~ Roots of Equations ~

Bracketing Methods

Chapter 5

Credit: Prof. Lale Yurttas, Chemical Eng., Texas A&M University

Roots of Equations

- Easy

$$ax^2 + bx + c = 0 \quad \Rightarrow \quad x = \frac{-b \mp \sqrt{b^2 - 4ac}}{2a}$$

- But, not easy

$$ax^5 + bx^4 + cx^3 + dx^2 + ex + f = 0 \quad \Rightarrow \quad x = ?$$

- How about these?

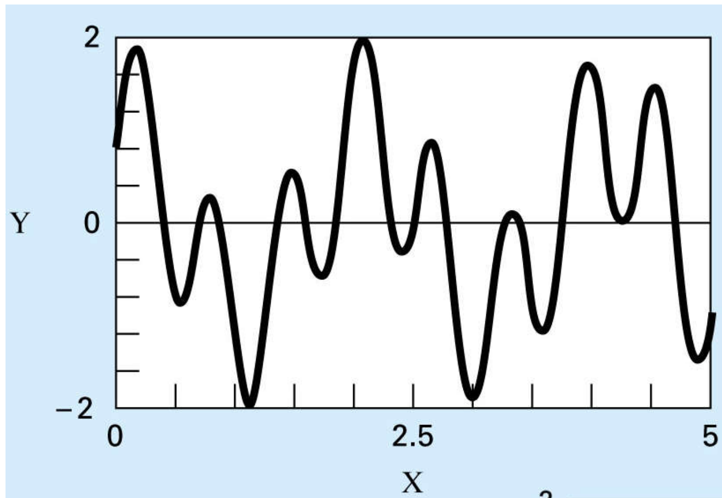
$$\sin x + x = 0 \quad \Rightarrow x = ?$$

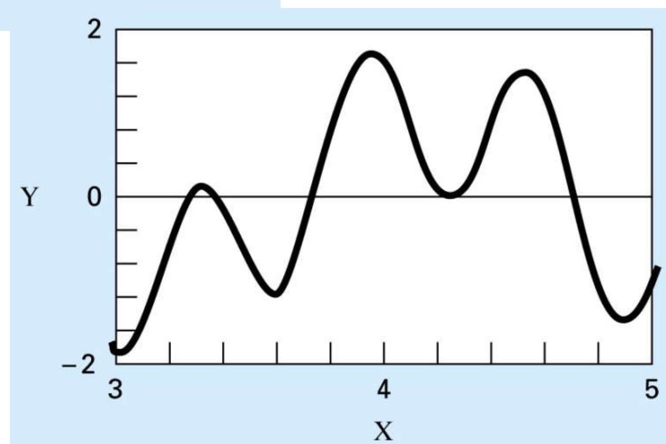
$$\cos(10x) + \sin(3x) = 0 \quad \Rightarrow x = ?$$

Graphical Approach

- Make a plot of the function f(x) and observe where it crosses the x-axis, i.e. f(x) = 0
- Not very practical but can be used to obtain rough estimates for roots
- These estimates can be used as initial guesses for numerical methods that we'll study here.

