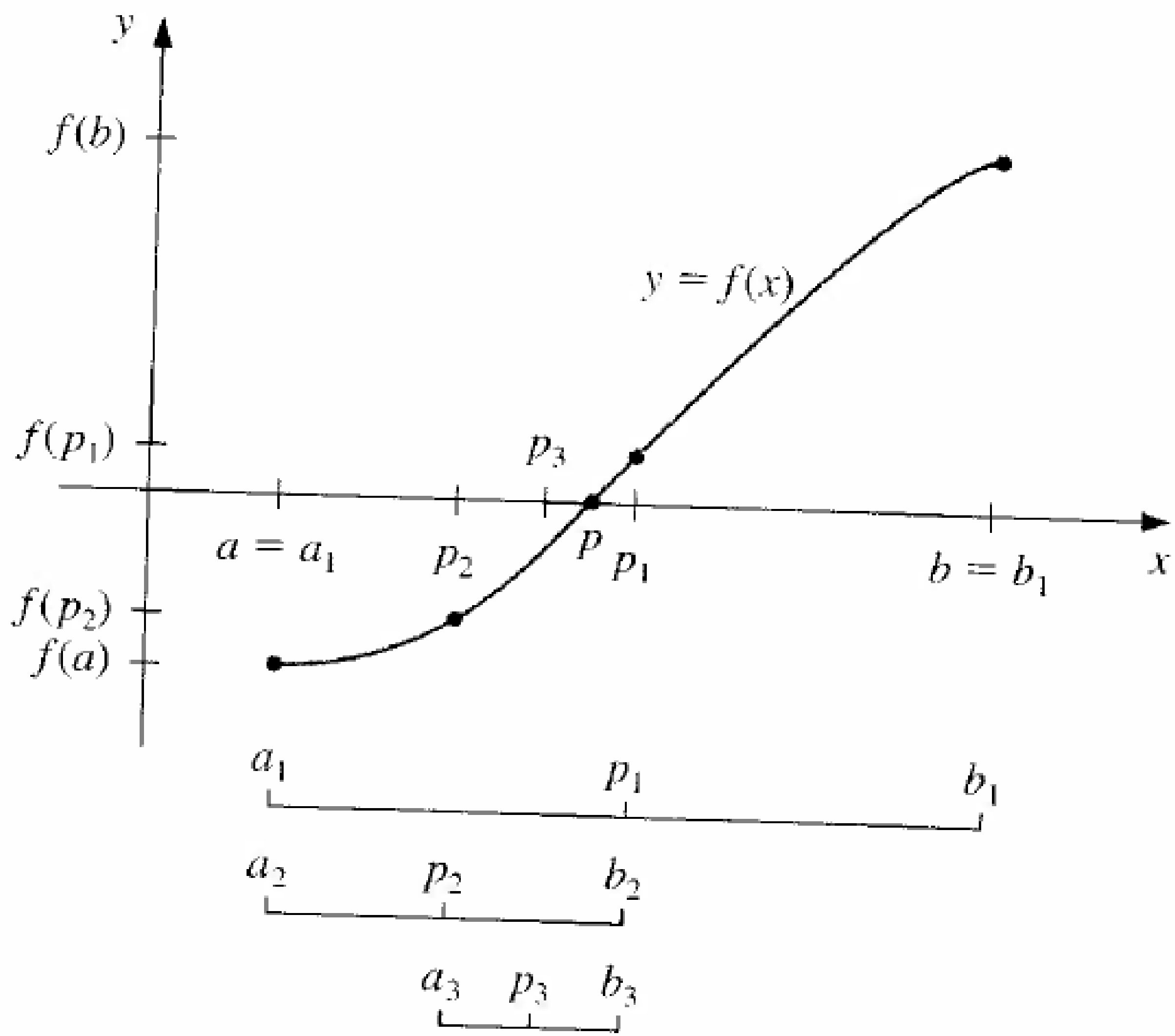
2.1 The Bisection Method

- Bisection method is a method for finding a root of equation of the form $f(x) = 0$.
- Suppose f is continuous on the interval $[a, b]$, with $f(a)$ and $f(b)$ of opposite sign, then Bisection method performs as follows:

Set $a_1 = a, b_1 = b, p_1 = \frac{a_1+b_1}{2}$

- If $f(p_1) = 0$, then $p = p_1$. DONE
- If $f(p_1) \neq 0$:
  - If $f(p_1)$ and $f(a_1)$ have the same sign, then set $a_2 = p_1, b_2 = b_1$.
  - If $f(p_1)$ and $f(b_1)$ have the same sign, then set $a_2 = a_1, b_2 = p_1$.

