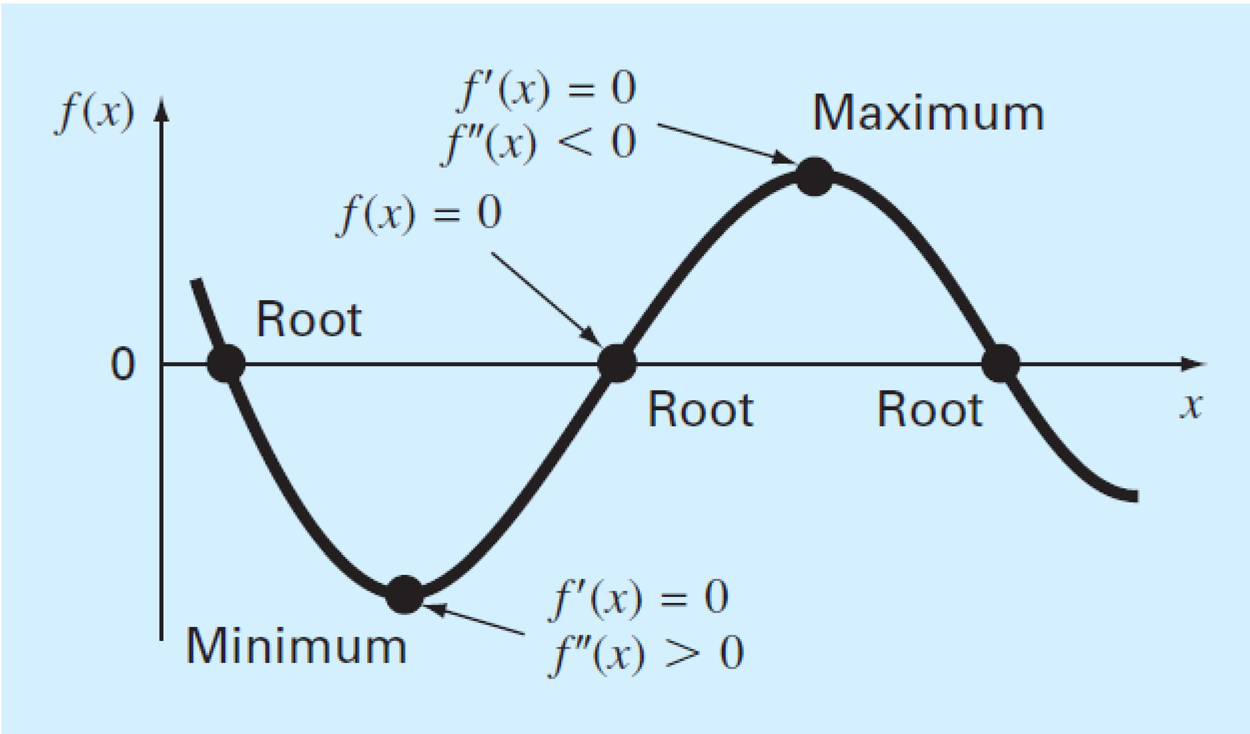
Roots by Bracket method

Hsiao-Lung Chan
Dept Electrical Engineering
Chang Gung University, Taiwan
chanhl@mail.cgu.edu.tw

Roots problem

- The solutions to *f(x)=0*
- Often occur when a design problem presents an implicit equation for a required parameter



Model for bungee jumper

- Explicit model function

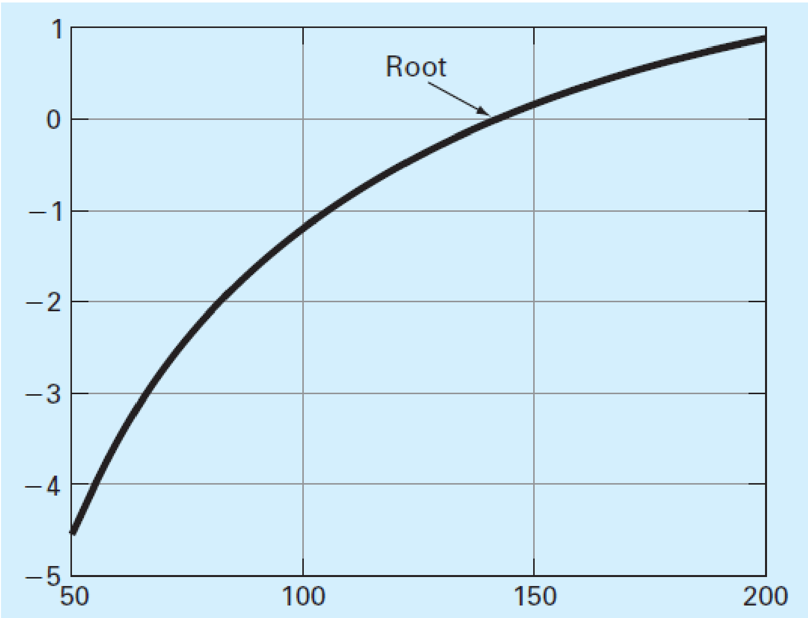
$$v(t) = \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}} t\right)$$