Using MATLAB, plot *$f(x)=\sin(10x)+\cos(3x)$*





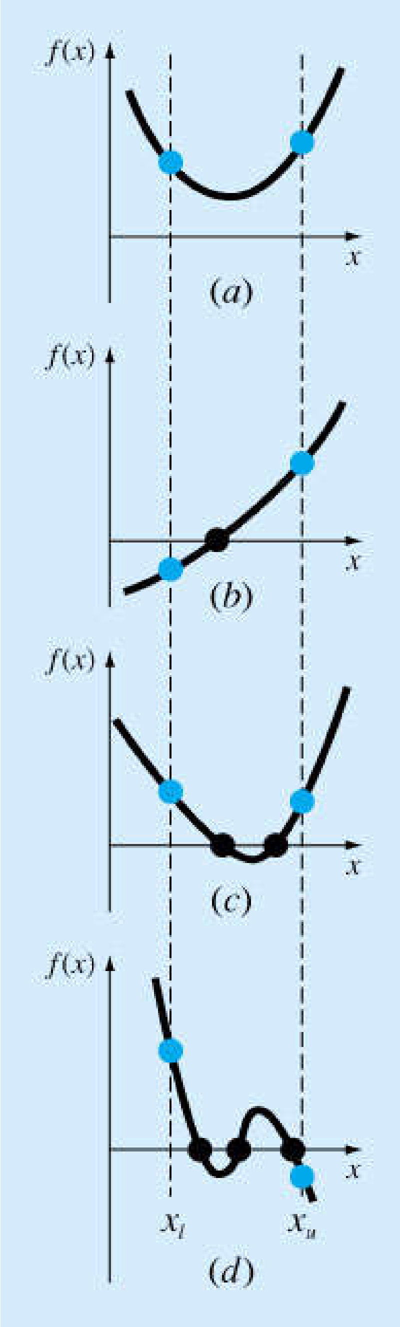
Two distinct roots between

x= 4.2 and 4.3

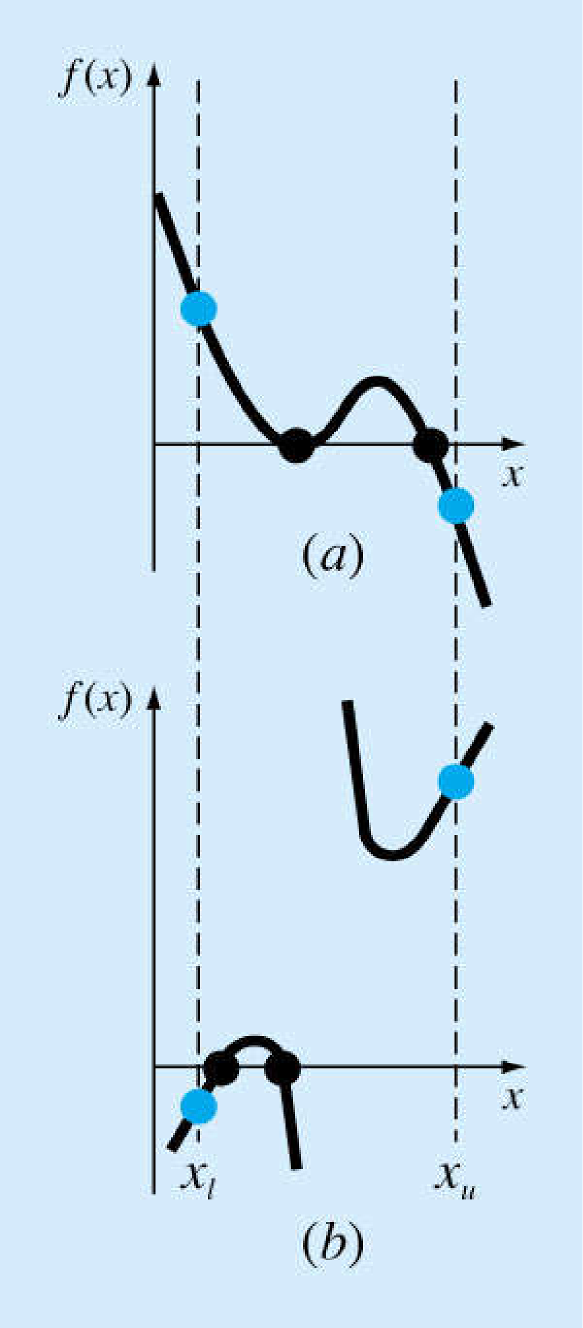
need to be careful

Bracketing:

Odd and even number of roots



exceptions



Bisection Method

Step 1: Choose 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 that $f(x_l)f(x_u) < 0$.

Step 2: An estimate of the root x_r is determined by

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

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 $x_u = x_r$ and return to step 2.

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

(c) If $f(x_l)f(x_r) = 0$, the root equals x_r; terminate the computation.

$$\text{Relative error estimate:} \quad \varepsilon = \frac{\left|x_r^{new} - x_r^{old}\right|}{\left|x_r^{new}\right|} 100\%$$

Termination criteria: $\varepsilon < \varepsilon_{tol}$ OR *Max.Iteration* is reached

MATLAB code

Bisection Method

- **Minimize function evaluations in the code.**

  Why?

- **Because they are costly (takes more time)**

```
% Bisection Method
% function is available in another file e.g. func1.m
% A sample call:    bisection2(@func1, -2, 4, 0.001, 500)

function root = bisection(fx, xl, xu, es, imax);

if   fx(xl)*fx(xu) > 0              % if guesses do not bracket
    disp('no bracket')
    return
end

for i=1:1:imax
   xr=(xu+xl)/2
   ea = abs((xu-xl)/xl);

   test= fx(xl)*fx(xr);
   if test < 0
       xu=xr;
   end
   if test > 0
       xl=xr;
   end
   if test == 0
       ea=0;
   end
   if ea < es
       break;
   end

end
```

How Many Iterations will It Take?

