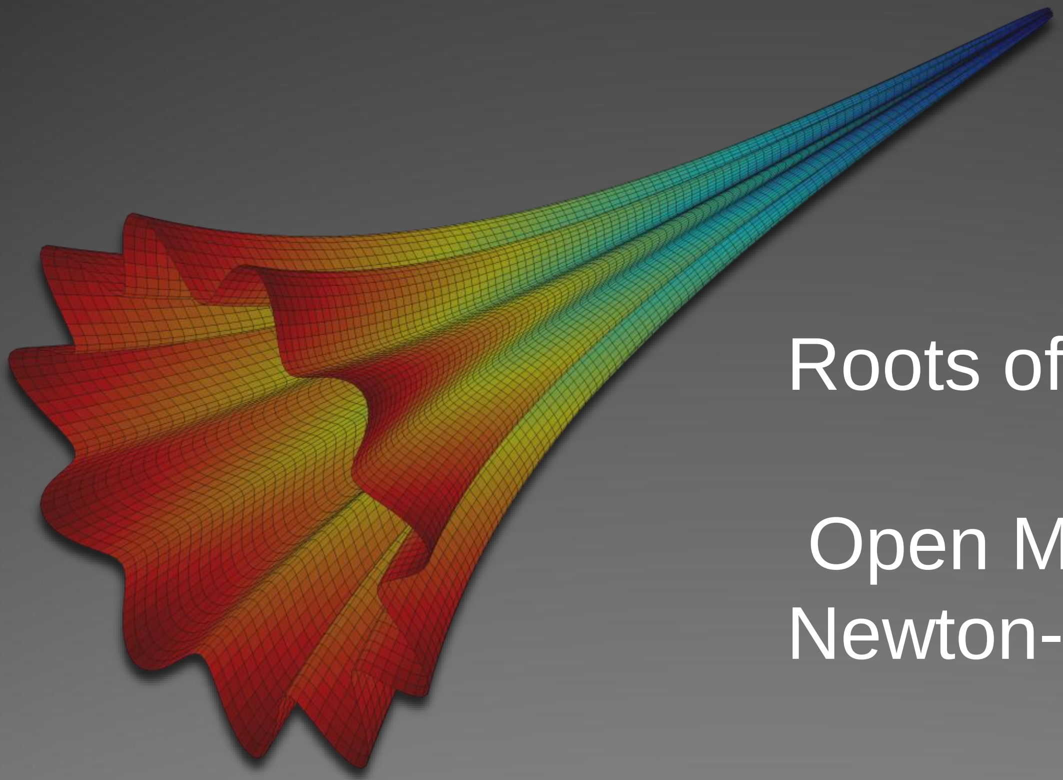
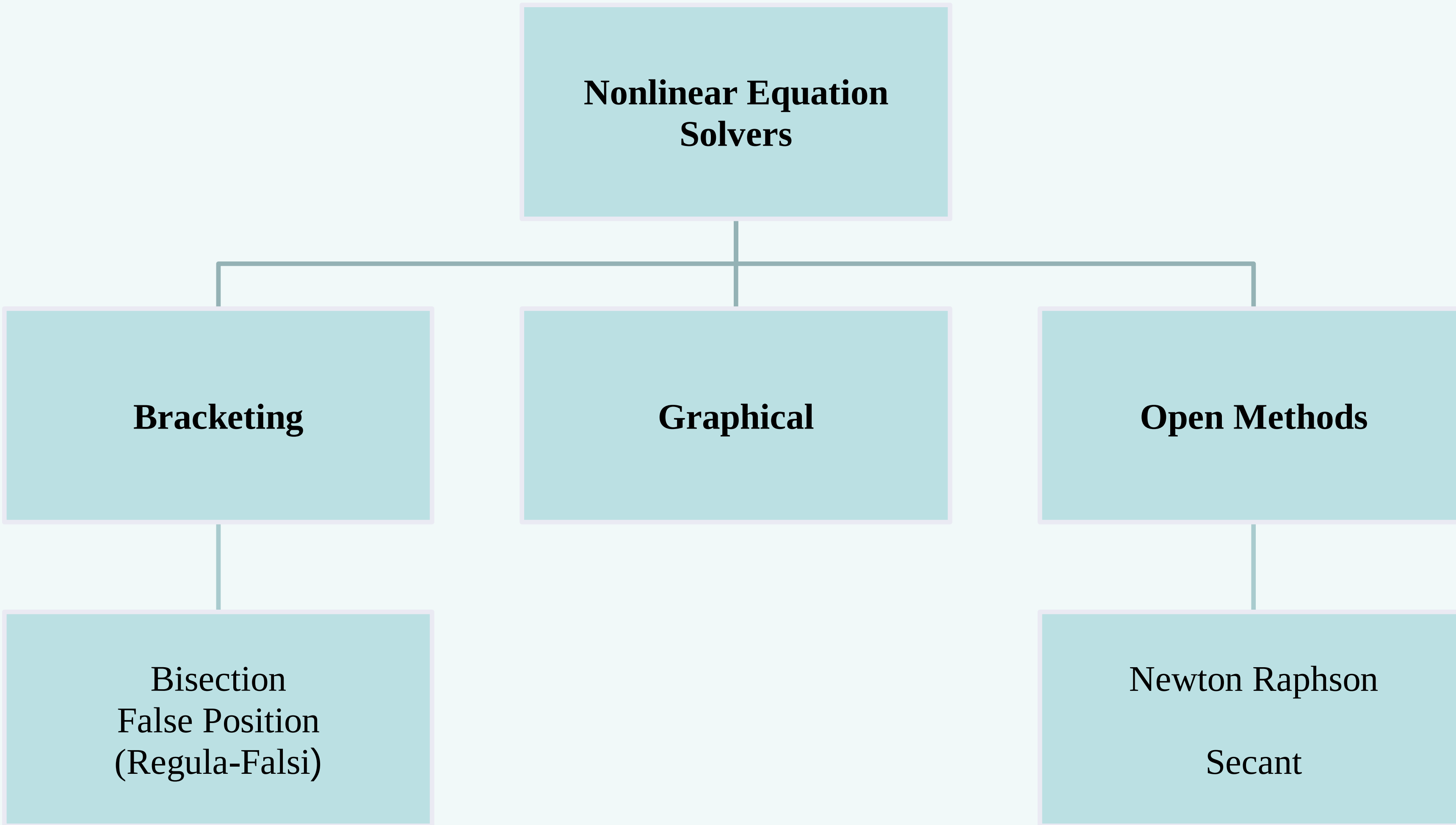