- Medical studies
  - A bungee jumper may sustain a vertebrae injury if the free-fall velocity exceeds 36 m/s after 4 s of free fall.
- Determine the mass (*m*) under the criterion given a drag coefficient (c_d)

Graphic approach

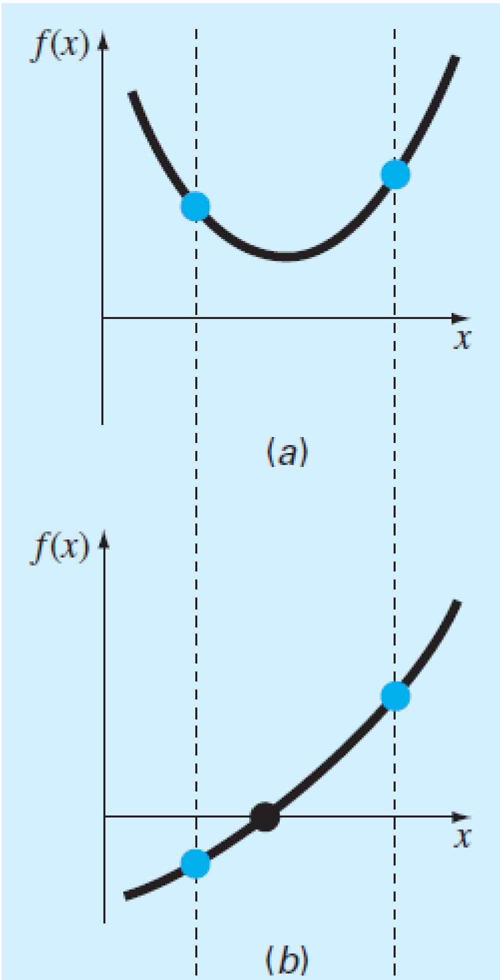
- Implicit equation

$$f(m) = \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}} t \right) - v(t)$$



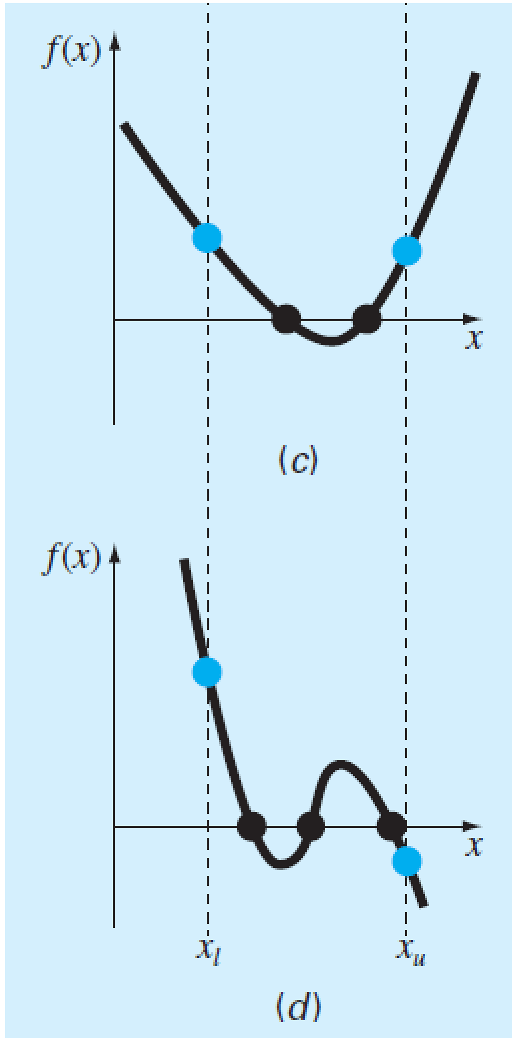

```
cd = 0.25;
g = 9.81;
v = 36;
t = 4;
mp = linspace(50,200);
fp = sqrt(g*mp/cd).* ...
   tanh(sqrt(g*cd./mp)*t) - v;
plot(mp,fp)
grid
```