Repeat the process to the interval $[a_2, b_2]$.


y
f(b)
y = f(x)
f(p1)
p3
a = a1
p2
p
p1
b = b1
x
f(p2)
f(a)
a1
p1
b1
a2
p2
b2
a3
p3
b3

Facts to remember:

1. The sequence of intervals $\{(a_i, b_i)\}_{i=1}^{\infty}$ contains the desired root.

2. Intervals containing the root: $(a_1, b_1) \supset (a_2, b_2) \supset (a_3, b_3) \supset (a_4, b_4) \ldots$

3. After n steps, the interval (a_n, b_n) has the length: $b_n - a_n = (1/2)^{n-1}(b - a)$

4. Let $p_n = \frac{b_n + a_n}{2}$ be the mid-point of (a_n, b_n). The limit of sequence $\{p_n\}_{n=1}^{\infty}$ is the root.

Convergence

- **Theorem 2.1**

Suppose function $f(x)$ is continuous on $[a, b]$, and $f(a) \cdot f(b) < 0$. The Bisection method generates a sequence $\{p_n\}_{n=1}^{\infty}$ approximating a zero p of $f(x)$ with

$$|p_n - p| \leq \left({}^{1}\!/_{2}\right)^n (b - a), \qquad \text{when } n \geq 1$$

- **Convergence rate**

The sequence $\{p_n\}_{n=1}^{\infty}$ converges to p with the rate of convergence $O\left(\left({}^{1}\!/_{2}\right)^n\right)$:

$$p_n = p + O\left(\left({}^{1}\!/_{2}\right)^n\right)$$

Example 1. Consider equation

$$3x^2 - 2 = x + 1$$

on the interval $[a, b] = [0,2]$. If p_1 is the midpoint between a and b, use the Bisection Method to find the third approximation p_3.

Example 2. Determine the number of iteration needed such that the absolute error in *Example 1* is less than 10^{-3}. (**hint: apply Theorem 2.1**)

Solution: Since $|p_n - p| \leq (1/2)^n (b_1 - a_1) \leq 10^{-3}, \rightarrow\ 2^{-n}(2-0) \leq 10^{-3}$.

Solve for $n \rightarrow n \approx 10.96$.

So $n = 11$ is needed.

Exercise 3. Find an approximation to $\sqrt[3]{25}$ correct to within 10^{-2} using bisection method.

Solution: Consider to solve $f(x) = x^3 - 25 = 0$ by the Bisection method.

By trial and error, we can choose $a_1 = 2, b_1 = 3$.

Because $f(a_1) \cdot f(b_1) < 0$.

We need n such that $(^1/_2)^n(b_1 - a_1) \leq 10^{-2}$, solve for n → $n = 7$ iterations is needed. So, we need to compute $p_1, p_2, \dots, p_7$.

Algorithm 2.1

INPUT **a,b**; tolerance **TOL**; maximum number of iterations **N0**.

OUTPUT solution p or message of failure.

STEP1 Set i = 1;
FA = f(**a**);

STEP2 While i ≤ **N0** do STEPs 3-6.

STEP3 Set **p = a + (b-a)/2**; // a good way of computing middle point
FP = f(**p**).

STEP4 IF FP = 0 or (**b-a**) < TOL then
OUTPUT (**p**);
STOP.

STEP5 Set i = i +1.

STEP6 If FP·FA > 0 then
Set **a** = p;
FA = FP.
else
set **b** = p;

STEP7 OUTPUT(“Method failed after N0 iterations”);
STOP.

Matlab DEMO

- Apply **ALG 2.1** to solve the following problem (**Example 1** in page 50) :

Use bisection method to find a root of $f(x) = x^3 + 4\,x^2 - 10$ on the interval $[1,2]$, take accuracy tolerance to be 10^{-4}.

The complete code is also provided in the course webpage. (see next page for a matlab snippet)

MATLAB snippet for ALG 2.1 (Bisection method)

see complete code in course webpage

```
% -------------- inputs -------------------
f = @(x) (x+4)*x*x-10;
a = 1; b = 2;
% tolerance / max iter
TOL = 1e-4; NI = 50;
% ---------------------------------------------------

% STEP 1: initialization
i = 1;
fa = f(a);
converge = false; % convergence flag
% STEP 2: iteration
while i<=NI
    % STEP 3: compute p at the i's step
    p = a+(b-a)/2;
    fp = f(p);
    % STEP 4: check if meets the stopping criteria
    if (abs(fp)<eps || (b-a)/2 < TOL) % eps is Matlab-machine zero
        converge = true; % bisection method converged!
        break; % exit out of while loop
    else
        % STEP 5
        i = i+1;
        % STEP 6
        if fa*fp > 0
            a = p;  fa = fp;
        else
            b = p;
        end
    end
end
```