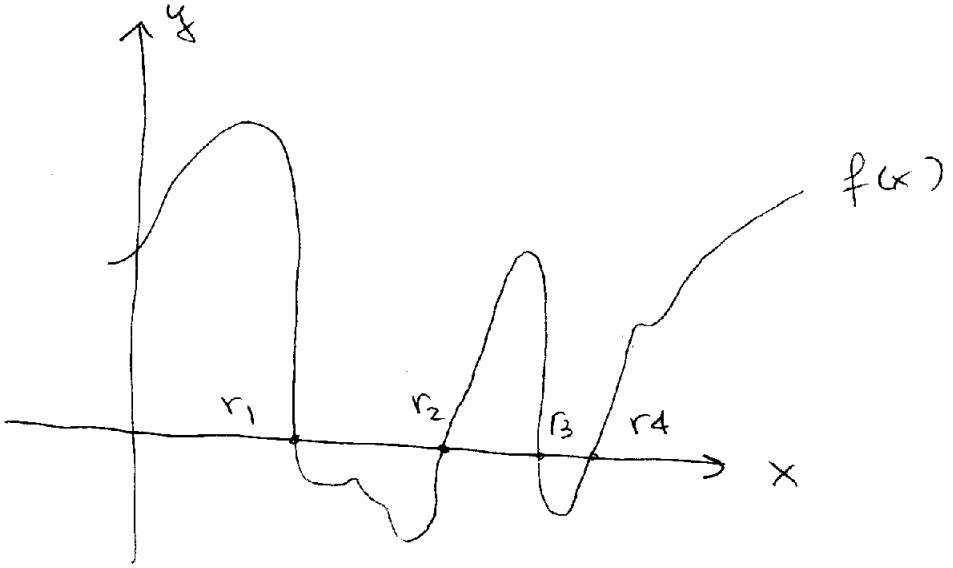
II. Solutions of nonlinear equations $f(x) = 0$ (root finding)

Problem: find the root of the function $f(x)$, i.e., the solution of an equation of the form $f(x) = 0$



1. Bisection method

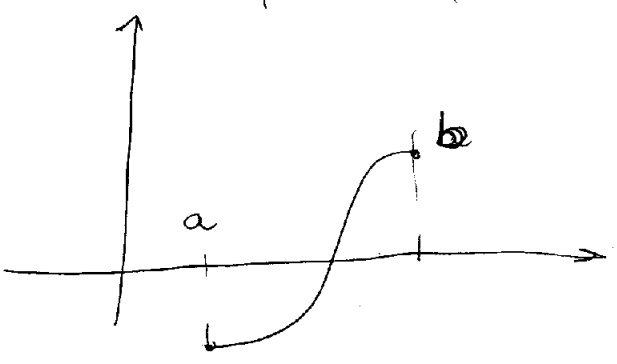
Suppose $f(x)$ is a continuous function defined in the interval $[a,b]$, with $f(a)$ and $f(b)$ of opposite signs. Then, there exists $x = p$ in $[a,b]$ so that $f(p) = 0$

Task: find p

Method: • We will do a repeated halving of subintervals of $[a,b]$

• At each step, we will locate the half containing p. (i.e., check if sign of $f(x)$ is different)

("Do" loop(s) + If statement)



