Math 4543: Numerical Methods

Lecture 3 — Newton–Raphson Method

Syed Rifat Raiyan
Lecturer
Department of Computer Science & Engineering
Islamic University of Technology, Dhaka, Bangladesh
Email: rifatraiyan@iut-dhaka.edu

Lecture Plan

The agenda for today

- What is the Newton–Raphson method?
- A quick recap of some basic coordinate geometry
- The underlying principle of the method
- Derivation of the next approximation (using ***Taylor series***)
- The algorithm for the method
- Pros and cons
- Application as a substitute for costly mathematical operations

Newton–Raphson Method

What is it?

In the Newton–Raphson method, the root is **not bracketed**. In fact, **only one initial guess** of the root is needed to get the iterative process started to find the root of an equation. The method hence falls in the category of ***open methods***.

× Convergence in open methods is **not guaranteed**.

✓ But if the method does converge, it does so **much faster** than the bracketing methods.

Newton–Raphson Method

Recapitulating some background knowledge

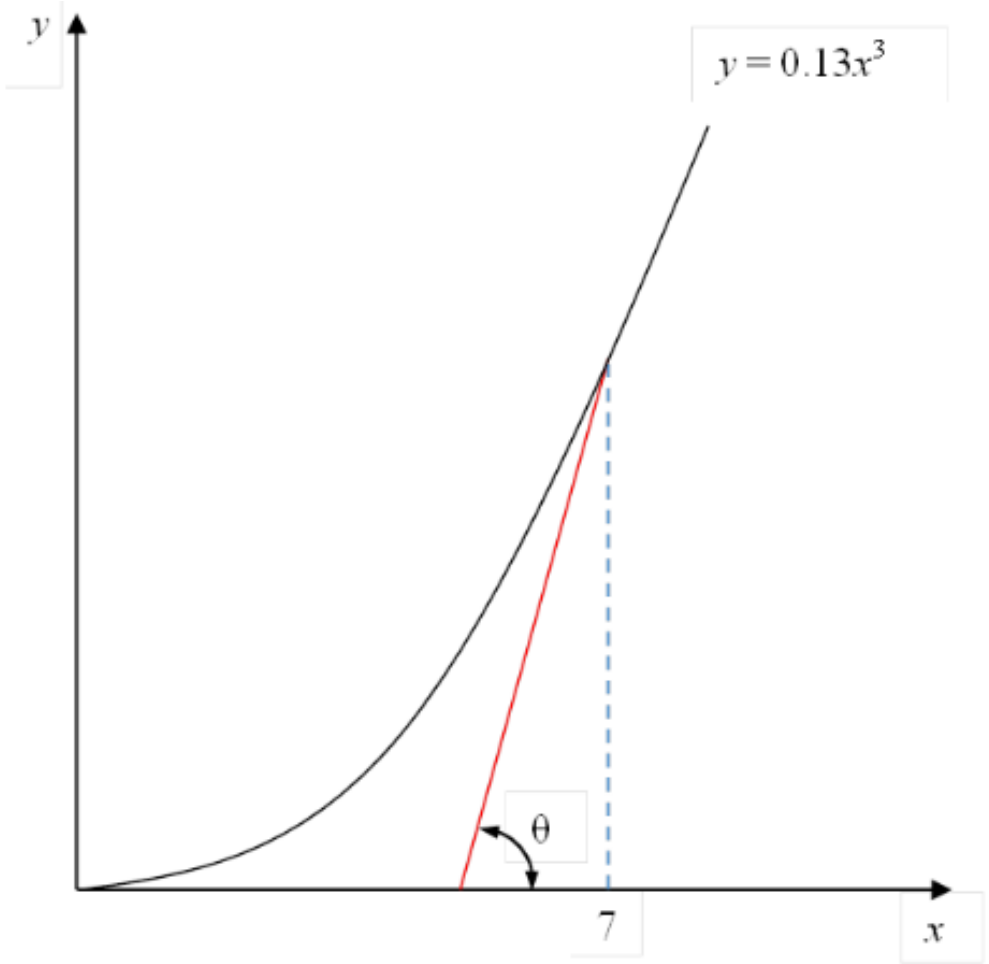


Figure 1: Tangent line crossing the x-axis

How to determine the intersection point between the red line and the X-axis?

Newton–Raphson Method

What is it based on?

The Newton-Raphson method is based on the principle that if the initial guess of the root of $f(x) = 0$ is at x_i, then if one draws the tangent to the curve at $(x_i, f(x_i))$ the point x_{i+1} where the tangent crosses the x -axis is an improved estimate of the root (Figure 1). One can then use x_{i+1} as the next point to draw the tangent line to the function $f(x)$ and find out where that tangent line crosses the x -axis. Continuing this process brings us closer and closer to the root of the equation.