- Length of the first Interval $L_o = x_u - x_l$
- After 1 iteration $L_1 = L_o/2$
- After 2 iterations $L_2 = L_o/4$

  ….. …..
- After k iterations $L_k = L_o/2^k$

- Then we can write:

$$\left|\frac{L_k}{x_r}\right| \le error\_tolerance \qquad \text{where } x_r = Min\{|x_l|, |x_u|\}$$

$$\left|\frac{L_0}{2^k}\right| \le |x_r| * \varepsilon_{tol}$$

$$2^k \ge \left|\frac{L_0}{|x_r| * \varepsilon_{tol}}\right| \qquad \Rightarrow \qquad \boxed{k \ge \left\lceil \log_2 \left|\frac{L_0}{x_r * \varepsilon_{tol}}\right| \right\rceil}$$

*here Bisection Method

Pros

- Easy
- Always finds a root
- Number of iterations required to attain an absolute error can be computed a priori.

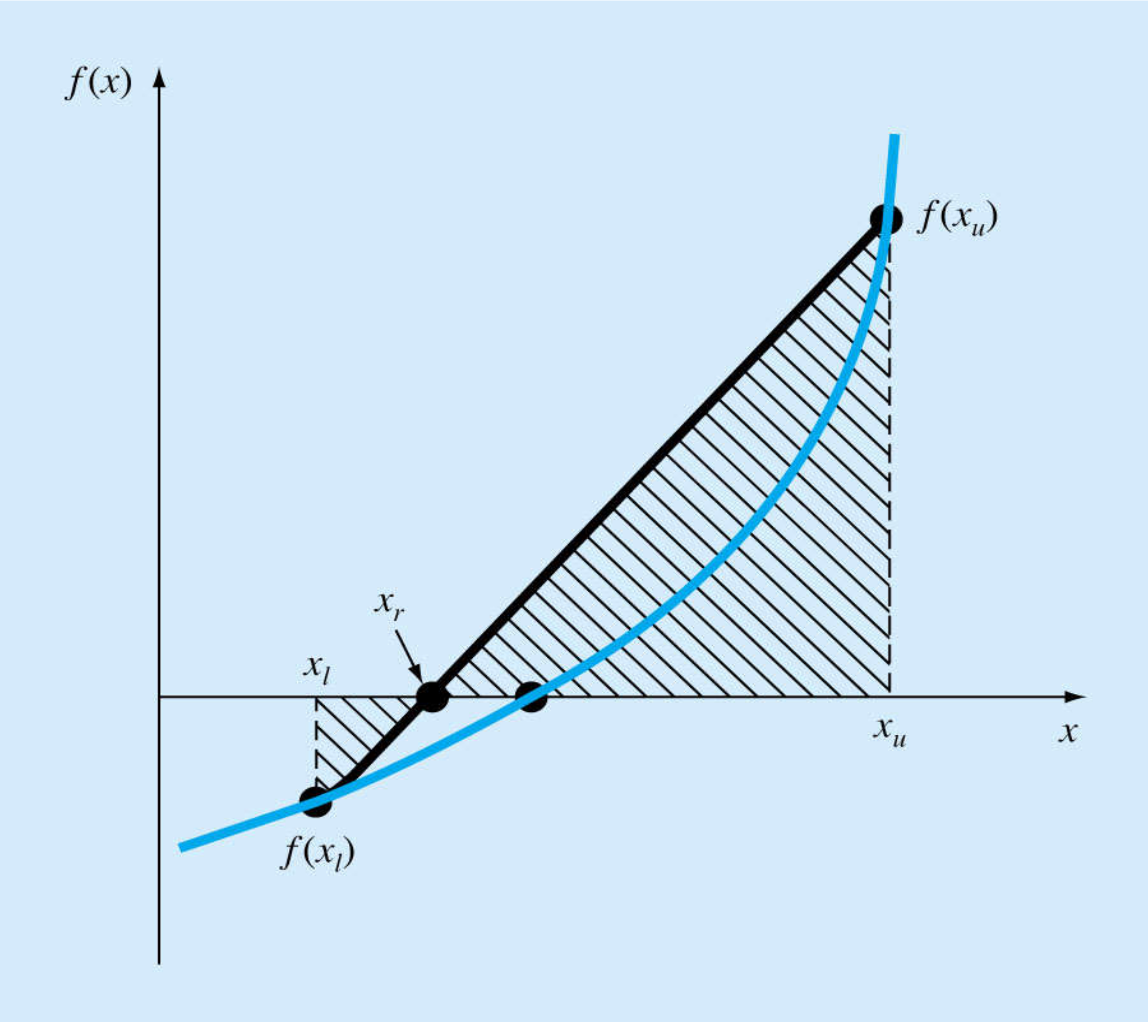
Cons

- Slow
- Need to find initial guesses for x_l and x_u
- No account is taken of the fact that if $f(x_l)$ is closer to zero, it is likely that root is closer to x_l .