General rule for number of roots in an interval



(a) Same sign, no roots
(b) Different sign, one root

(c) Same sign, two roots
(d) Different sign, three roots



Bracketing methods

- Based on two initial guesses that "bracket" the root
- Find **brackets** by incremental search
  - If f(x) is real and continuous on in the interval from x_l to x_u and $f(x_l)\ f(x_u) < 0$ (opposite signs)

    then there is at least one root between x_l and x_u

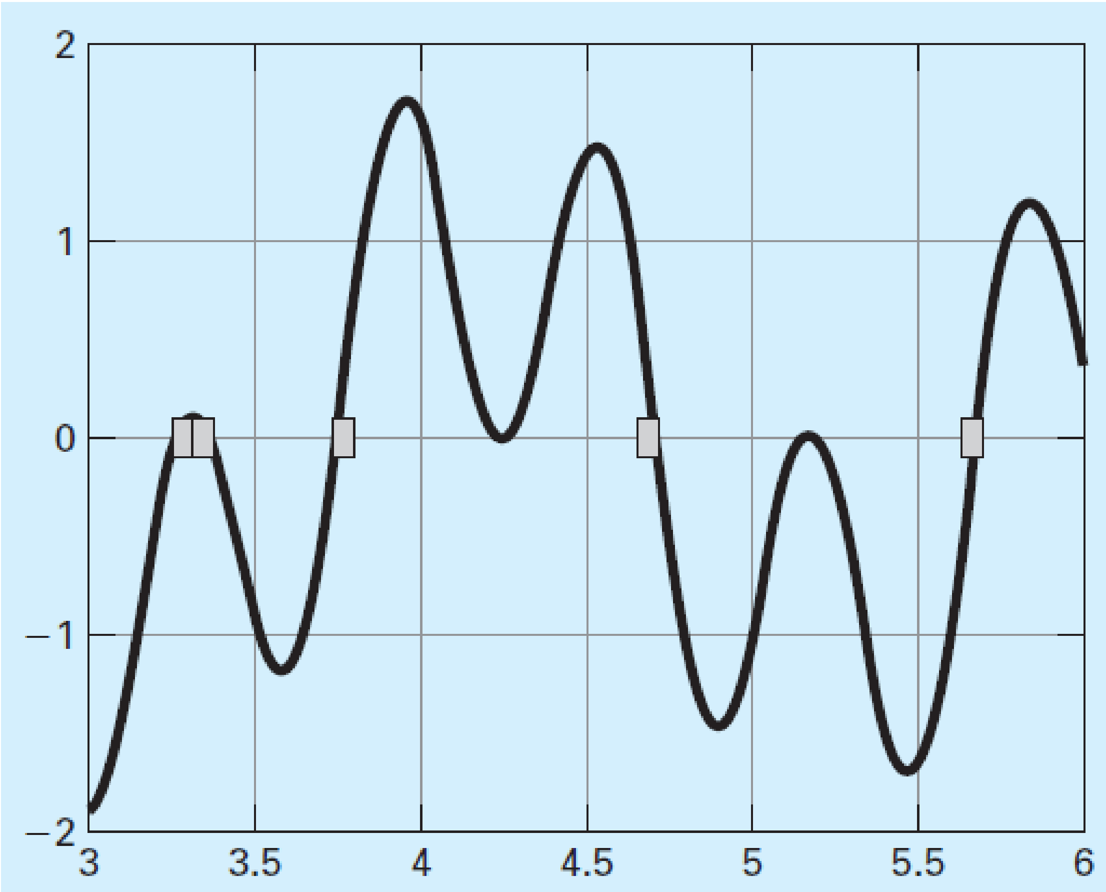
M-function for incremental research

```
function xb = incsearch(func,xmin,xmax,ns)
  if nargin < 4, ns = 50; end
  x = linspace(xmin,xmax,ns);
  f = func(x);
  nb = 0; xb = [];
  for k = 1:length(x)-1
  if sign(f(k)) ~= sign(f(k+1)) % check for sign change
    nb = nb + 1;
    xb(nb,1) = x(k);
    xb(nb,2) = x(k+1);
  end
end
```

