10	1.165561	1.2	1.1655611852	1.18057E-11	1.79375E-09

```
Function FalsePos(xl, xu, eps, imax, xr, iter, ea)
    xr = xu
    iter = 0#
    fl = Fun(xl)
    fu = Fun(xu)
    Do
        xrold = xr
        xr = xu - fu * (xl - xu) / (fl - fu)
        fr = Fun(xr)
        iter = iter + 1
        If xr <> 0 Then
            ea = Abs((xr - xrold) / xr) * 100
        End If
        Test = fl * fr
        If Test < 0 Then
            xu = xr: fu = fr
            Else
            xl = xr: fl = fr
        End If
        if ea>eps or iter>imax Exit
    End Do
FalsePos = xr
End FalsePos
```

Only one statement is different between the bisection and false position methods

```
Function ModFalsePos(xl, xu, eps, imax, xr, iter, ea)
    xr = xu
    iter = 0#
    fl = Fun(xl)
    fu = Fun(xu)
    Do
        xrold = xr
        xr = xu - fu * (xl - xu) / (fl - fu)
        fr = f(xr)
        iter = iter + 1
        If xr <> 0 Then
            ea = Abs((xr - xrold) / xr) * 100
        End If
        Test = fl * fr
        If Test < 0 Then
            xu = xr: fu = fr
            iu = 0: il = il + 1
            If il >= 2 Then fl = fl / 2
            Else If Test>0
            xl = xr: fl = fr
            il = 0:  iu = iu + 1
            If iu >= 2 Then fu = fu / 2
            Else
            ea = 0
        End If
        if ea>eps or iter>imax Exit
    End Do
ModFalsePos = xr
End ModFalsePos
```




10
1
10^{-1}
10^{-2}
10^{-3}
10^{-4}
True percent relative error
Bisection
False position
0
3
6
Iterations