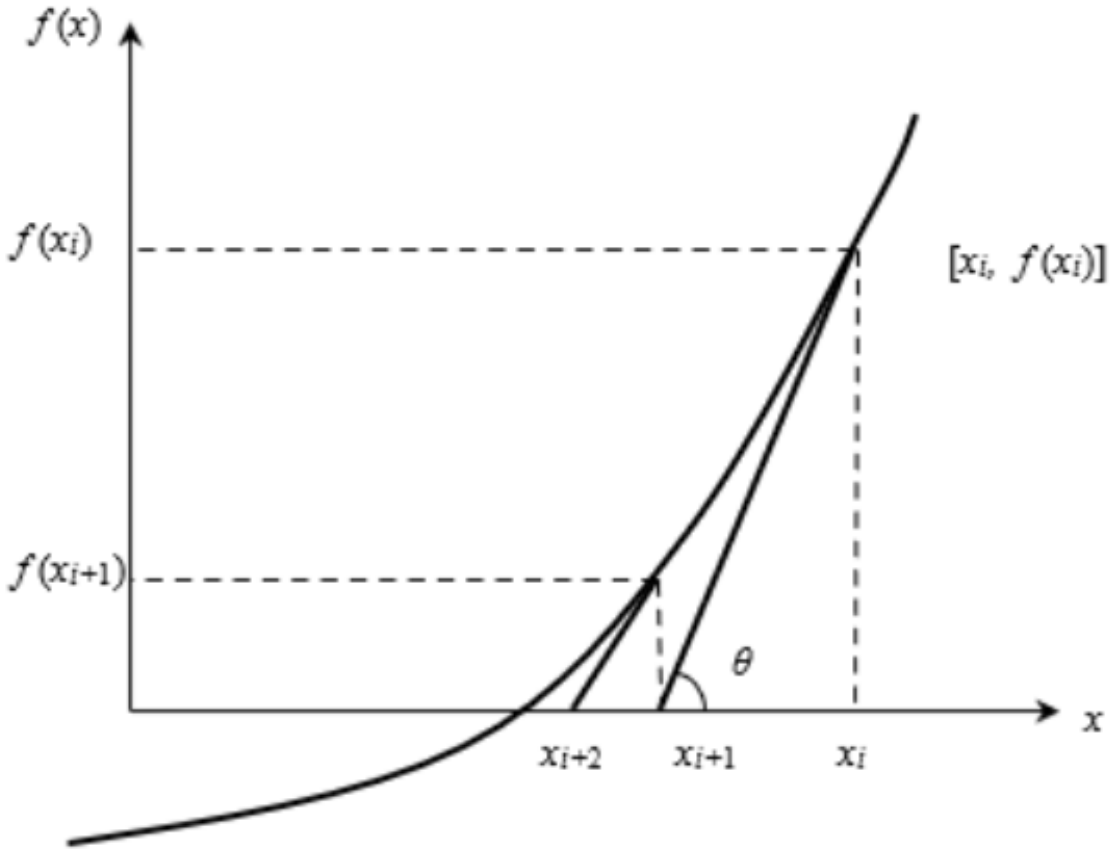


Figure 1 Geometrical illustration of the Newton-Raphson method.

Newton–Raphson Method

Deriving the next approximation

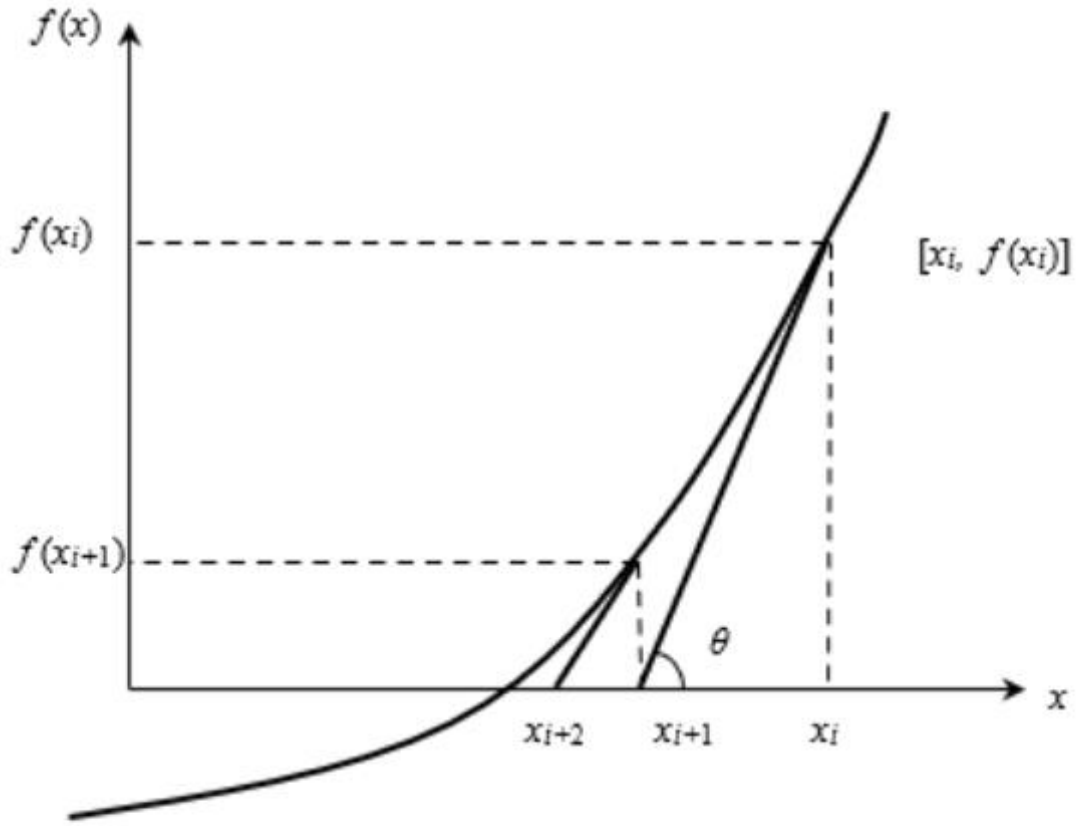


Figure 1 Geometrical illustration of the Newton-Raphson method.