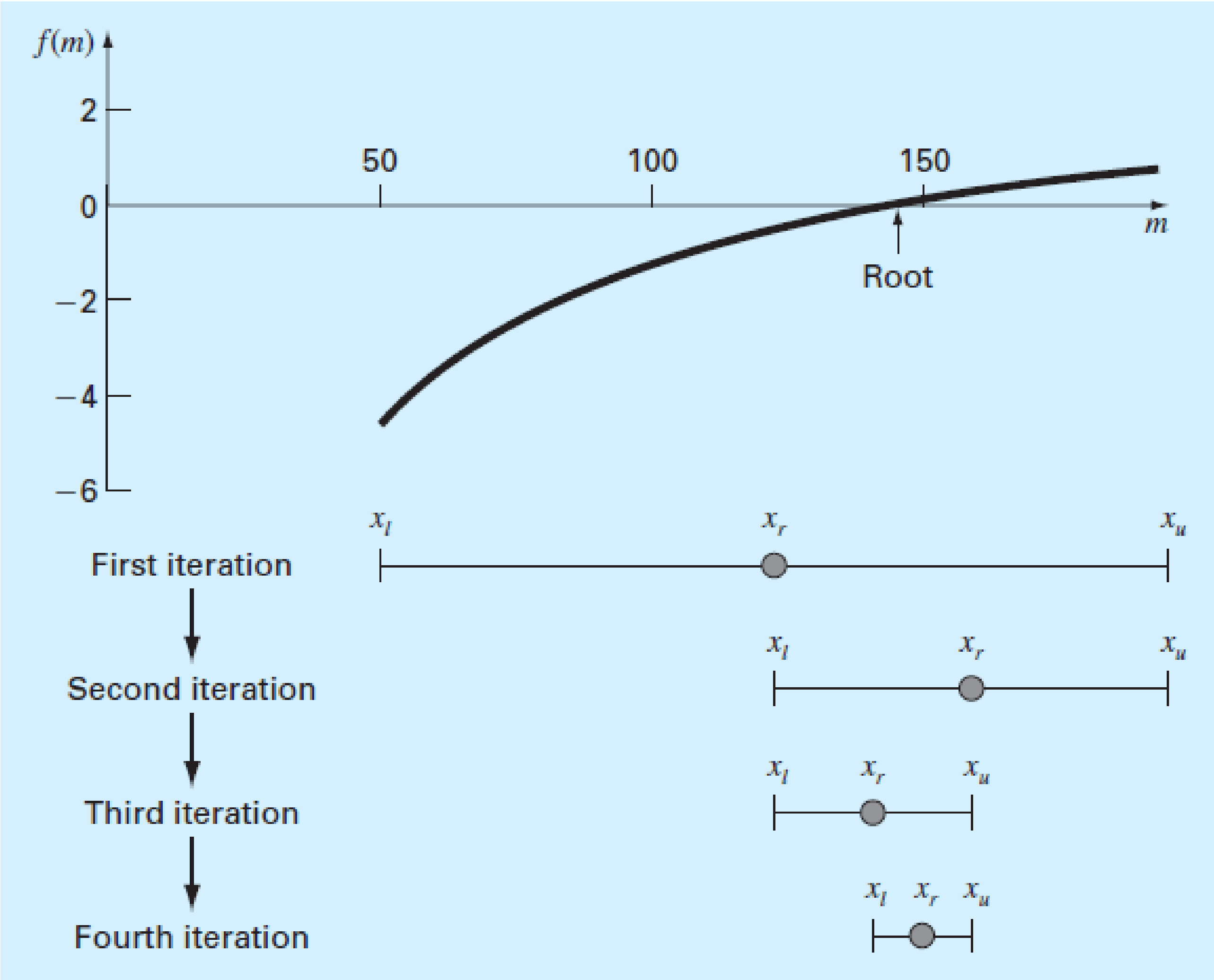
Calling code:

```
xb=incsearch(@(x) sin(10*x)+cos(3*x), 3,6);
```