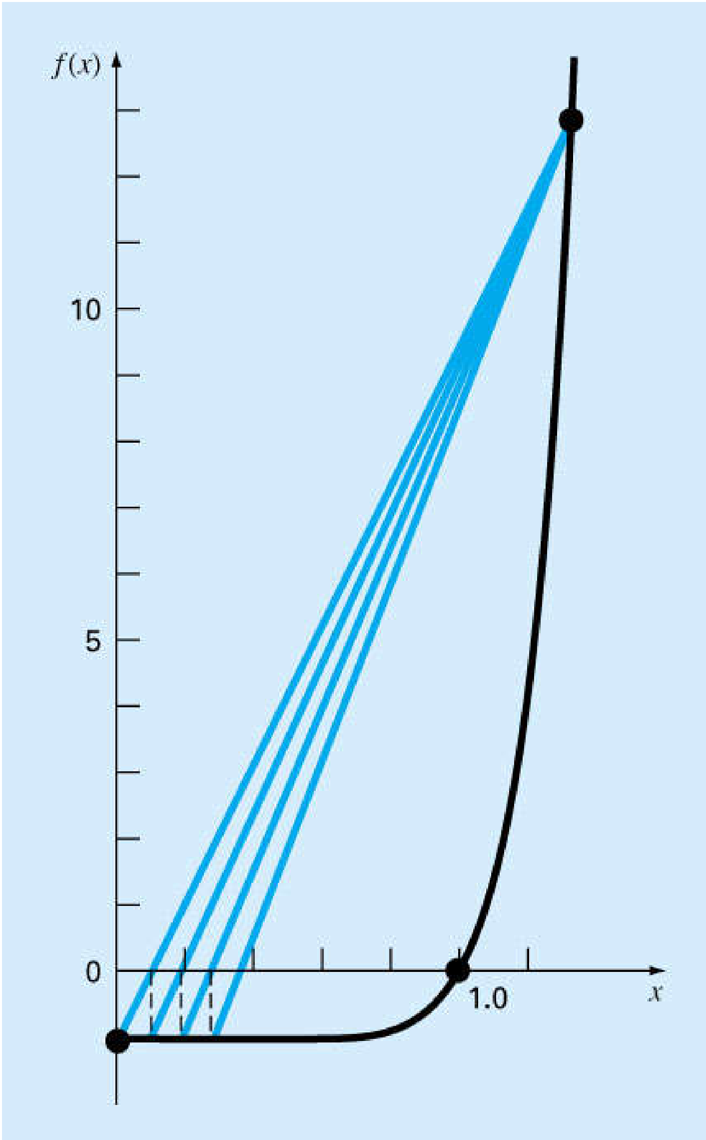
The False-Position Method (Regula-Falsi)

- We can approximate the solution by doing a *linear interpolation* between $f(x_u)$ and $f(x_l)$
- Find x_r such that $l(x_r)=0$, where $l(x)$ is the linear approximation of $f(x)$ between x_l and x_u
- Derive x_r using similar triangles

$$x_r = \frac{x_l f_u - x_u f_l}{f_u - f_l}$$



The False-Position Method



Works well, but not always!

←← Here is a pitfall ☹

Modified False-Position

One way to mitigate the "one-sided" nature of the **false position** (i.e. the pitfall case) is to have the algorithm pick the smallest bracket (between the Bisection Method & this one)

How to find good initial guesses?

- Start at one end of the region of interest (x_a) and evaluate $f(x_a)$, $f(x_a+\Delta x)$, $f(x_a+2\Delta x)$, $f(x_a+3\Delta x)$, ...

- Continue until the *sign* of the result changes.
  If that happens between $f(x_a+k*\Delta x)$ and $f(x_a+(k+1)*\Delta x)$

  then pick $x_l = x_a+k*\Delta x$ and $x_u = x_a+(k+1)*\Delta x$

Problem:

if Δx is too small ➔ search is very time consuming
if Δx is too large ➔ could jump over two closely spaced roots

Suggestions:

- Generate random x values and evaluate f(x) each time until you find two values that satisfy f(x1)*f(x2) < 0
- Know the application and plot the function to see the location of the roots, and pick x_l and x_u accordingly to start the iterations.