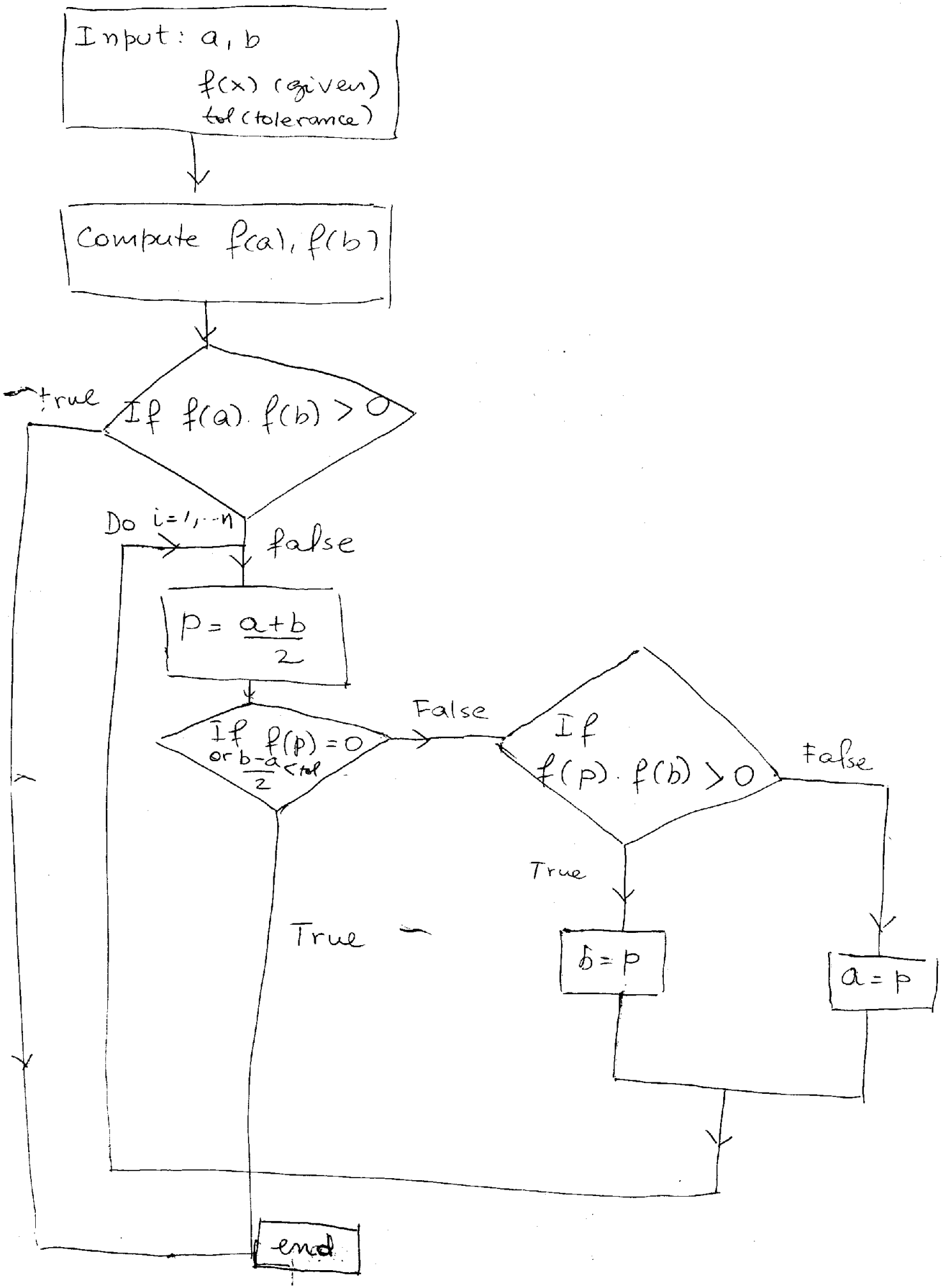
• Step 1: Compute $f(a)$
take midpoint: $p_1 = \frac{a+b}{2}$

• Step 2: Divide $[a,b]$ in $[a,p_1]$ and $[p_1,b]$

• Step 3: Check signs of $f(x)$ in each interval

• Step 4: take the subinterval where sign changed and repeat everything

Flow chart :


Input : a, b
f(x) (given)
tol (tolerance)
Compute f(a), f(b)
If f(a). f(b) > 0
true
Do i = 1, ... n
false
p = a+b / 2
If f(p) = 0 or b-a/2 < tol
False
If f(p). f(b) > 0
false
True
True
b = p
a = p
end