Roots of Equations

Open Methods
Newton-Raphson

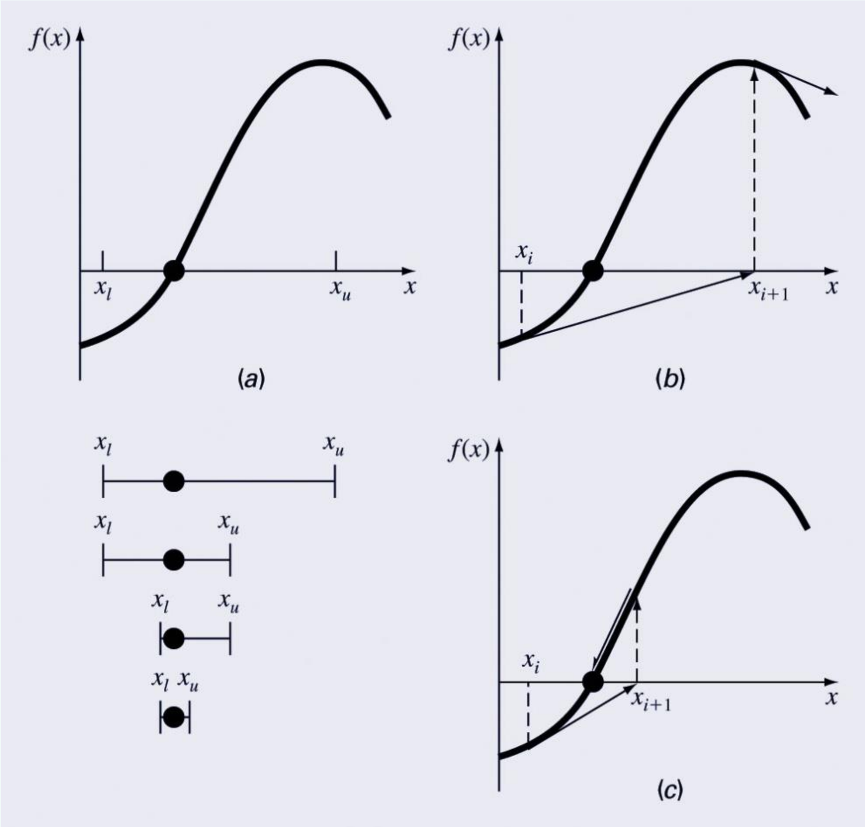


All Iterative

Open Methods

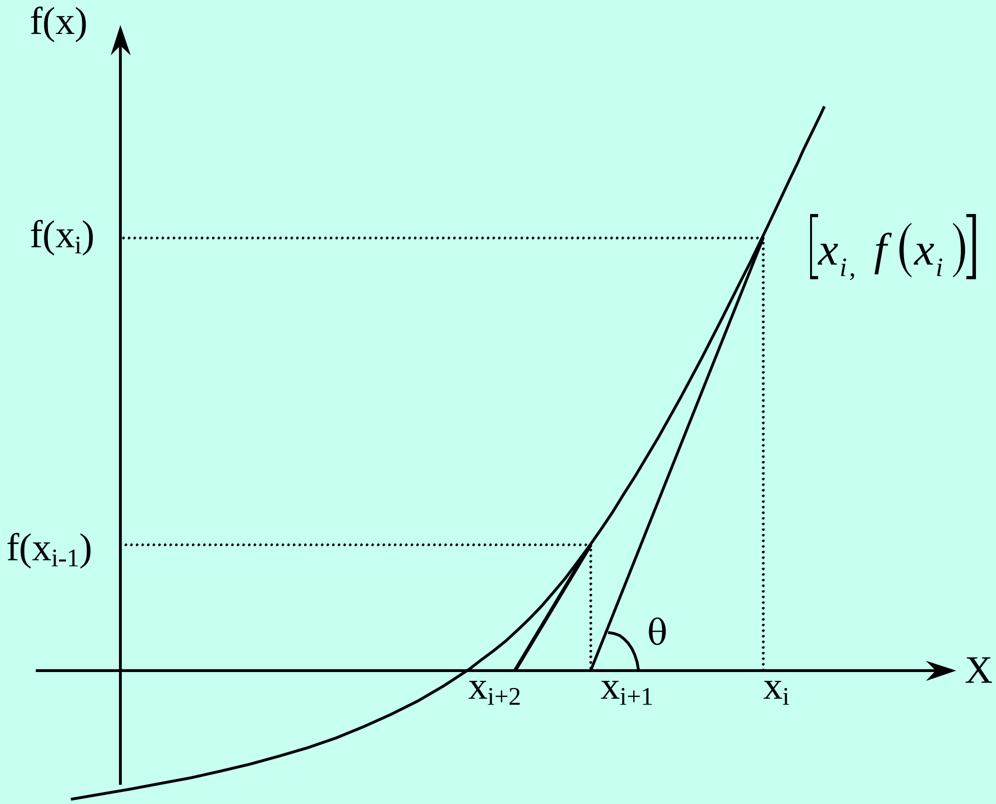
- *Open methods* differ from bracketing methods, in that open methods require only a single starting value (NR).
- Used in computer programs today to solve extremely complicated equations
- Open methods may diverge as the computation progresses, but when they do converge, they usually do so much faster than bracketing methods.