Using the definition of the slope of a function, at $x = x_i$

$$\begin{aligned} f'(x_i) &= \tan\theta \\ &= \frac{f(x_i) - 0}{x_i - x_{i+1}}, \end{aligned}$$

which gives

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

Equation (1) is called the Newton-Raphson formula for solving nonlinear equations of the form $f(x) = 0$. So starting with an initial guess, x_i, one can find the next guess, x_{i+1} by using Equation (1). One can repeat this process until one finds the root within a desirable tolerance.

Newton–Raphson Method

Deriving the next approximation (using Taylor series)

$$f(x+h) = f(x) + f'(x)\,h + f''(x)\,\frac{h^2}{2!} + f'''(x)\,\frac{h^3}{3!} + \dots$$

The Newton-Raphson method can also be derived from the Taylor series. For a general function $f(x)$, the Taylor series around the point x_i is

$$f(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1} - x_i) + \frac{f''(x_i)}{2!}(x_{i+1} - x_i)^2 + \dots$$

We are seeking a point where $f(x) = 0$, that is if we assume

$$f(x_{i+1}) = 0,$$

$$0 = f(x_i) + f'(x_i)(x_{i+1} - x_i) + \frac{f''(x_i)}{2!}(x_{i+1} - x_i)^2 + \dots$$

Using only the first two terms of the series gives

$$0 \approx f(x_i) + f'(x_i)(x_{i+1} - x_i)$$

Solving for x_{i+1} gives

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

This is the same Newton-Raphson method formula as derived in a previous lesson using geometry and differential calculus.

Newton–Raphson Method

The algorithm

The steps of the Newton-Raphson method to find the root of an equation $f(x) = 0$ are

1. Evaluate $f'(x)$ symbolically
2. Use an initial guess of the root, x_i, to estimate the new value of the root, x_{i+1}, as

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

3. Find the absolute relative approximate error $|\epsilon_a|$ as

$$|\epsilon_a| = \left|\frac{x_{i+1} - x_i}{x_{i+1}}\right| \times 100 \qquad (2)$$

4. Compare the absolute relative approximate error with the pre-specified relative error tolerance, ϵ_s. If $|\epsilon_a| > \epsilon_s$, then go to Step 2, else stop the algorithm. 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.

Newton–Raphson Method

An example

Solve the nonlinear equation $x^3 = 20$ by the Newton-Raphson method using an initial guess of $x_0 = 3$.

Conduct three iterations to estimate the root of the above equation.

Find the absolute relative approximate error at the end of each iteration.

Find the number of significant digits at least correct at the end of each iteration.

Solution

First, we rewrite the equation $x^3 = 20$ in the form $f(x) = 0$

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

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

The initial guess is

$$x_0 = 3$$

Newton–Raphson Method

An example

Iteration 1

The estimate of the root is

$$\begin{aligned} x_1 &= x_0 - \frac{f(x_0)}{f'(x_0)} \\ &= 3 - \frac{3^3 - 20}{(3)^2} \\ &= 3 - \frac{7}{27} \\ &= 3 - 0.259259 \\ &= 2.74074 \end{aligned}$$

The absolute approximate relative error $|\epsilon_a|$ at the end of Iteration 1 is

$$\begin{aligned} |\epsilon_a| &= \left| \frac{x_1 - x_o}{x_1} \right| \times 100 \\ &= \left| \frac{2.74074 - 3}{2.74074} \right| \times 100 \\ &= 9.46\% \end{aligned}$$

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 digit to be correct in your result.