Algorithm

Input: endpoints a / b ; tolerance TOL ; maximum number of iterations No

Output: approximate solution p or message of failure

Step 1: Set $i = 1$
$FA = f(a)$

Step 2: While $i \leq No$ do Steps 3-6

Step 3: Set $p = a + (b-a)/2$
$FP = f(p)$

Step 4: If $FP = 0$ or $(b-a)/2 < TOL$ then
Output(p) (procedure completed successfully)
Stop

Step 5: Set $i = i+1$

Step 6: If $FA \cdot FP > 0$ then set $a = p$
$FA = FP$
else set $b = p$

Step 7: Output ('Method failed after No iterations, No = ', No)
(The procedure was unsuccessful)
STOP

(Note that here we started " from the left')

Other stopping procedures

One iterates until one of the following conditions is met:

$$|p_N - p_{N-1}| < \epsilon$$

$$\frac{|p_N - p_{N-1}|}{|p_N|} < \epsilon \qquad p_N \neq 0 \text{ or}$$

$$f(p_N) < \epsilon$$

$\epsilon \equiv$ tolerance ; $p_N \equiv$ midpoint at the N-th interation

$p_{N-1} =$ " " " " $N-1$-th iteration

- DRAWBACKS: It is slow to converge (one needs a larger number of iterations until $|p - p_N|$ is sufficiently small)

Suppose that $f \in C[a,b]$ and that $f(a) \cdot f(b) < 0$

The bisection method generates a sequence approximating a zero p of f with

$$|p - p_n| \leq \frac{b-a}{2^n}$$

For instance, for a bisection in the interval $[1,2]$ and after 9 interactions, $|p - p_9| \leq \frac{2-1}{2^9} \approx 2 \times 10^{-3}$

(*) Please note: this is a BOUND and the actual error may be much smaller

- May throw away intermediate results better than the final one.

Example

Find the root of $f(x) = 2x^3 - x^2 + x - 1$ in the interval $[-2, 2]$ so that the tolerance is 0.1

(root at $x = 0.738984$)

- Check is there a root in this interval?

$f(-2) = -16 - 4 - 2 - 1 = -23$

$f(2) = 16 - 4 + 2 - 1 = 13$

$\rightarrow f(2) \cdot f(-2) < 0$

there is a root!

However, there is also a root in $[0, 1]$:

$f(0) = -1$

$f(1) = 1$

$\rightarrow f(0) \cdot f(1) < 0$

(*) Please note: it is an advantage to start with an interval as small as possible

