Root finding $F(x) = 0$

One of the most important computational problems !
(and mathematical)

Root of $F(x)$ or zero of $F(x)$ or solution of $F(x) = 0$

is a value x such that $F(x) = 0$

Basic methods: (for simplest 1-dim $F: \mathbb{R} \to \mathbb{R}$, single equ. in one variable)

1. plotting ! (for 1-D problems)

2. Bisection method (only for 1-D) $F(x) = 0$

3. Fixed Point Iteration for $x = g(x)$

4. Newton-Raphson method for $F(x) = 0$

⋮ ⋮

- All root finding methods are iterative: start with an initial guess x_0,
  generate $x_1, x_2, \ldots, x_n, \ldots$
  hopefully $\{x_n\} \longrightarrow$ root
- Finding (approximating) one root at a time.

Matlab root finders
roots of polynomials
fzero of any $F(x)$

Bisection Method $F(x)=0$, $F:\mathbb{R}\to\mathbb{R}$ only

1. Applies to $F\in C[a,b]$ that changes sign: $F(a)\cdot F(b)<0$

2. based on Intermediate Value Thm:

   F cont's on $[a,b]$, $F(a)\cdot F(b)<0 \Rightarrow F(r)=0$ for <u>some</u> $r\in(a,b)$.

3. Idea: ~~x~~ $\approx \frac{a+b}{2}$ i.e. bisect $[a,b]$

   - sign will change on one of subintervals
   - repeat on that subinterval
   - $|\text{error}| \le \frac{b-a}{2} \to 0$ so it <u>will</u> converge to root

4. Math algorithm: Given $F(x)$ cont's on an interval I containing a root r:

   1. Find $a, b\in I$: $F(a)F(b)<0$ by any means (plotting, guessing, praying, trial...)
   2. Set $a_0=a$, $b_0=b$, $x_0=\frac{a_0+b_0}{2}\equiv a_0+\frac{b_0-a_0}{2}$. Residual $F(x_0)\overset{?}{\approx}0$

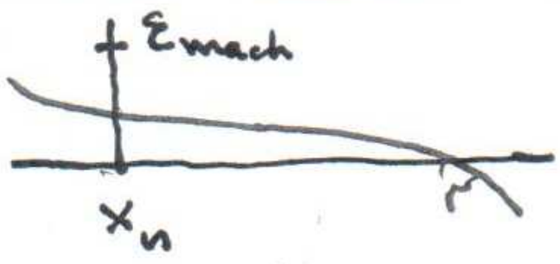
      If $|F(x_0)|<\text{TOL}$ DONE? perhaps... $|\text{error}|\le\frac{b-a}{2}$
   3. if $F(a_0)\cdot F(x_0)<0$ then $\exists$ root in $[a_0,x_0]$: set $a_1=a_0$, $b_1=x_0$

      else if $F(x_0)\cdot F(b_0)<0$ then root in $[x_0,b_0]$: set $a_1=x_0$, $b_1=b_0$

      $|\text{error}| = |x_0-r| \le \frac{b_0-a_0}{2}$
   4. Repeat: $x_n=\frac{a_n+b_n}{2}$, $n=1,2,\ldots$ till $\boxed{|F(x_n)|<\text{TOL} \text{ and } |\text{error}|<\text{TOL}}$

      $|\text{error}| = |x_n-r| \le \frac{|b_n-a_n|}{2} \le \ldots \le \frac{|b_0-a_0|}{2^n} \to 0$ as $n\to\infty$

<u>Caution</u>: **ill-conditioned problems:** small residual may $\not\Rightarrow$ small error!



$F(x_n)\approx 0$ even though x_n far from root r!

Must check <u>both</u> residual <u>and</u> error:

$|F(x_n)|<\text{TOL}$ <u>and</u> $|\text{error}|<\text{TOL}$

This applies to <u>all</u> root finders.

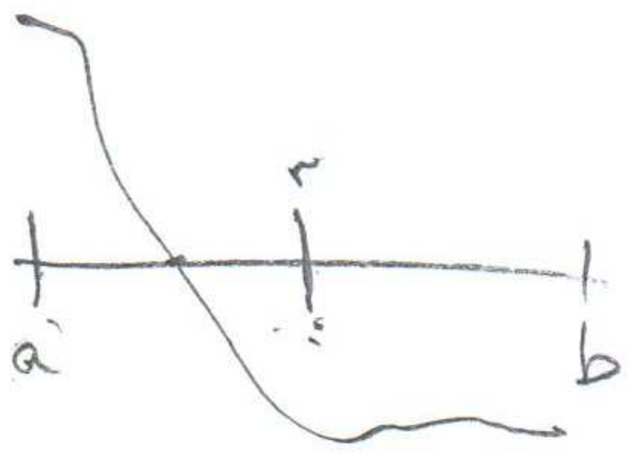
Moreover, any iterative process should be terminated by convergence criteria <u>and</u> independently (by maxITerations).