Graphical Comparison of Methods



a) Bracketing method
b) Diverging open method
c) Converging open method - note speed!

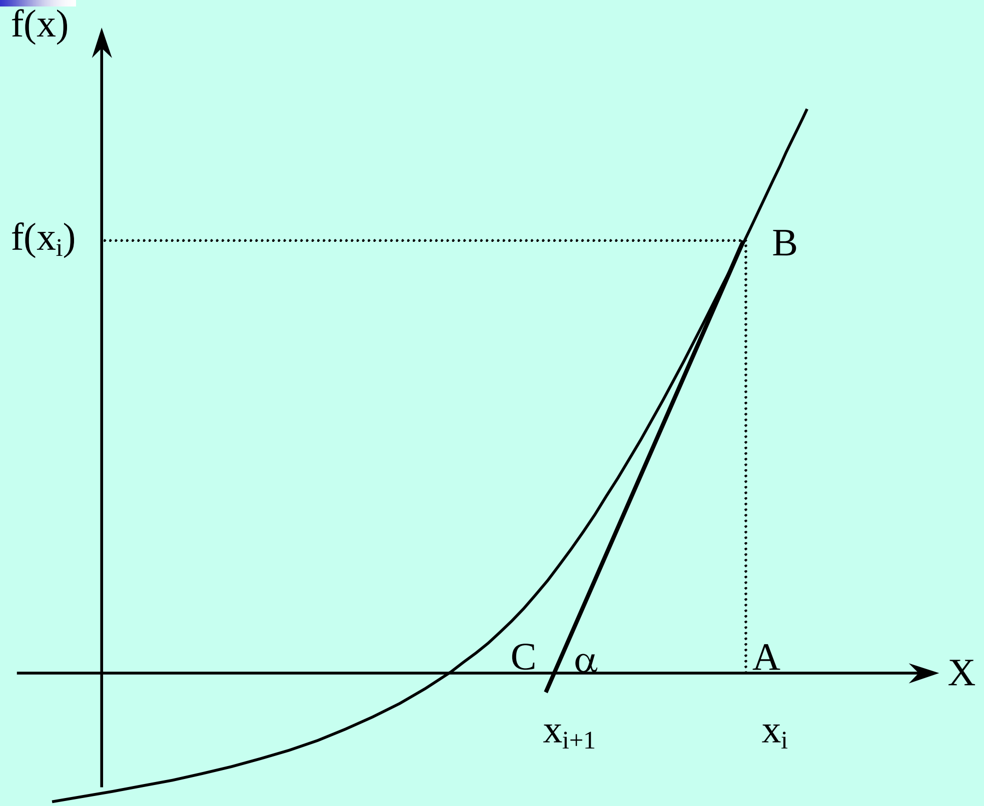
Newton-Raphson Method



$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Figure 1 Geometrical illustration of the Newton-Raphson method.

Derivation



$$\tan(\alpha) = \frac{AB}{AC}$$

$$f'(x_i) = \frac{f(x_i)}{x_i - x_{i+1}}$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Figure 2 Derivation of the Newton-Raphson method.

Algorithm for Newton-Raphson Method

Step 1

Evaluate $f'(x)$ symbolically.

Step 2

Use an initial guess of the root, x_i , to estimate the new value of the root, x_{i+1} , as

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

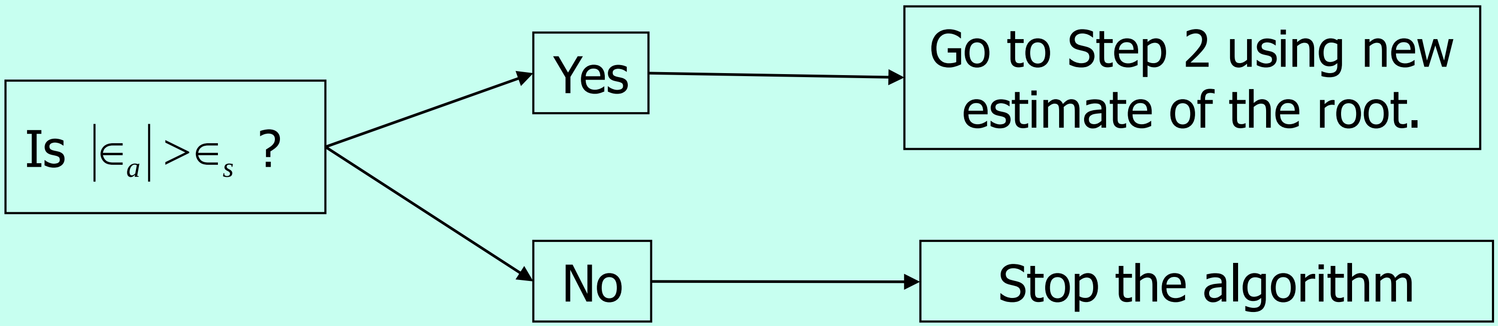
Step 3

Find the absolute relative approximate error $\left|\in_a\right|$ as

$$\left|\in_a\right| = \left|\frac{x_{i+1} - x_i}{x_{i+1}}\right| \times 100$$

Step 4

Compare the absolute relative approximate error with the pre-specified relative error tolerance $\in_s$.



Also, check if the number of iterations has exceeded the maximum number of iterations allowed. If so, one needs to terminate the algorithm and notify the user.