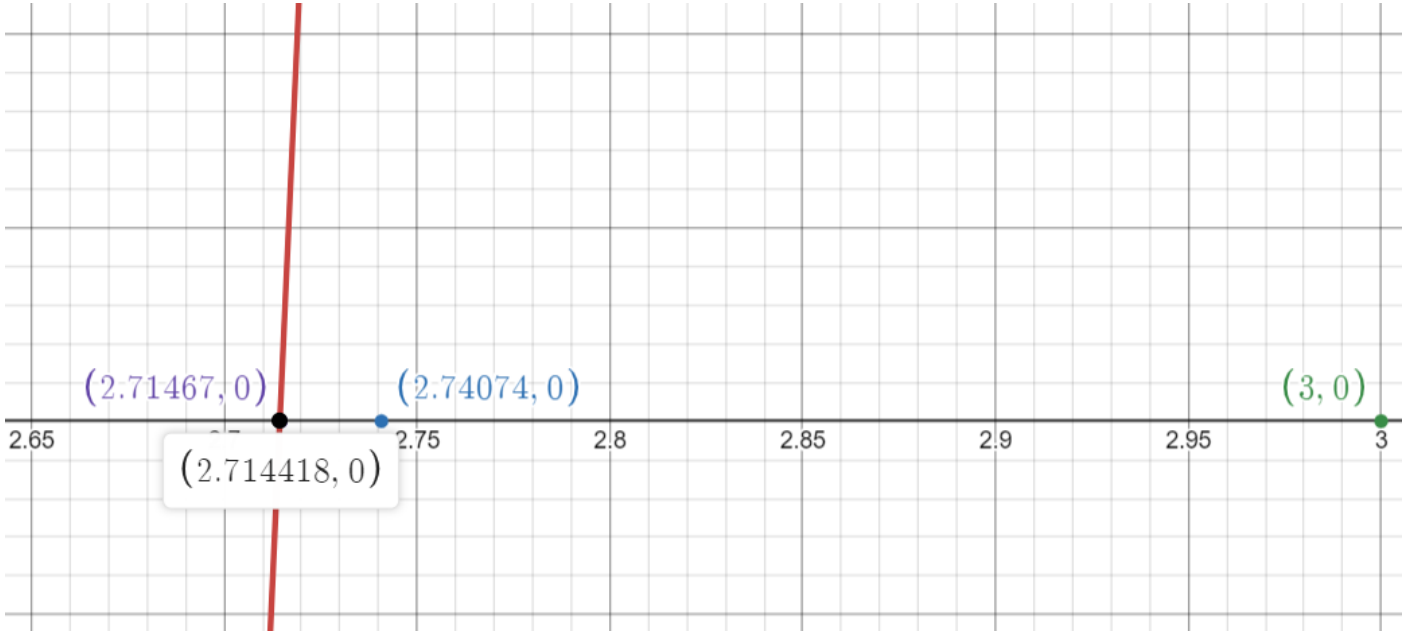
Newton–Raphson Method

An example

Iteration 2

The estimate of the root is

$$
\begin{aligned}
x_2 &= x_1 - \frac{f(x_1)}{f'(x_1)} \\
&= 2.74074 - \frac{(2.74074)^3 - 20}{3(2.74074)^2} \\
&= 2.74074 - \frac{0.587495}{22.5350} \\
&= 2.71467
\end{aligned}
$$



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

$$
\begin{aligned}
|\epsilon_a| &= \left| \frac{x_2 - x_1}{x_2} \right| \times 100 \\
&= \left| \frac{2.71467 - 2.74074}{2.71467} \right| \times 100 \\
&= 0.960\%
\end{aligned}
$$

The maximum value of m for which $|\epsilon_a| \leq 0.5 \times 10^{2-m}\%$ is 1.71669. Hence, the number of significant digits at least correct is 1.

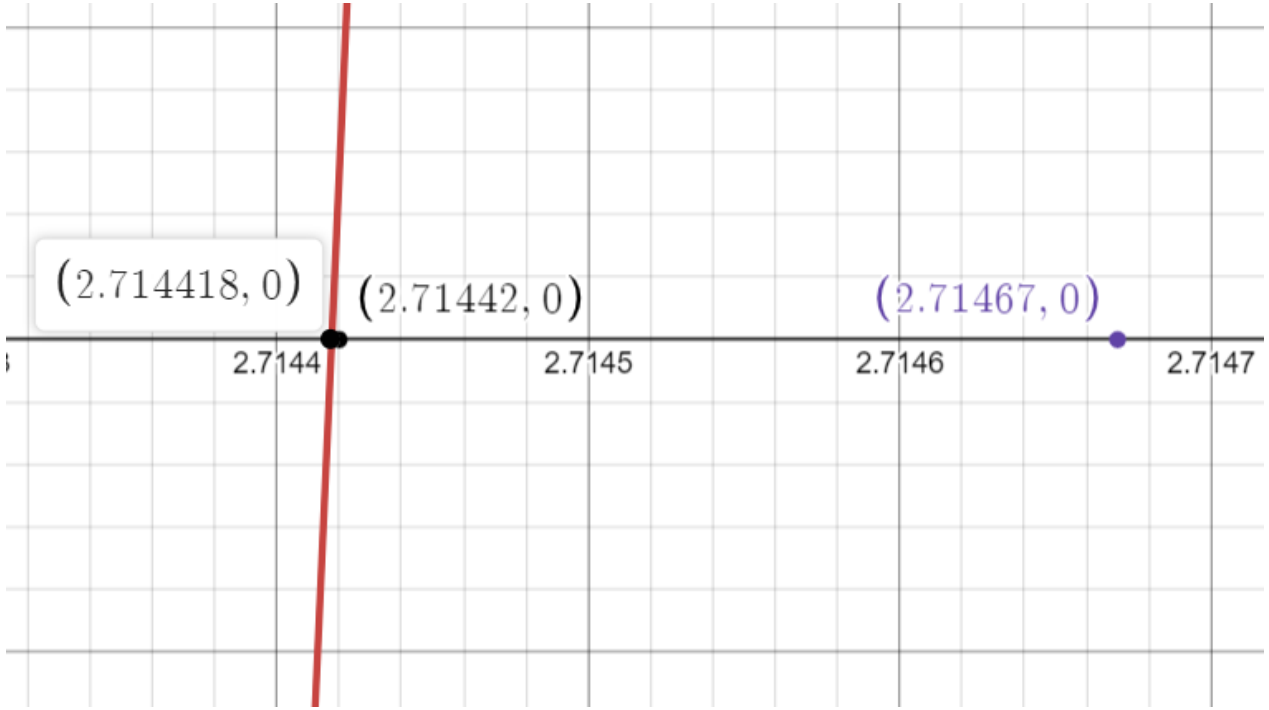
Newton–Raphson Method

An example

Iteration 3

The estimate of the root is

$$
\begin{aligned}
x_3 &= x_2 - \frac{f(x_2)}{f'(x_2)} \\
&= 2.71467 - \frac{(2.71467)^3 - 20}{(2.71476)^2} \\
&= 2.71467 - \frac{5.57925 \times 10^{-3}}{22.1083} \\
&= 2.71467 - 2.52360 \times 10^{-4} \\
&= 2.71442
\end{aligned}
$$



The absolute relative approximate error $|\epsilon_a|$ at the end of Iteration 3 is