Bisection Method for $F(x)=0$

Computational Algorithm

0. define $F(x)$ in function $y = FCN(x)$

1. Inputs: a, b, TOL, maxIT
   and print them out



2. Evaluate F at a, b:
   $Fa = F(a)$, $Fb = F(b)$

3. Test if Bisection is applicable:
   if $Fa \cdot Fb > 0$ STOP (after printing diagnostic msg)
   error $= |b-a|$

4. (Else) do Bisection: for n=1:maxIT
   $r = \frac{a+b}{2}$, $Fr = F(r)$

   if $Fr \cdot Fa < 0$
      $b = r$, $Fb = Fr$
   else ($Fr \cdot Fb < 0$)
      $a = r$, $Fa = Fr$
   end
   error $= |b-a|$ or error $= \frac{\text{error}}{2}$
   print: n, r, Fr, error

5. Test convergence:

   if error < TOL and $|Fr| <$ TOL
      print: DONE! root = r, F = Fr, error, after n iterations
      break
   end
   Go to 4 (continue iterations)

Bisection Method for F(x)=0

Coding: FCN.m , Bisect.m

```
function y = FCN(x)
    y = .... (formula for F(x)
end
```

```
% Bisection code
%1. -- set/get inputs: ---
   a=-5 ; b=10; TOL=1.e-14 ; maxIT=50;
   fprintf('Inputs: a=%f, b=%f, TOL=%  , maxIT=% \n', a, b, TOL, maxIT)
%2. --- evaluate F: ---
   Fa=      ; Fb=     ;
%3. --- Is bisection applicable? ---
   if Fa*Fb >0 ; fprintf('no sign change, try different a,b. EXITING')
      exit
   end
%4. --- Bisection ---
   error =
   for n=1:maxIT
      r=(a+b)/2;  Fr=FCN(r);
      if( Fa*Fr < 0) %...root in [a,r]:
         b=r; Fb=Fr;
      else                %...root in [r,b]:
         a=r; Fa=Fr;
      end %if
      error = error/2;
      fprintf('%d %f %f %e \n', n, r, Fr, error)
%5. --- Convergence test:
      if( error < TOL) && ( abs(Fr)<TOL)
         fprintf('DONE! root=%f, F=%f, error=%e, after %d iters \n',
                                          r, Fr, error, n)
         break
      end %if
   end %for
if( n > maxIT )
      fprintf('>>> out of iters, NOT converged, try bigger maxIT \n')
end %if
```

%--- using while-loop:

while(

Bisection advantages / disadvantages

Advantages:

1. Robust method: if it applies it will work!

2. error is known explicitly (rare case):

$$|x_n - r| \leq \frac{b_n - a_n}{2} \leq \frac{b_0 - a_0}{2^n}, \quad n = 0, 1, 2, \ldots$$

3. Uses only sign of $F(x)$, not the value!

Disadvantages

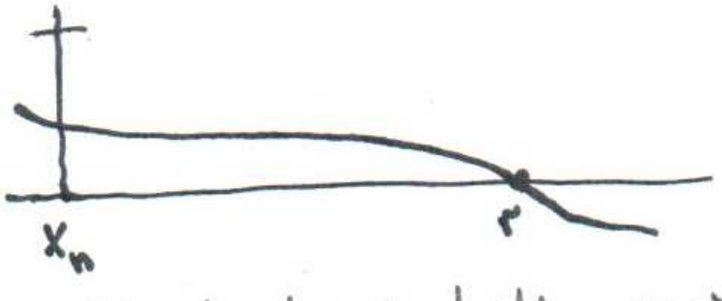
1. Slow: gains one binary digit per iteration
   (so need ~4 iterations for a decimal digit)

2. Only for simple roots, cannot be applied to 

   ($F(r) = 0$, $F'(r) \neq 0$)

3. Only for 1-D, only for real-valued F, not for complex

4. Like <u>all</u> root finders, trouble on ill-conditioned problems
   (small residual $\not\Rightarrow$ small error)



$F(x_n) \approx 0 \not\Rightarrow$ small error
(x_n near r)

Must check <u>both</u> residual <u>and</u> error

$|F(x_n)| < \text{TOL}$ $\quad$ $|\text{error}| < \text{TOL}$

Root finding: Fixed Point Iteration for $x = g(x)$

General method for any problem in "fixed point form": $x = g(x)$

Looking for x that remains unchanged by g (a fixed pt of $g(x)$)
so for intersection of $y = g(x)$ with $y = x$.