- Tolerance = 0.1 means the program will stop for $\frac{b-a}{2} < 0.1$

So we take $a = 0$

$b = 1$

Bisection procedure: • Take the midpoint $p = \frac{a+b}{2} = 0.5$

Compute $f(0.5) = 2 \cdot (0.5)^3 - 0.25 + 0.5 - 1 = -0$

$f(0.5) \cdot f(1) < 0 \rightarrow$ The root is in $[0.5, 1]$

take $a = 0.5$

$b = 1.0$

$\frac{b-a}{2} = 0.25 >$ Tolerance

Compute the midpoint $p = \frac{a+b}{2} \Rightarrow p = 0.75$

$f(0.75) = 0.03125 > 0 \Rightarrow$ the root is in $[0.5, 0.$

Take $a = 0.5$

$b = 0.75$

$\frac{b-a}{2} = 0.125 >$ Tolerance

Midpoint : $\frac{0.5 + 0.75}{2} = 0.625$

$f(0.625) = -0.2773 < 0 \Rightarrow$ the root is in $[0.625,$

$a = 0.625$
$b = 0.75$

$\frac{b-a}{2} = 0.0625 < TOL$

$\Rightarrow$ this computation will be the last

Midpoint : $\frac{0.625 + 0.75}{2} = 0.6875$

$f(0.6875) = -0.135254$

The answer would be : root = 0.6875

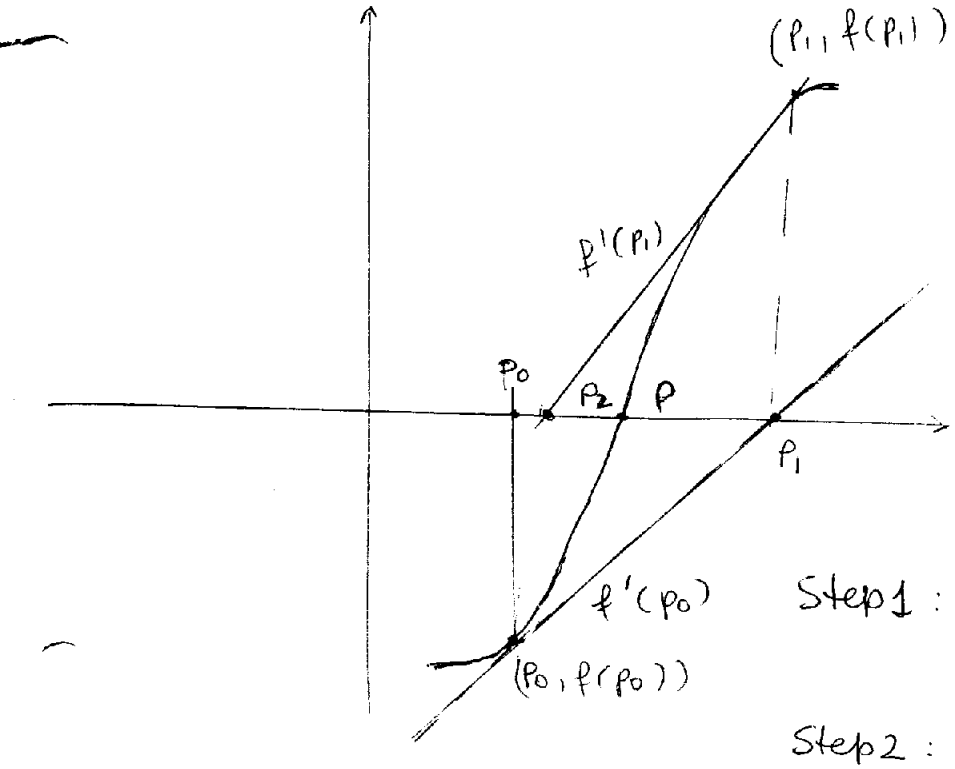
(*) Please note :

- This is converging very slowly
- We "threw away" an intermediate value (root = 0.75) which was much better.

2 : - Newton-Raphson method

- One of the most powerful methods for finding a root of a function
- Converges faster than the bisection method
- Relies on the continuity of the first and the second derivatives of a function $f(x)$

GRAPHICAL DESCRIPTION



Step 1 : Start from one "guess root" p_0, close to p

Step 2 : Draw a tangent to the curve $f(x)$ at p_0

(Remember that the derivative $f'(p_0)$ gives the slope of the tangent to the curve $f(x)$ at p_0)

Step 3 : Take the interception of the tangent with the x axis as a new "guess root" p_1

Step 4 : Repeat the procedure until root is found

• Slope of the tangent to the curve at p_0:

(passing through $(p_1, 0)$ and $(p_0, f(p_0))$

$$m = \frac{0 - f(p_0)}{p_1 - p_0} \quad (*)$$

• On the other hand, $m = f'(p_0)$ (*)

Hence, $f'(p_0) = \frac{-f(p_0)}{p_1 - p_0}$

$$\boxed{p_1 = p_0 - \frac{f(p_0)}{f'(p_0)}}$$

$\Rightarrow$ One is constructing a sequence

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}$$

$p_n \equiv$ root for the n^{th} iterat

$p_{n-1} \equiv$ root for the $n-1^{th}$ it

that converges to p.

(*) Please note: the function $g(x)$ defined by the formula $g(x) = x - \frac{f(x)}{f'(x)}$ is called the "Newton-Raphson" iteration function. At the root, $f(p) = 0$ so that $g(p) = p$.

Then, finding the root $f(x) = 0$ is accomplished by finding a fixed point of g

(*) Question: why does p_0 need to be near p?