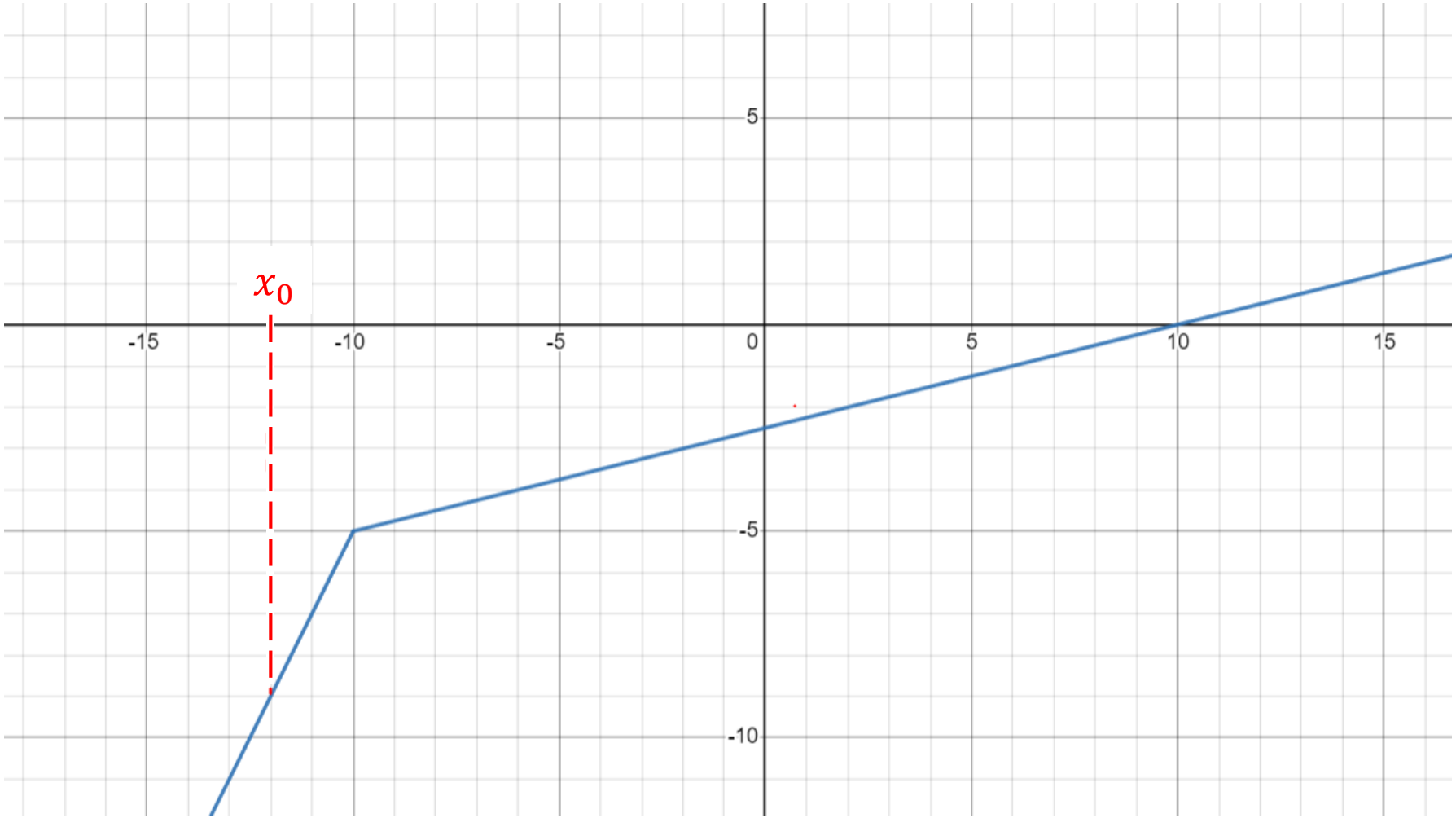
$$
\begin{aligned}
|\epsilon_a| &= \left| \frac{2.71442 - 2.71467}{2.71442} \right| \times 100 \\
&= 0.00921\%
\end{aligned}
$$

The maximum value of m for which $|\epsilon_a| \leq 0.5 \times 10^{2-m}\%$ is 3.73471. Hence, the number of significant digits at least correct is 3.

Mini Quiz

A Piecewise Linear graph

Think intuitively — How many **iterations** should Newton-Raphson take to find the root?



Newton-Raphson Method

Pros and cons

Advantages of the Newton-Raphson Method

1) Fast convergence: Newton-Raphson method converges fast when it converges. The method converges quadratically – one can interpret that the number of significant digits correct gets approximately squared at each iteration.

2) Only one initial estimate is needed: Newton-Raphson method is an open method that requires only one initial estimate of the root of the equation. However, since the method is an open method, convergence is not guaranteed.