$F(x) = 0$ can be written in fixed pt form in many different ways:

e.g. $F(x) = x^2 - x - 1 = 0$

(a) $x = x^2 - 1$

(b) $x = \pm\sqrt{x+1}$

(c) $x = \frac{x+1}{x} = 1 + \frac{1}{x}$

....

(roots $\frac{1 \pm \sqrt{5}}{2}$ golden ratio!)

e.g. $F(x) = x + \ln x = 0$

(a) $x = -\ln x$ (d) $x = 2x + \ln x$

(b) $x = e^{-x}$

(c) $x = \frac{x + e^{-x}}{2}$

....

Fixed point Iteration: 1. Let x_0 be an initial guess

1. Generate successive approximation by repeated substitution:

$$x_{n+1} = g(x_n), \quad n = 0, 1, \cdots$$

till convergence (if lucky!): $|\Delta x| < TOL$ & $|x_n - g(x_n$

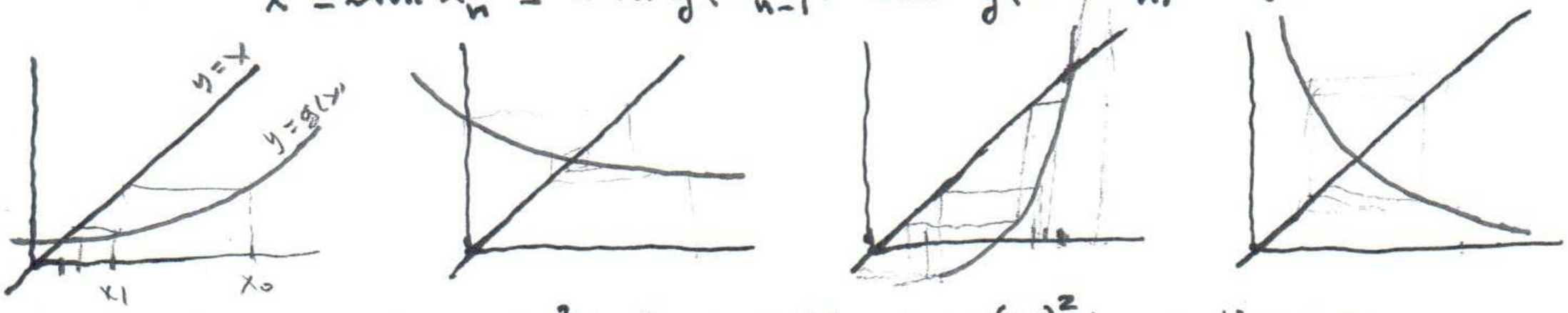
To be succesful we must have:

(a) convergence i.e. $\lim_{n\to\infty} x_n = \bar{x}$ exists.

(b) the limit $\bar{x}$ is a fixed pt: $\bar{x} = g(\bar{x})$, solution.

If $g(x)$ is continuous then (a) $\Rightarrow$ (b):

$$\bar{x} = \lim x_n = \lim g(x_{n-1}) \overset{\text{cont's}}{=\!=} g(\lim x_n) = g(\bar{x}).$$



e.g. $x = x^2 - 1$: $x_0 = 3$, $x_1 = 3^2 - 1 = 8$, $x_2 = 63$, $x_3 = (63)^2 - 1$, ... diverges.

$x = 1 + \frac{1}{x}$: $x_0 = 3$, $x_1 = 1 + \frac{1}{3} = \frac{4}{3}$, $x_2 = 1 + \frac{3}{4} = \frac{7}{4}$, $x_3 = 1 + \frac{4}{7} = \frac{11}{7}$, ... converges.

Thm: Let $\bar{x}$ be a fixed pt of $g(x)$.
If $g(x)$, $g'(x)$ cont's near $\bar{x}$ and $|g'(\bar{x})| < 1$ then $\{x_n\} \to \bar{x}$ if x_0 close to $\bar{x}$.
$|\quad| > 1$ then $\{x_n\}$ diverges away from $\bar{x}$.

Fixed Point Iteration root finder

Advantages: 1. very simple to implement, and very general

2. applies in any dimention, to systems,
to operator equations, to anything of the form

$$x = g(x)$$

Disadvantages: 1. need a good $g(x)$, with $|g'(\bar{x})| < 1$

2. need x_0 close to $\bar{x}$

3. linearly convergent (if it converges...)
gains ~ one binary digit per iteration