Example: $\varepsilon_s = 0{,}05\%$

$x^3 = 20$

$x_0 = 3{,}0$

Conduct 3 iterations (for simplicity).

Sol: $f(x) = 0$ we need to rewrite our fct

$f(x) = x^3 - 20 = 0$

General formula:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

we need the derivative:

$f(x) = x^3 - 20 \rightarrow f'(x) = 3x^2$.

$i = 0$:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

$$= 3 - \frac{3^3 - 20}{3(3)^2}$$

$$x_1 = 2.741$$

Calculate: Relative App Error:

$$|\varepsilon_a| = \left|\frac{2.741 - 3.0}{2.741}\right| \times 100$$

$$= 9.45\ \%$$

$\underline{i = 1}$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

$$= x_1 - \frac{x_1^3 - 20}{3x_1^2}$$

$$= 2,741 - \frac{2,741^3 - 20}{3(2,741)^2}$$

$$= 2,715$$

$$|\varepsilon_a| = \left| \frac{2,715 - 2,741}{2,715} \right| 100$$

$= 0,96\%$ ⟸ from 9,45 to 0,96%

when this method converge it does so fast.

$i = 2$

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)}$$

$$= x_2 - \frac{x_2^3 - 20}{3x_2^2} = 2,715 - \frac{2,715^3 - 20}{3(2,715)^2}$$

$$= 2,714$$

$$|\varepsilon_a| = \left| \frac{2,714 - 2,715}{2,714} \right| \times 100 = 0,009\%$$

9,45% → 0,96% → 0,009% in 3 iteration only

Example 1

You are working for 'DOWN THE TOILET COMPANY' that makes floats for ABC commodes. The floating ball has a specific gravity of 0.6 and has a radius of 5.5 cm. You are asked to find the depth to which the ball is submerged when floating in water.

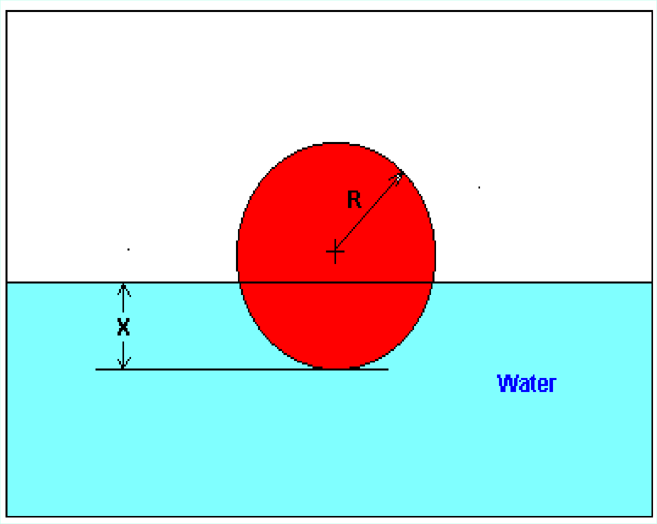


Figure 3 Floating ball problem.

Example 1 Cont.

The equation that gives the depth *x* in meters to which the ball is submerged under water is given by

$$f(x) = x^3 - 0.165x^2 + 3.993 \times 10^{-4}$$

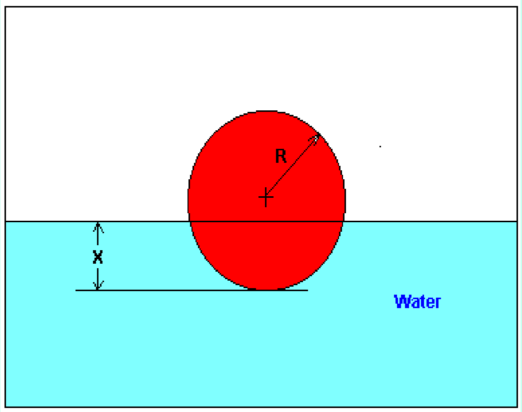


Figure 3 Floating ball problem.

Use the Newton's method of finding roots of equations to find

a) the depth 'x' to which the ball is submerged under water. Conduct three iterations to estimate the root of the above equation.

b) The absolute relative approximate error at the end of each iteration, and

c) The number of significant digits at least correct at the end of each iteration.

Example 1 Cont.

Solution

To aid in the understanding of how this method works to find the root of an equation, the graph of f(x) is shown to the right,

