EECE 231: Root Finding Algorithms

Department of Electrical and Computer Engineering

The Root Finding Problem

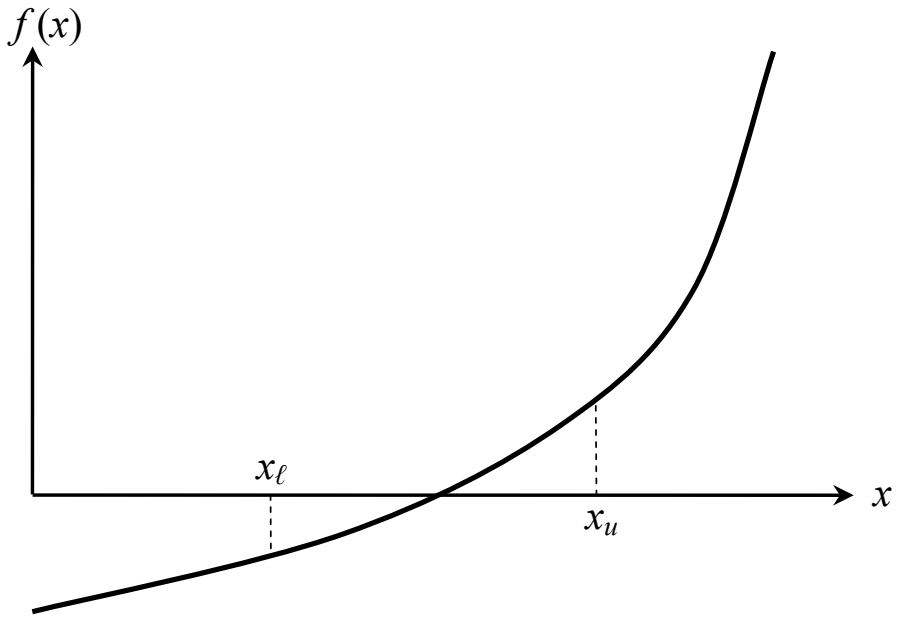
- Given a function $f(x)$, find c such that $f(c) = 0$.
- We will consider three algorithms for solving the root finding problem:
  - Bisection method
  - Newton method
  - Secant method

The Bisection Method

- One of the first numerical methods developed to find the root of a nonlinear equation $f(x) = 0$.

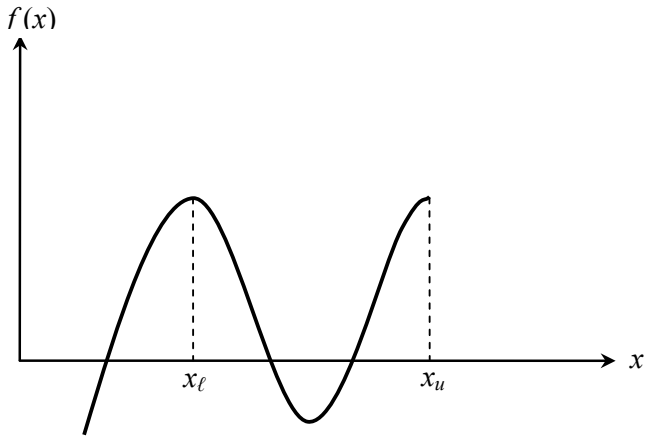
Theorem

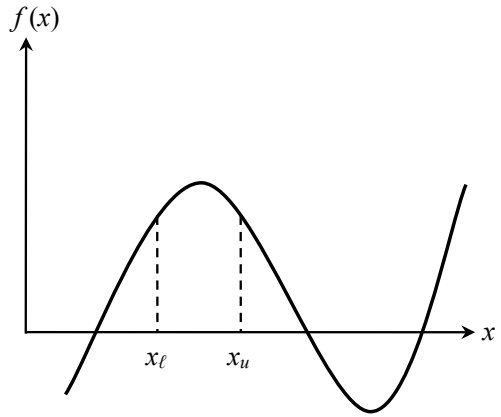
A real, continuous function $f(x)$ *has at least one root* $c \in [x_l, x_u]$ *where* $f(c) = 0$ *if* $f(x_l)f(x_u) < 0$.

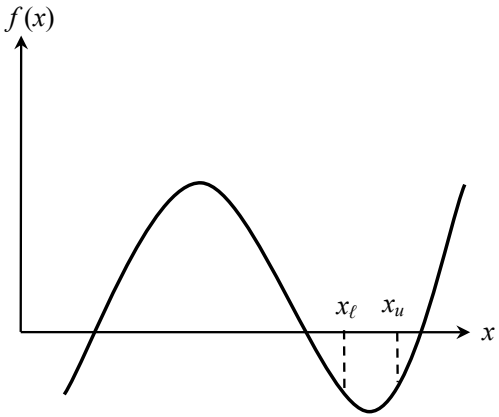


The Bisection Method (2)

- If $f(x_l)f(x_u) > 0$ a root may or may not exist between x_l and x_u.