Newton–Raphson Method

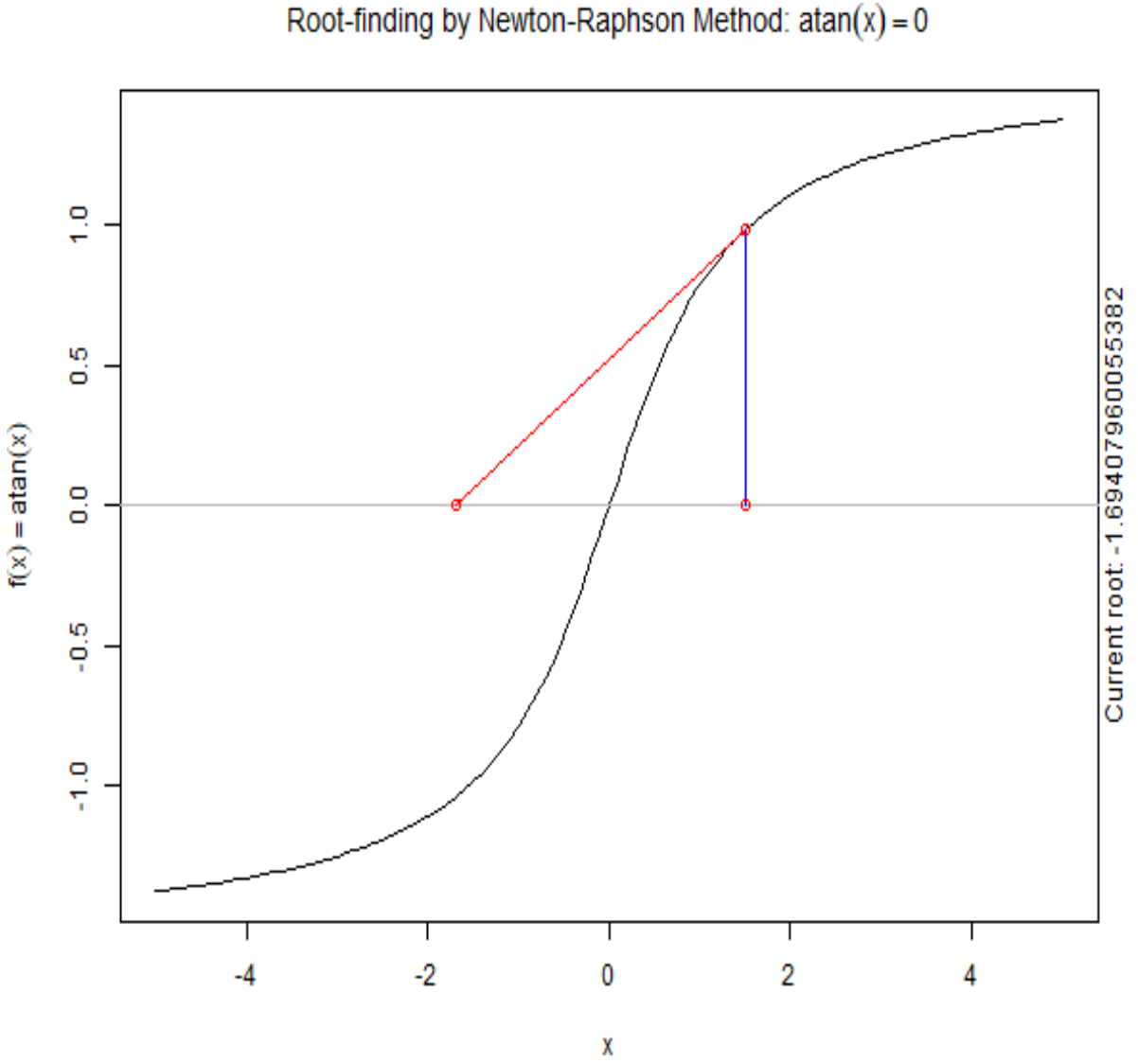
Pros and cons

Link: https://iq.opengenus.org/content/images/2019/09/nr.gif

Drawbacks of the Newton-Raphson Method

1) Divergence at inflection points: If the selection of the initial guess or an iterated value of the root turns out to be close to the inflection point of the function $f(x)$ in the equation $f(x) = 0$, Newton-Raphson method may start diverging away from the root. It may then begin to converging back to the root.

For a function $f(x)$, the point where the concavity changes from up-to-down or down-to-up is called its ***inflection point***, *i.e.*, the sign of curvature changes.