where

$$f(x) = x^3 - 0.165x^2 + 3.993\times10^{-4}$$

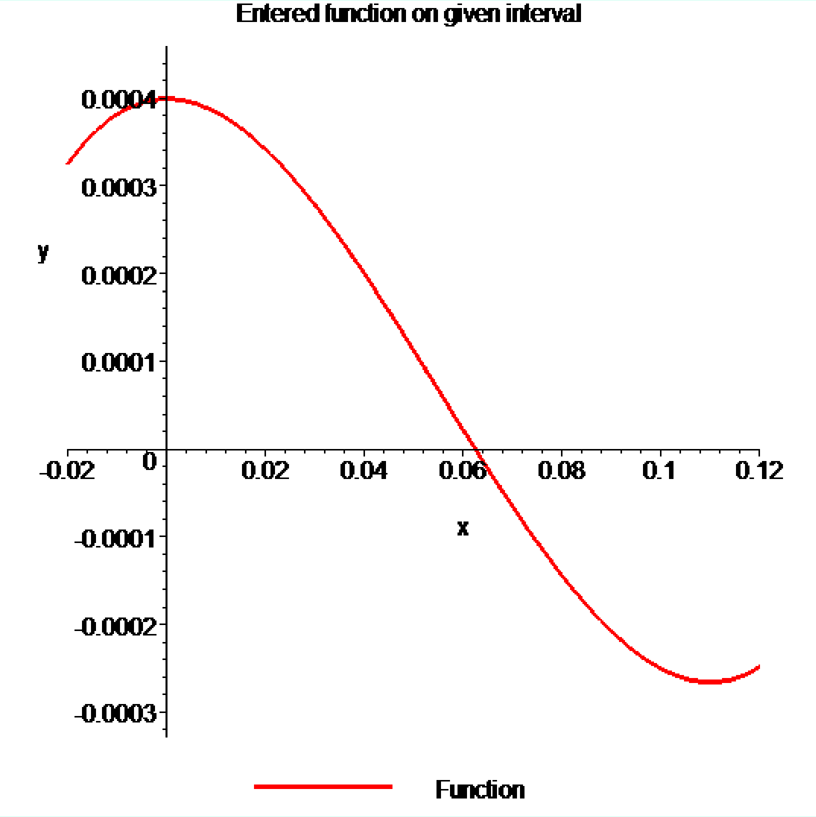


Figure 4 Graph of the function f(x)

Example 1 Cont.

Solve for $f'(x)$

$$f(x) = x^3 - 0.165x^2 + 3.993\times10^{-4}$$

$$f'(x) = 3x^2 - 0.33x$$

Let us assume the initial guess of the root of $f(x)=0$ is $x_0 = 0.05\text{m}$. This is a reasonable guess (discuss why $x=0$ and $x = 0.11\text{m}$ are not good choices) as the extreme values of the depth x would be 0 and the diameter (0.11 m) of the ball.

Example 1 Cont.

<u>Iteration 1</u>

The estimate of the root is

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

$$= 0.05 - \frac{(0.05)^3 - 0.165(0.05)^2 + 3.993 \times 10^{-4}}{3(0.05)^2 - 0.33(0.05)}$$

$$= 0.05 - \frac{1.118 \times 10^{-4}}{-9 \times 10^{-3}}$$

$$= 0.05 - (-0.01242)$$

$$= 0.06242$$

Example 1 Cont.

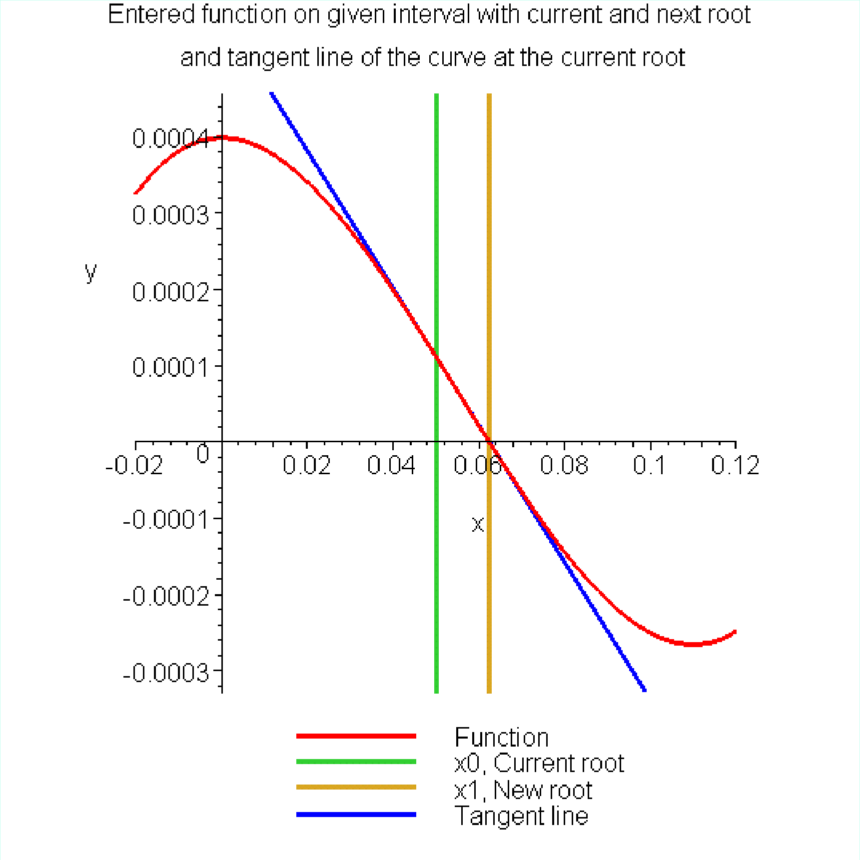


Figure 5 Estimate of the root for the first iteration.

Example 1 Cont.

The absolute relative approximate error $\left|\in_a\right|$ at the end of Iteration 1 is

$$\left|\in_a\right| = \left|\frac{x_1 - x_0}{x_1}\right| \times 100$$

$$= \left|\frac{0.06242 - 0.05}{0.06242}\right| \times 100$$

$$= 19.90\%$$

The number of significant digits at least correct is 0, as you need an absolute relative approximate error of 5% or less for at least one significant digits to be correct in your result.

Example 1 Cont.

<u>Iteration 2</u>

The estimate of the root is

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

$$= 0.06242 - \frac{(0.06242)^3 - 0.165(0.06242)^2 + 3.993\times10^{-4}}{3(0.06242)^2 - 0.33(0.06242)}$$

$$= 0.06242 - \frac{-3.97781\times10^{-7}}{-8.90973\times10^{-3}}$$

$$= 0.06242 - (4.4646\times10^{-5})$$

$$= 0.06238$$

Example 1 Cont.