Answer: Because we are using the first-order Taylor expansion for $f(x)$:

Let us consider

p_0 → approximate root

p → exact root

$f(x)$ → function

Taylor expansion of $f(x)$ around p_0 :

$$f(x) = f(p_0) + (x-p_0) f'(x)\Big|_{x=p_0} + \frac{(x-p_0)^2}{2!} f''(x)\Big|_{x=p_0} + \cdots$$

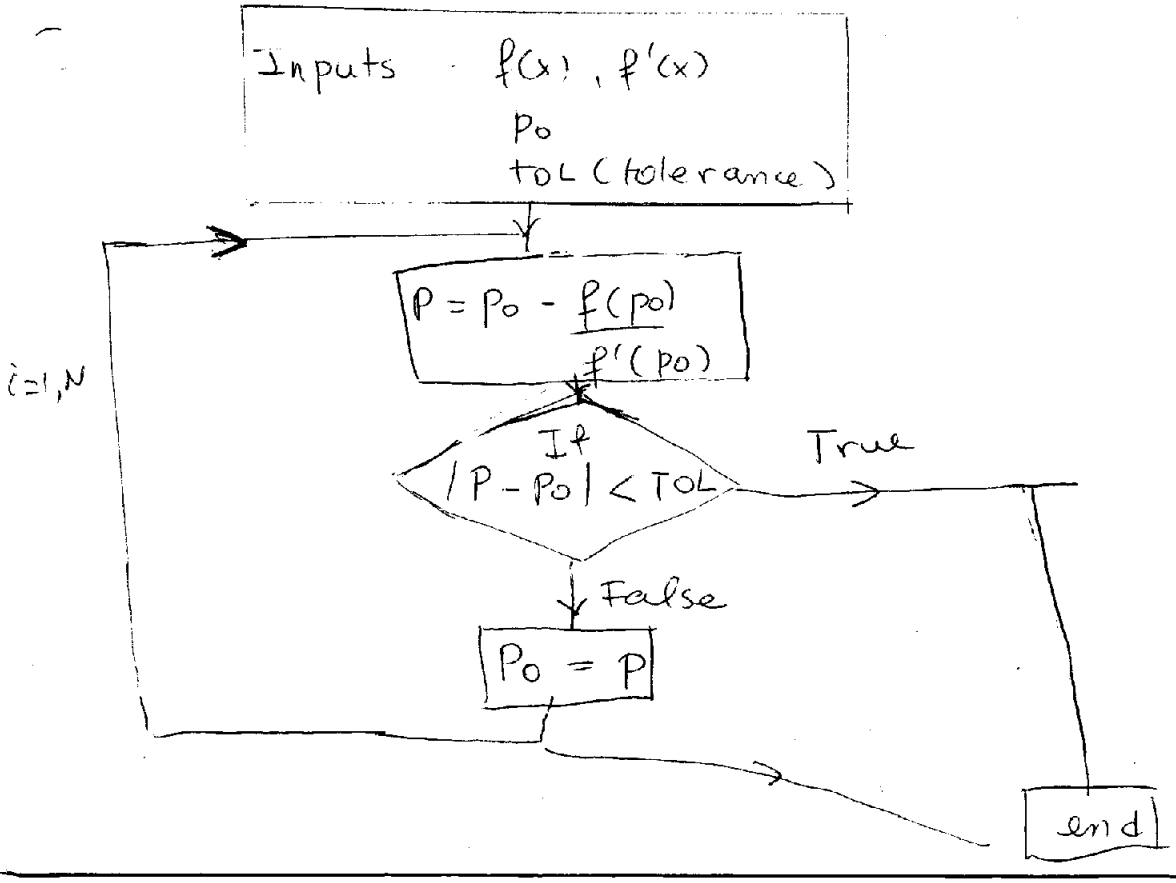
Root: $f(p) = 0$

$$0 = f(p_0) + (p-p_0) f'(x)\Big|_{x=p_0} + \frac{(x-p_0)^2}{2!} f''(x)\Big|_{x=p_0} + \cdots$$

If p near p_0 $\Rightarrow 0 \simeq f(p_0) + (p-p_0) f'(x)\Big|_{x=p_0}$

$$\boxed{p \simeq p_0 - \frac{f(p_0)}{f'(p_0)}}$$

Flow chart



Algorithm:

Input: Initial approximation p_0, Tolerance TOL, maximum number of iterations N_0

Output: Approximate solution p or message of failure.