Newton–Raphson Method

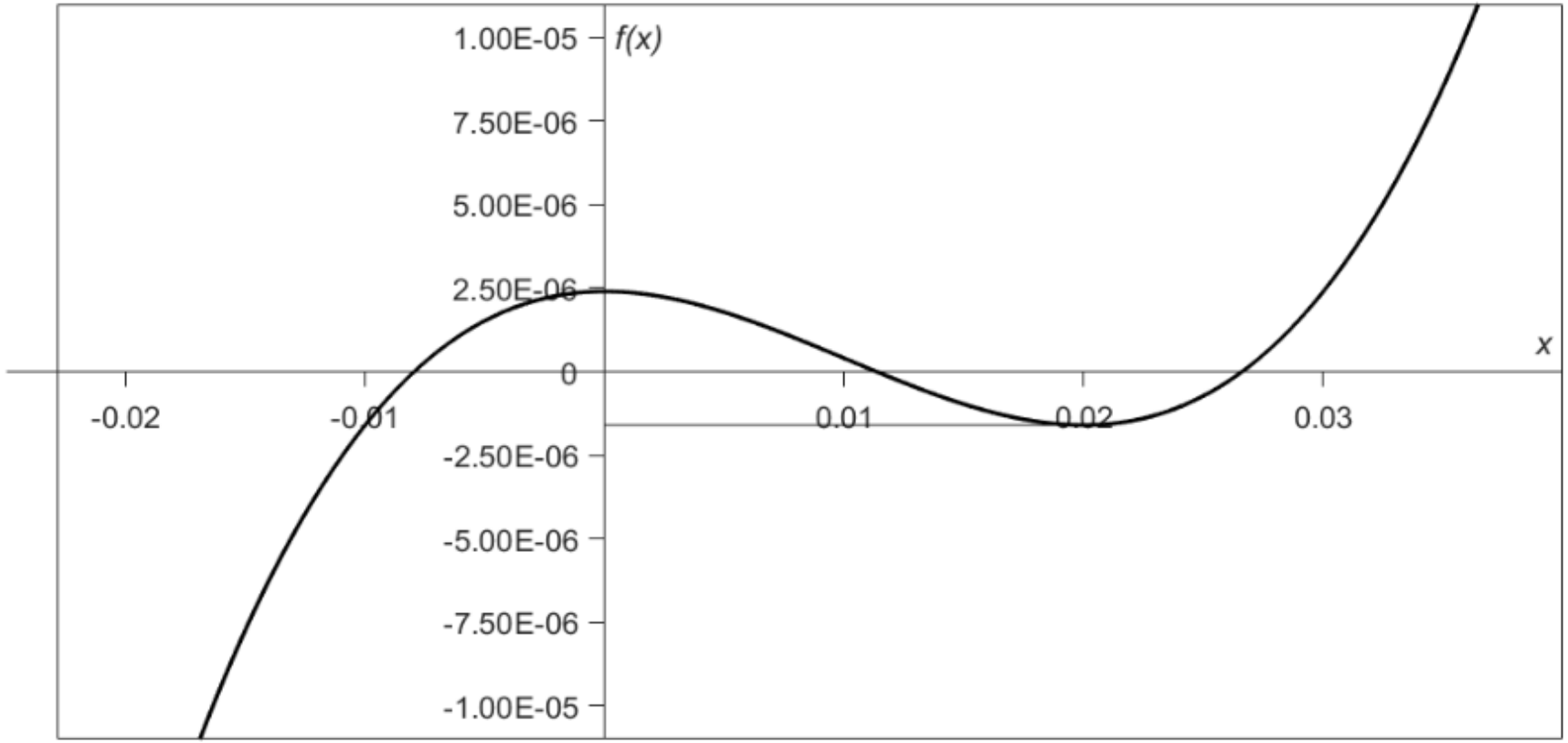
Pros and cons

2) Division by zero: Division by zero may occur during the implementation of the Newton-Raphson method. For example, for the equation

$$f(x) = x^3 - 0.03x^2 + 2.4 \times 10^{-6} = 0$$

then

$$f'(x) = 3x^2 - 0.06x$$

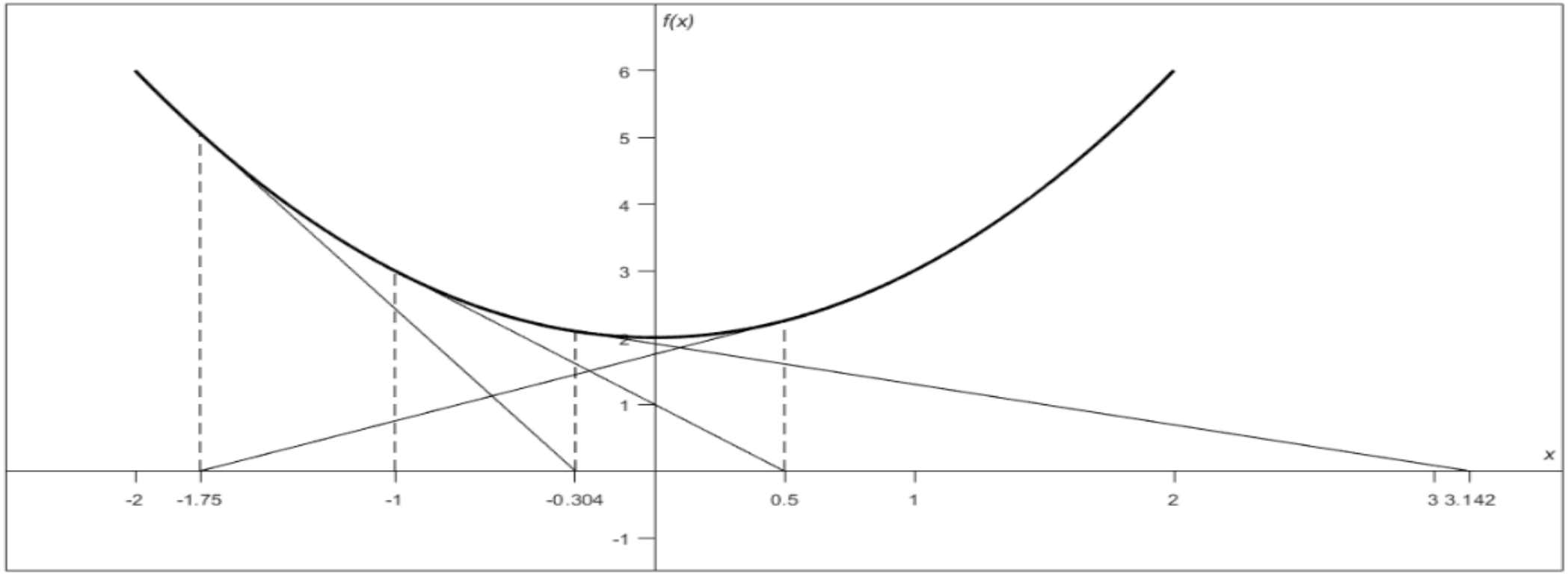


Newton–Raphson Method

Pros and cons

3) Oscillations near local maximum and minimum: 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.