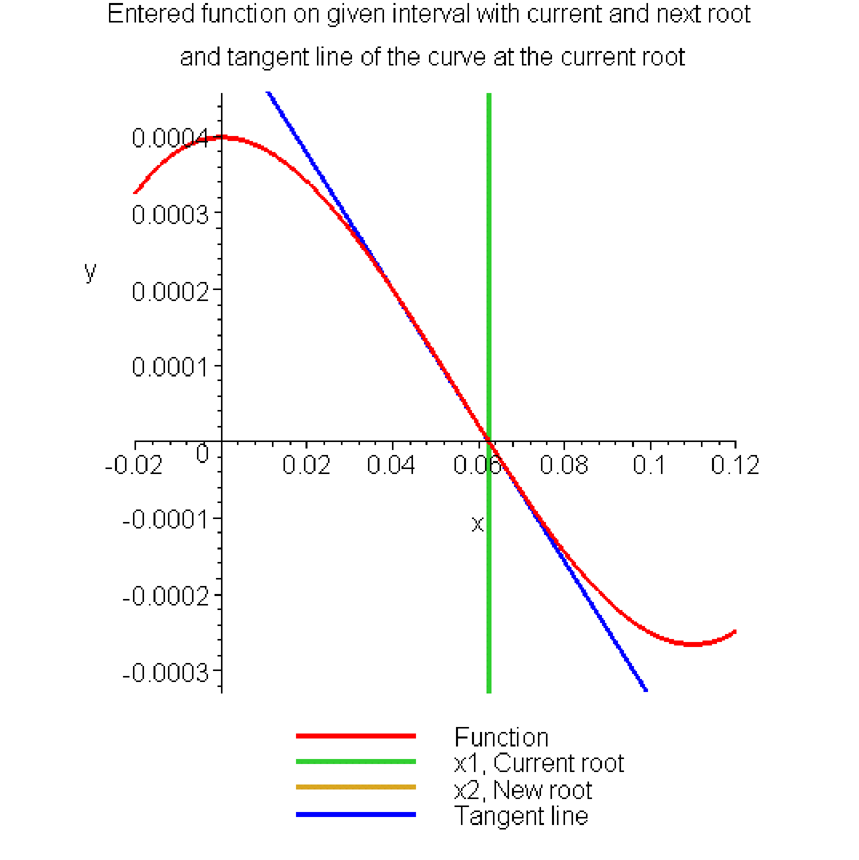


Figure 6 Estimate of the root for the Iteration 2.

Example 1 Cont.

The absolute relative approximate error $|\in_a|$ at the end of Iteration 2 is

$$|\in_a| = \left|\frac{x_2 - x_1}{x_2}\right| \times 100$$

$$= \left|\frac{0.06238 - 0.06242}{0.06238}\right| \times 100$$

$$= 0.0716\,\%$$

The maximum value of m for which $|\in_a| \leq 0.5 \times 10^{2-m}$ is 2.844. Hence, the number of significant digits at least correct in the answer is 2.

Example 1 Cont.

<u>Iteration 3</u>

The estimate of the root is

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)}$$

$$= 0.06238 - \frac{(0.06238)^3 - 0.165(0.06238)^2 + 3.993\times10^{-4}}{3(0.06238)^2 - 0.33(0.06238)}$$

$$= 0.06238 - \frac{4.44\times10^{-11}}{-8.91171\times10^{-3}}$$

$$= 0.06238 - \left(-4.9822\times10^{-9}\right)$$

$$= 0.06238$$

Example 1 Cont.

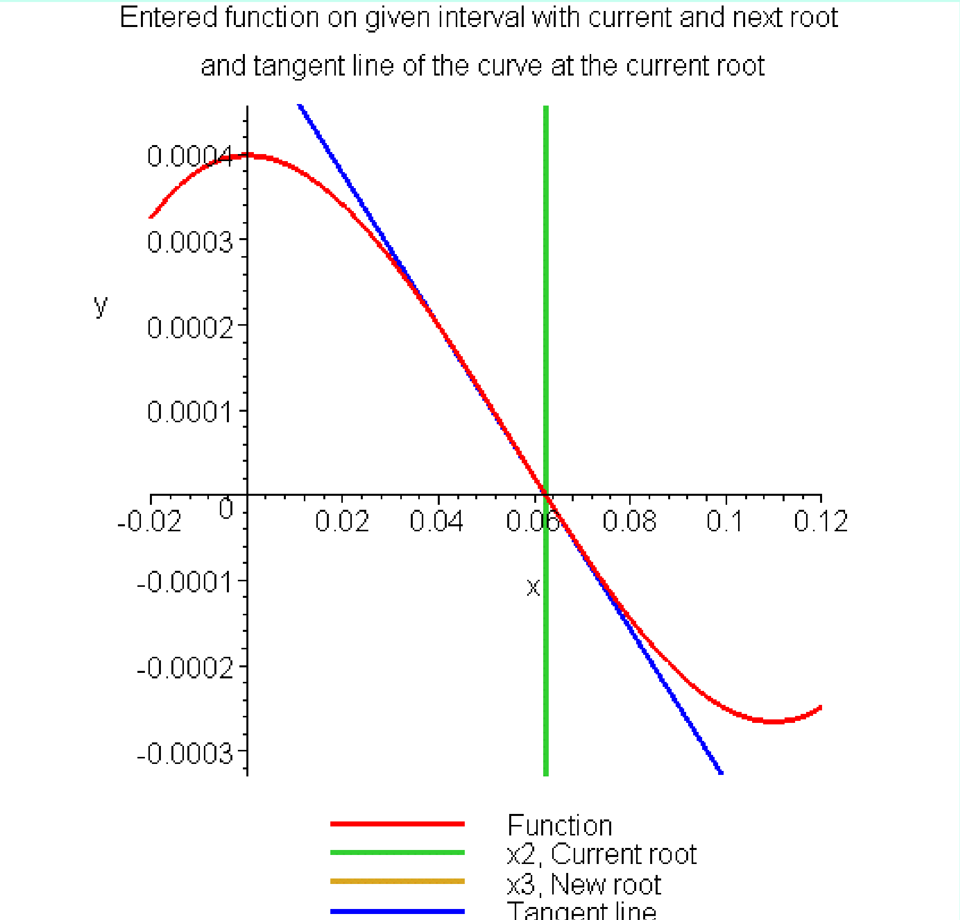


Figure 7 Estimate of the root for the Iteration 3.