Step 1: Set $i = 1$

Step 2 While $i \leq N_0$ do steps 3 - 6

Step 3 Set $p = p_0 - \frac{f(p_0)}{f'(p_0)}$

Step 4 If $|p - p_0| < TOL$ then
Output(p),
Stop

Step 5 Set $i = i + 1$

Step 6 Set $p_0 = p$

Step 7: Output ('The method failed after N_0 interactions, $N_0 =$ ', N_0)
Stop

DRAWBACKS:

- Fails if $f'(p_0) = 0$
- One needs to know $f'(x)$ at each iteration steps
- p_0 needs to be near p (for this reason this method is very appropriate for REFINING the roots found with other methods)
- Fails if a function has no real roots (e.g. $f(x) = x^2 + 4x + 5$)

- Convergene

  - Simple roots

$$|P - P_{n+1}| \approx \frac{|f''(P)|}{2|f'(P)|} |P - P_n|^2$$

for n sufficiently large

(the error in each successive iteration is proportional to the square of the error in the previous iteration).

  - Multiple root (order M)

$$|P - P_{n+1}| \approx \frac{M-1}{M} |P - P_n|$$

Example

- Simple root.

$x_1 = +1$
and
$x_2 = -1$
are simple roots of $f(x) = x^2 - 1 = 0$

- Double (multiple root:

1 is a double root of

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

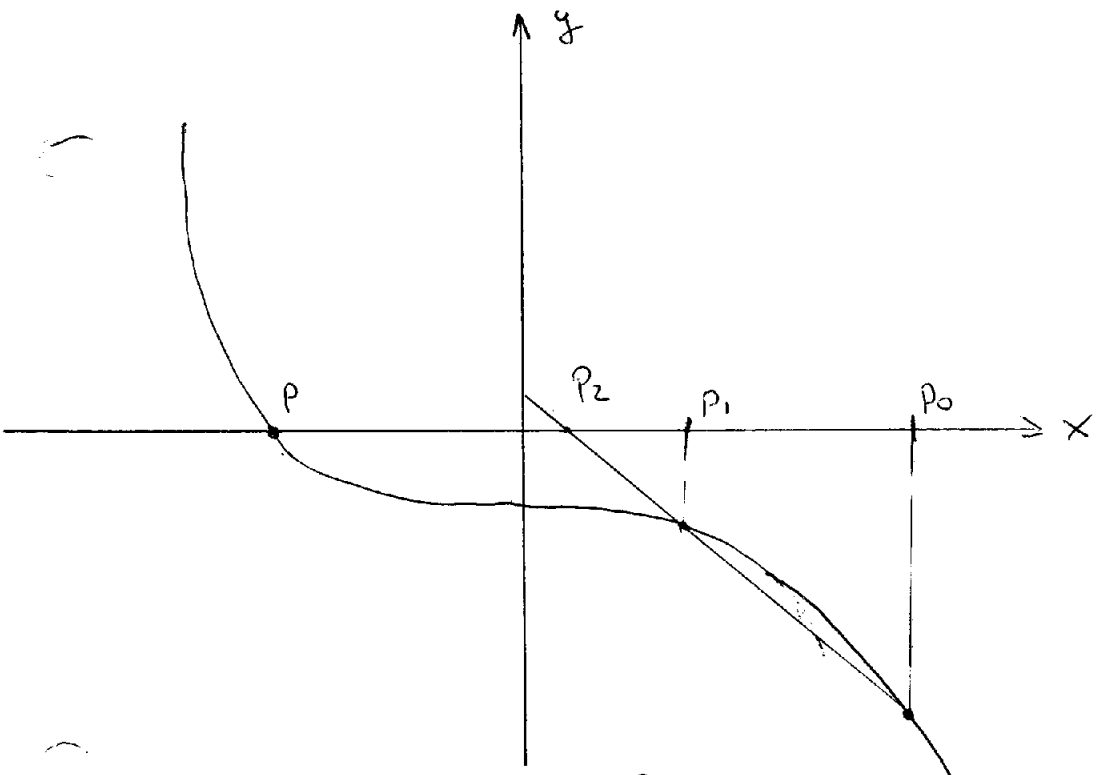
and a triple root of

$$f(x) = (x-1)^3$$

3. Secant method

- Converges almost as rapidly as Newton-Raphson's method
- Does not require a pre-knowledge of the derivative of $f(x)$

GRAPHICAL DESCRIPTION



Step 1: take <u>two</u> initial points $(P_0, f(P_0))$ and $(P_1, f(P_1))$ near a root $(P, 0)$

Step 2: Draw a ~~Secant~~ to the curve passing through these two points

Step 3: Take the interception of the secant with the x axis (P_2)

Step 4: repeat the procedure with

$P_1 \rightarrow P_2$

$P_0 \rightarrow P_1$

- Slope of the secant line through $(p_1, f(p_1))$ and $(p_0, f(p_0))$:

$$m = \frac{f(p_1) - f(p_0)}{p_1 - p_0} \quad (*)$$

- Slope of the secant line through $(p_1, f(p_1))$ and $(p_2, 0)$

$$m = \frac{0 - f(p_1)}{p_2 - p_1} \quad (**)$$

$(*) = (**) \Rightarrow$

$$\frac{f(p_1) - f(p_0)}{p_1 - p_0} = -\frac{f(p_1)}{p_2 - p_1}$$

$$p_2 = p_1 - \frac{f(p_1)(p_1 - p_0)}{f(p_1) - f(p_0)} = g(p_1, p_0)$$

In general,

$$p_{k+1} = g(p_k, p_{k-1}) = p_k - \frac{f(p_k)(p_k - p_{k-1})}{f(p_k) - f(p_{k-1})}$$

- This procedure is very similar to that adopted in the Newton-Raphson method, with the main difference that one now takes the SECANT instead of the TANGENT

p_k = p at the k^{th} iteration
p_{k+1} = p at the $k+1^{st}$ iteration
p_{k-1} = p at the $k-1^{th}$ iteration

- Roughly speaking, this can be viewed as being an additional approximation for the derivative of a function near an approximate root

Convergence
- simple roots

$$|p - p_{n+1}| = \left|\frac{f''(p)}{2f'(p)}\right|^{0.618} \times |p - p_n|^{1.618}$$

Where $R = (1+\sqrt{5})/2 \approx 1.618$

SUMMARY

Method	Convergence
Bisection	$\|p - p_{n+1}\| \approx \frac{1}{2}\|p - p_n\|$
Newton-Raphson	$\|p - p_{n+1}\| \approx A\|p - p_n\|^2$ (simple root)
	$\|p - p_{n+1}\| \approx A\|p - p_n\|$ (Multiple root)
Secant	$\|p - p_{n+1}\| \approx A\|p - p_n\|^{1.618}$ (simple root)

(*) Please note: There are built-in functions in many platforms to find roots of functions.

Examples: Mathematica: FindRoot[lhs == rhs, {x, x0}]

(searches for a numerical solution to the equation lhs == rhs starting with $x = x_0$)

FindRoot[lhs == rhs, {x, x0, x1}]

(Does the same as above within the interval $[x_0, x_1]$)

r = roots(c) finds roots of a polynomial

p = [1 -6 -72 -27]

r = roots(p)

r = 12.1229
-0.7345
-0.3884

(14)

X = fzero (@ function, x0)

Example : fzero(@(x) 2*x^3 - x^2 + x - 1, 0.75)

ans = 0.7390

(mixes bisection, secant, and inverse interpolation methods)

Example : p = [2 -1 1 -1]

r = roots(p)