$$f(x) = \sin(10x) + \cos(3x)$$



Bisection: find a root with a bracket



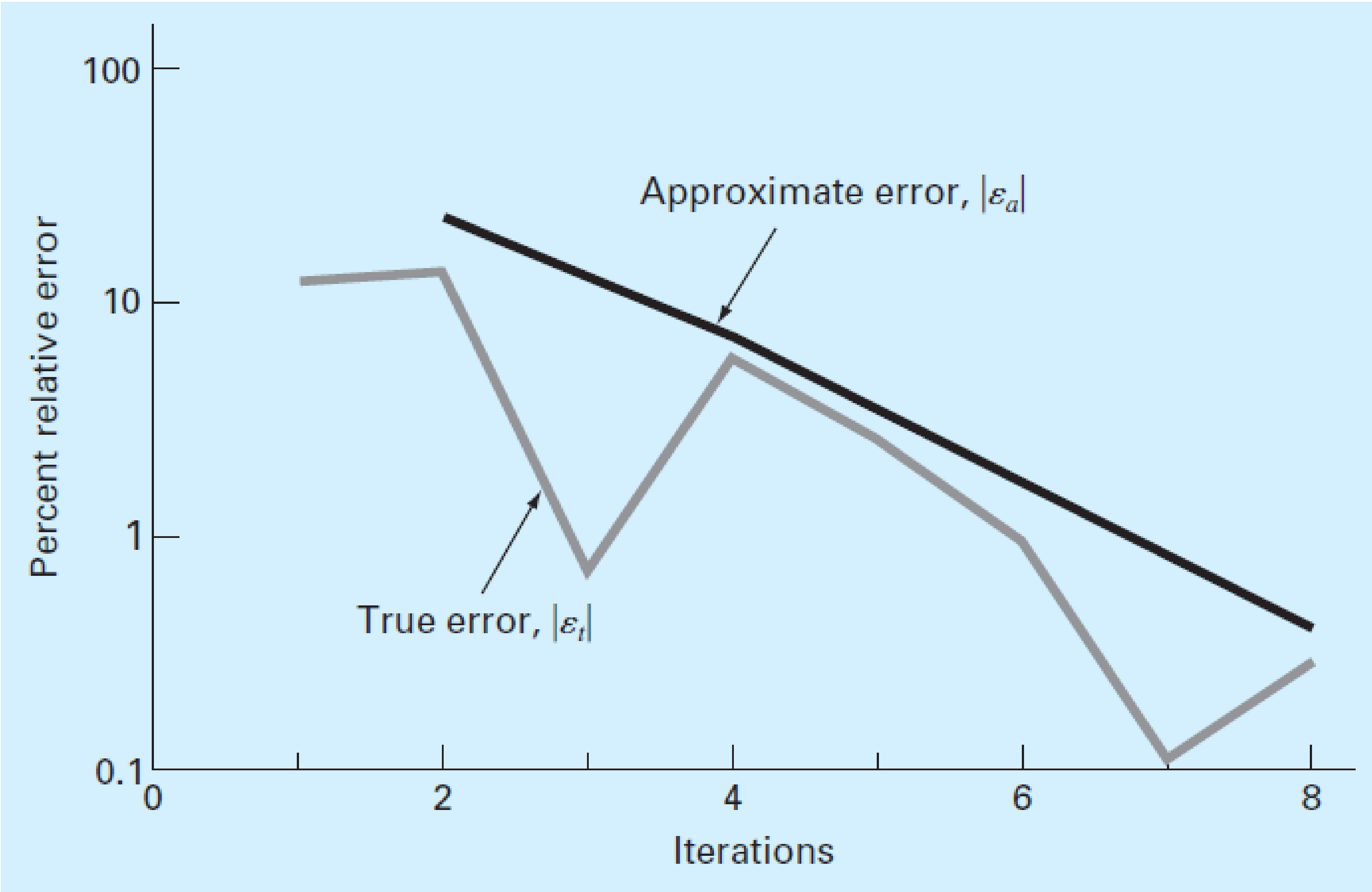
Bisection (cont.)

- Iteration search until the result is accurate enough (e.g. percent relative error < 0.5%)

$$|\varepsilon| = \left|\frac{x_{root} - x_r}{x_{root}}\right| \times 100\% \quad \xrightarrow{\text{Approximation}} \quad |\varepsilon_a| = \left|\frac{x_r^{new} - x_r^{old}}{x_r^{new}}\right| \times 100\%$$

Iteration	x_l	x_u	x_r	$\lvert\varepsilon_a\rvert$ (%)	$\lvert\varepsilon_t\rvert$ (%)
1	50	200	125		12.43
2	125	200	162.5	23.08	13.85
3	125	162.5	143.75	13.04	0.71
4	125	143.75	134.375	6.98	5.86
5	134.375	143.75	139.0625	3.37	2.58
6	139.0625	143.75	141.4063	1.66	0.93
7	141.4063	143.75	142.5781	0.82	0.11
8	142.5781	143.75	143.1641	0.41	0.30

Bisection (cont.)