Example 1 Cont.

The absolute relative approximate error $\left|\in_a\right|$ at the end of Iteration 3 is

$$\left|\in_a\right| = \left|\frac{x_2 - x_1}{x_2}\right| \times 100$$

$$= \left|\frac{0.06238 - 0.06238}{0.06238}\right| \times 100$$

$$= 0\%$$

The number of significant digits at least correct is 4, as only 4 significant digits are carried through all the calculations.

Advantages

- Converges fast (quadratic convergence), if it converges.
- Requires only one guess

Drawbacks – Oscillations near local maximum and minimum

3. <u>Oscillations near local maximum and minimum</u>

Results obtained from the Newton-Raphson method may oscillate about the local maximum or minimum without converging on a root but converging on the local maximum or minimum.

Eventually, it may lead to division by a number close to zero and may diverge.

For example for $f(x) = x^2 + 2 = 0$ the equation has no real roots.

Drawbacks – Oscillations near local maximum and minimum

Table 3 Oscillations near local maxima and mimima in Newton-Raphson method.

Iteration Number	x_i	$f(x_i)$	$\left\|\in_a\right\|\%$
0	–1.0000	3.00	
1	0.5	2.25	300.00
2	–1.75	5.063	128.571
3	–0.30357	2.092	476.47
4	3.1423	11.874	109.66
5	1.2529	3.570	150.80
6	–0.17166	2.029	829.88
7	5.7395	34.942	102.99
8	2.6955	9.266	112.93
9	0.97678	2.954	175.96

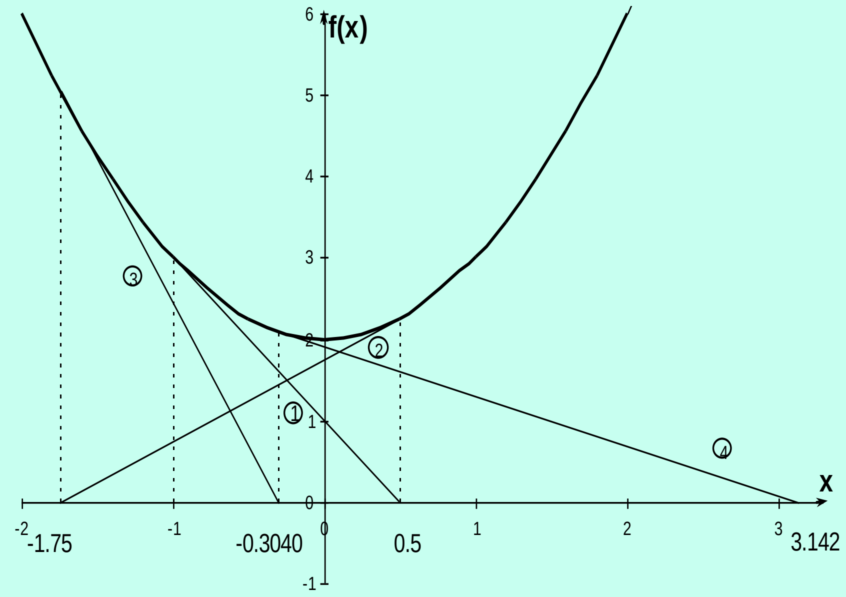


Figure 10 Oscillations around local minima for $f(x)=x^2+2$.

Drawbacks – Root Jumping

4. <u>Root Jumping</u>

In some cases where the function $f(x)$ is oscillating and has a number of roots, one may choose an initial guess close to a root. However, the guesses may jump and converge to some other root.