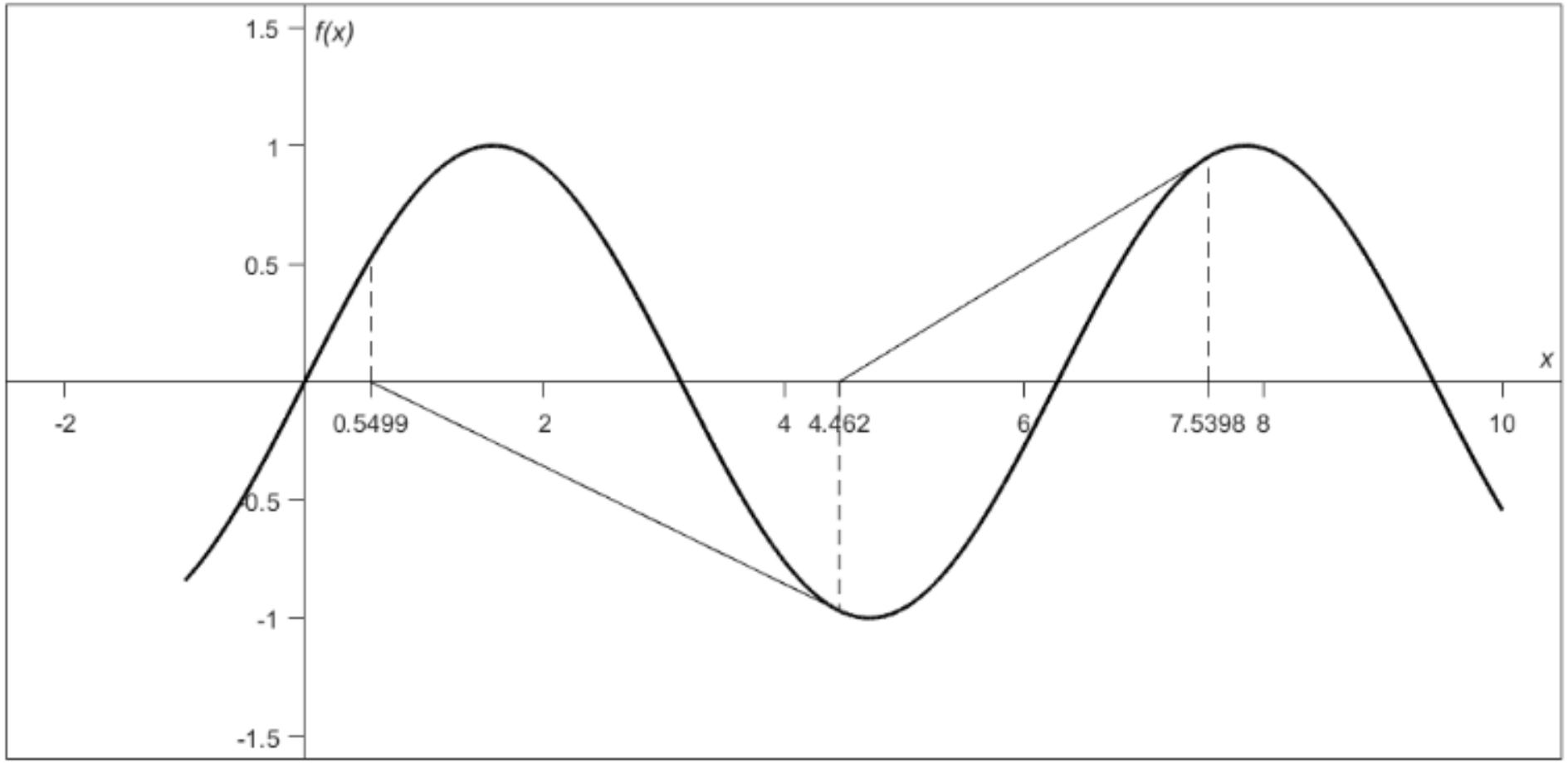
For example, $f(x) = x^2 + 2 = 0$ the equation has no real roots



Newton–Raphson Method

Pros and cons

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



Newton–Raphson Method

Application — Substitute for costly Mathematical Operations (*Division*)

If you want to find the value of b/a where b and a are real numbers, one can look at

$$\frac{b}{a} = b * \frac{1}{a} = b * c$$

where

$$c = \frac{1}{a}$$

$$f(c) = a - \frac{1}{c} = 0$$

gives

$$f'(c) = \frac{1}{c^2}$$

$$c_{i+1} = c_i - \frac{a - \frac{1}{c_i}}{\frac{1}{c_i^2}}$$

$$= c_i - c_i^2\left(a - \frac{1}{c_i}\right)$$

$$c_{i+1} = c_i(2 - c_i a)$$

This one is the acceptable iterative formula to find the inverse of a as it does not involve division.

- Clock cycles per iteration = **3 + 3 + 3 = 9**
- May take from **1** to **6** iterations to get a precise quotient.
- Using look-up table — **2** iterations [1].
- Can also be done using ***fused-multiply-subtract*** operations. (**`fmsub`**)

Typical functional unit speeds

Instruction	Latency	Issue rate
iadd/isub	1	1
and, or, etc.	1	1
shift, rotate	1	1
load/store	1-2	1-2
imul	3-15	3-15
fadd	3	1
fmul	3	1
fdiv	15-25	15-25
fsqrt	15-25	15-25

In general, most things take 1 or 2 clock cycles, except integer multiplication, and floating point division and square root.

Think: What can we do for calculating **square roots**?

References

Works cited in this presentation

[1] Wong, Derek C., and Michael J. Flynn. "Fast division using accurate quotient approximations to reduce the number of iterations."

Proceedings 10th IEEE Symposium on Computer Arithmetic. IEEE Computer Society, 1991.