M-function for bisection

```
function [root,fx,ea,iter]=bisect(func,xl,xu,es,maxit)
  if nargin<4 | isempty(es),  es=0.0001;  end
  if nargin<5|  isempty(maxit),   maxit=50;  end
  iter = 0; xr = xl; ea = 100;
  while (1)
     xrold = xr;
     xr = (xl + xu)/2;
     iter = iter + 1;
     if xr ~= 0
       ea = abs((xr - xrold)/xr) * 100;
     end
```

M-function for bisection (cont.)

```
    test = func(xl)*func(xr);
    if test < 0
      xu = xr;
    elseif test > 0
      xl = xr;
    else
     ea = 0;
    end
    if ea <= es | iter >= maxit,
      break;
    end
  end   % end of while loop
  root = xr;
  fx = func(xr);
```

Calling bisection function

- Bungee jumper problem

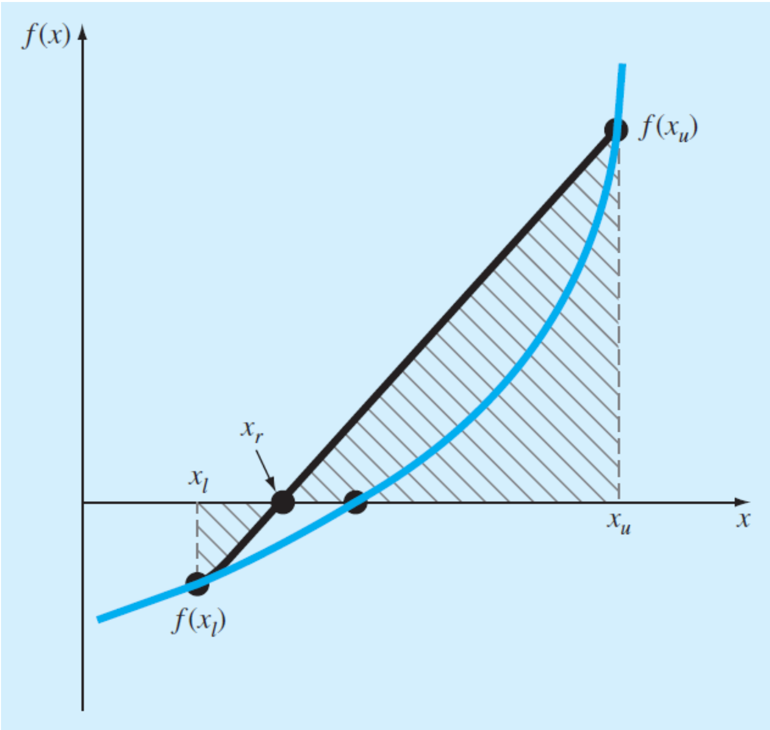
$$f(m) = \sqrt{\frac{9.81m}{0.25}} \tanh\left(\sqrt{\frac{9.81(0.25)}{m}} 4\right) - 36$$

```
fm=@(m) sqrt(9.81*m/0.25)*tanh(sqrt(9.81*0.25/m)*4)-36;
[mass fx ea iter]=bisect(fm,40,200);
```

False position (linear interpolation method)

- Similar bisection method but

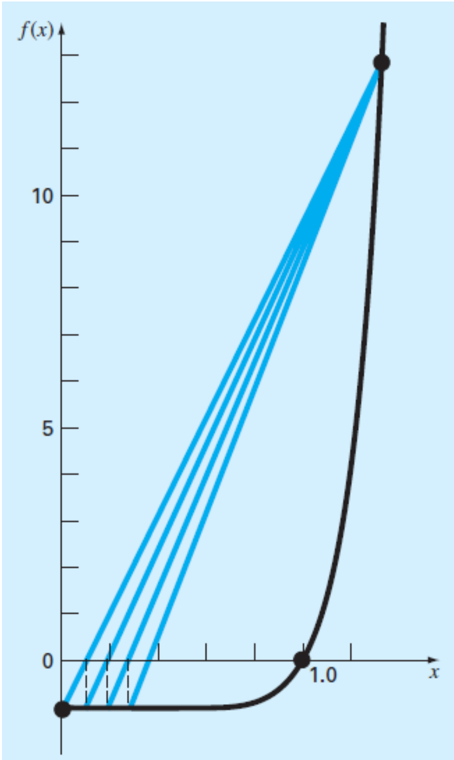
$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)}$$



Bisection vs. False position

- Drawback of bisection method
  - Does not take into account the shape of the function
- Drawback of false position method
  - Slow convergence in some functions

$$f(x) = x^{10} - 1$$



Reference

- Steven C. Chapra "Applied Numerical Methods with MATLAB", 3rd ed., McGraw Hill, 2012.