For example

$$f(x) = \sin x = 0$$

Choose

$$x_0 = 2.4\pi = 7.539822$$

It will converge to $x = 0$

instead of $x = 2\pi = 6.2831853$

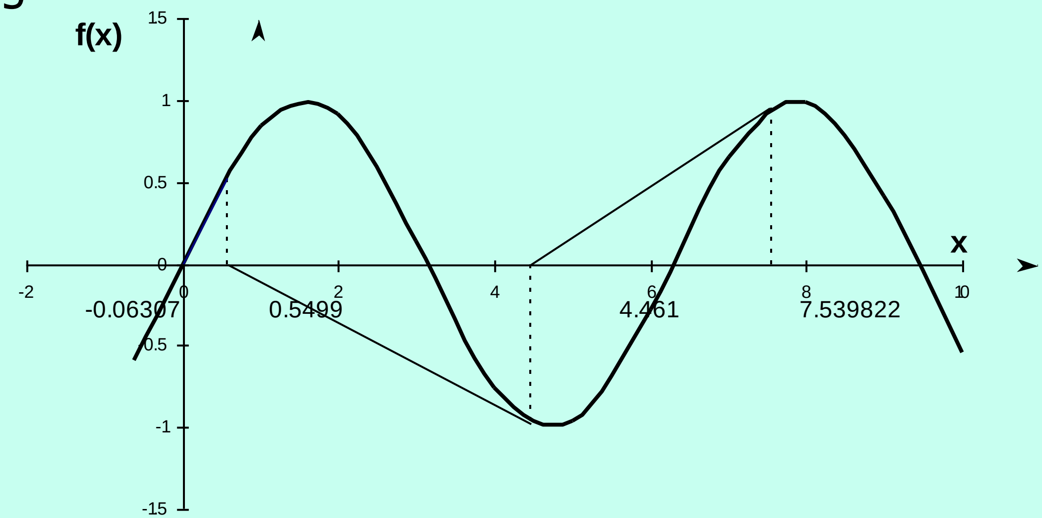
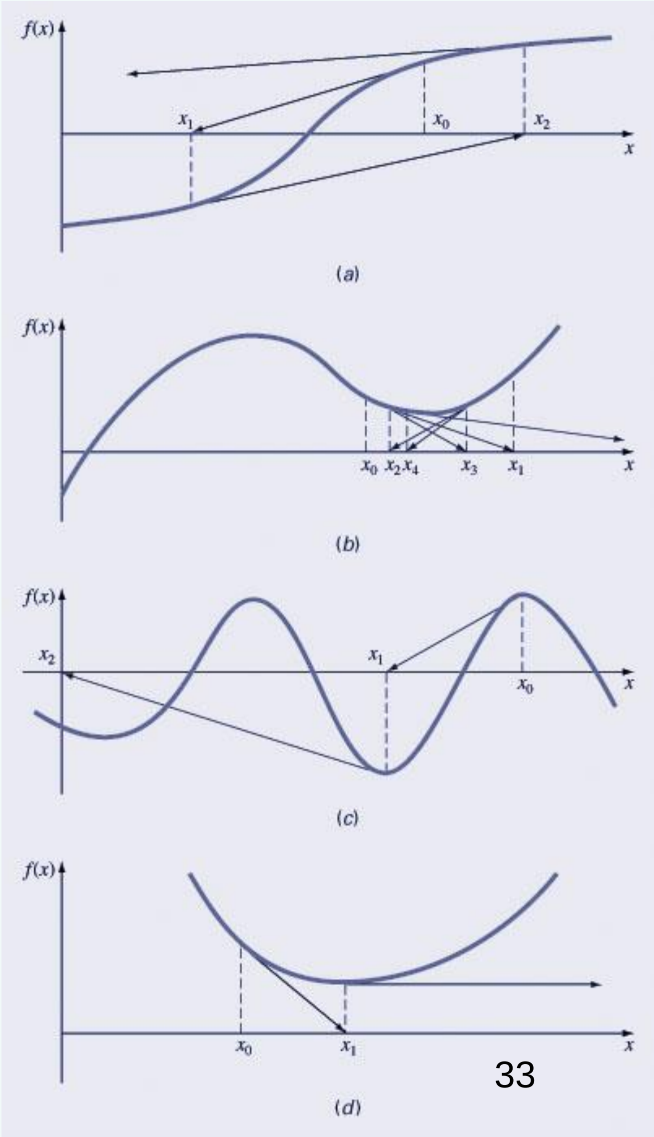


Figure 11 Root jumping from intended location of root for $f(x) = \sin x = 0$

Pros and Cons

- Pro: The error of the i+1th iteration is roughly proportional to the square of the error of the ith iteration - this is called *quadratic convergence*
- Con: Some functions show slow or poor convergence



MATLAB's `fzero` Function

- MATLAB's `fzero` provides the best qualities of both bracketing methods and open methods.
  - Using an initial guess:

    ```
    x = fzero(function, x0)
    [x, fx] = fzero(function, x0)
    ```

    - *`function`* is a function handle to the function being evaluated
    - *`x0`* is the initial guess
    - *`x`* is the location of the root
    - *`fx`* is the function evaluated at that root
  - Using an initial bracket:

    ```
    x = fzero(function, [x0 x1])
    [x, fx] = fzero(function, [x0 x1])
    ```

    - As above, except *`x0`* and *`x1`* are guesses that *must* bracket a sign change

`fzero` Options

- Options may be passed to fzero as a third input argument - the options are a data structure created by the `optimset` command
- `options = optimset(‘`par_1`’, `val_1`, ‘`par_2`’, `val_2`,…)`
  - par_n is the name of the parameter to be set
  - val_n is the value to which to set that parameter
  - The parameters commonly used with `fzero` are:
    - `display`: when set to ‘iter’ displays a detailed record of all the iterations
    - `tolx`: A positive scalar that sets a termination tolerance on x.

`fzero` Example

- `options = optimset('display', 'iter');`
  - Sets options to display each iteration of root finding process
- `[x, fx] = fzero(@(x) x^10-1, 0.5, options)`
  - Uses fzero to find roots of $f(x)=x^{10}-1$ starting with an initial guess of $x=0.5$.
- MATLAB reports `x=1, fx=0` after 35 function counts

Polynomials

- MATLAB has a built in program called `roots` to determine all the roots of a polynomial - including imaginary and complex ones.
- `x = roots(c)`
  - `x` is a column vector containing the roots
  - `c` is a row vector containing the polynomial coefficients
- Example:
  - Find the roots of $f(x)=x^5-3.5x^4+2.75x^3+2.125x^2-3.875x+1.25$
  - `x = roots([1 -3.5 2.75 2.125 -3.875 1.25])`

Polynomials (cont)

- MATLAB's `poly` function can be used to determine polynomial coefficients if roots are given:
  - `b = poly([0.5 -1])`
    - Finds $f(x)$ where $f(x) = 0$ for $x=0.5$ and $x=-1$
    - MATLAB reports `b = [1.000 0.5000 -0.5000]`
    - This corresponds to $f(x)=x^2+0.5x-0.5$
- MATLAB's `polyval` function can evaluate a polynomial at one or more points:
  - `a = [1 -3.5 2.75 2.125 -3.875 1.25];`
    - If used as coefficients of a polynomial, this corresponds to $f(x)=x^5-3.5x^4+2.75x^3+2.125x^2-3.875x+1.25$
  - `polyval(a, 1)`
    - This calculates $f(1)$, which MATLAB reports as -0.2500