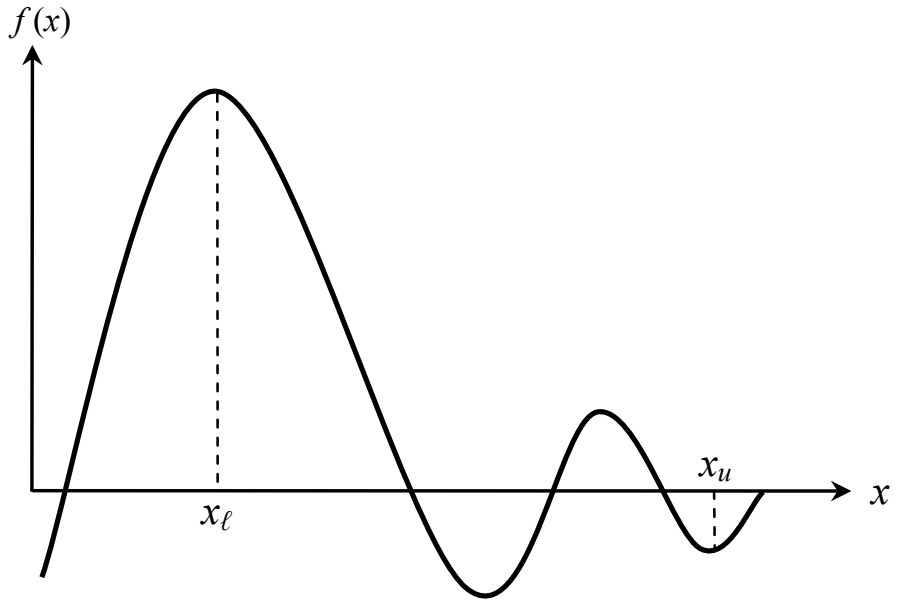






The Bisection Method (3)

- If $f(x_l)f(x_u) < 0$ there also may be more than one root between x_l and x_u.



The Bisection Method (4)

- The bisection method is similar to binary search. Because the root is in the interval $[x_l, x_u]$, we can try to guess its value.
- Let's consider the midpoint of the interval $[x_l, x_u]$, $x_m = \frac{x_l + x_u}{2}$.
- This splits $[x_l, x_u]$ into two halves: $[x_l, x_m]$ and $[x_m, x_u]$.
  - If $f(x_l)f(x_m) < 0$, the root is in $[x_l, x_m]$.
  - Otherwise, the root is in $[x_m, x_u]$
- We can now look for the root in the appropriate interval and continue to narrow intervals until we find the root, or reach a value that is very close to it.

Bisection Method Algorithm

1. Choose an interval $[x_l, x_u]$ such that $f(x_l)f(x_u) < 0$.
2. Estimate the root, x_m, as the mid-point between x_l and x_u:

$$x_m = \frac{x_l + x_u}{2}$$

3. Now check the following:
   a) If $f(x_l)f(x_m) < 0$ the root lies between x_l and x_m. Set $x_l = x_l$ and $x_u = x_m$.
   b) If $f(x_l)f(x_m) > 0$ the root lies between x_m and x_u. Set $x_l = x_m$ and $x_u = x_u$.
   c) If $f(x_l)f(x_m) = 0$ the root is x_m. Return x_m.

Bisection Method Algorithm (2)

4. Find the new root estimate, x_m^{new}:

$$x_m^{new} = \frac{x_l + x_u}{2}$$

5. Find the absolute relative approximate error:

$$|\epsilon_a| = \left| \frac{x_m^{new} - x_m^{old}}{x_m^{new}} \right| \times 100$$

   where x_m^{new} and x_m^{old} are the root estimates from the current (new) and previous iterations, respectively.

6. Compare $|\epsilon_a|$ to the pre-specified relative error tolerance ϵ_s. If $|\epsilon_a| > \epsilon_s$, go to Step 3. Otherwise stop, and return x_m^{new}.

Another Implementation of the Bisection Method Algorithm

Input: a function, two endpoints, a x-tolerance, and a y-tolerance
Output: a c such that $|f(c)|$ is smaller than the y-tolerance.

RUN_BISECTION$(f, a, b, \delta, \epsilon)$

(1) Let $fa \leftarrow \text{sign}\,(f(a))$.
(2) Let $fb \leftarrow \text{sign}\,(f(b))$.
(3) **if** $fafb > 0$
(4) throw an error.
(5) **else if** $fafb = 0$
(6) **if** $fa = 0$
(7) **return** a
(8) **else**
(9) **return** b
(10) Let $c \leftarrow \frac{a+b}{2}$
(11) **while** $b - a > 2\delta$
(12) Let $fc \leftarrow f(c)$.
(13) **if** $|fc| < \epsilon$
(14) **return** c
(15) **if** $\text{sign}\,(fa)\,\text{sign}\,(fc) < 0$
(16) Let $b \leftarrow c$, $fb \leftarrow fc$, $c \leftarrow \frac{a+c}{2}$.
(17) **else**
(18) Let $a \leftarrow c$, $fa \leftarrow fc$, $c \leftarrow \frac{c+b}{2}$.
(19) **return** c

Advantages of the Bisection Method

- The bisection method will always converge to a solution.
- Because the interval is constantly being halved, a bound on the error can be guaranteed.

Disadvantages of the Bisection Method

- The convergence rate of the bisection method is slow.
- If one of the initial guesses is close to the root, it will take longer to reach the root.
- If a function $f(x)$ touches the x-axis (e.g. $f(x) = x^2 = 0$) the bisection method will not be able to find lower and upper guesses x_l and x_u such that $f(x_l)f(x_u) < 0$.
- For functions $f(x)$ where there is a singularity and it reverses signs at the singularity (e.g. $f(x) = \frac{1}{x}$, the bisection method may converge on the singularity.

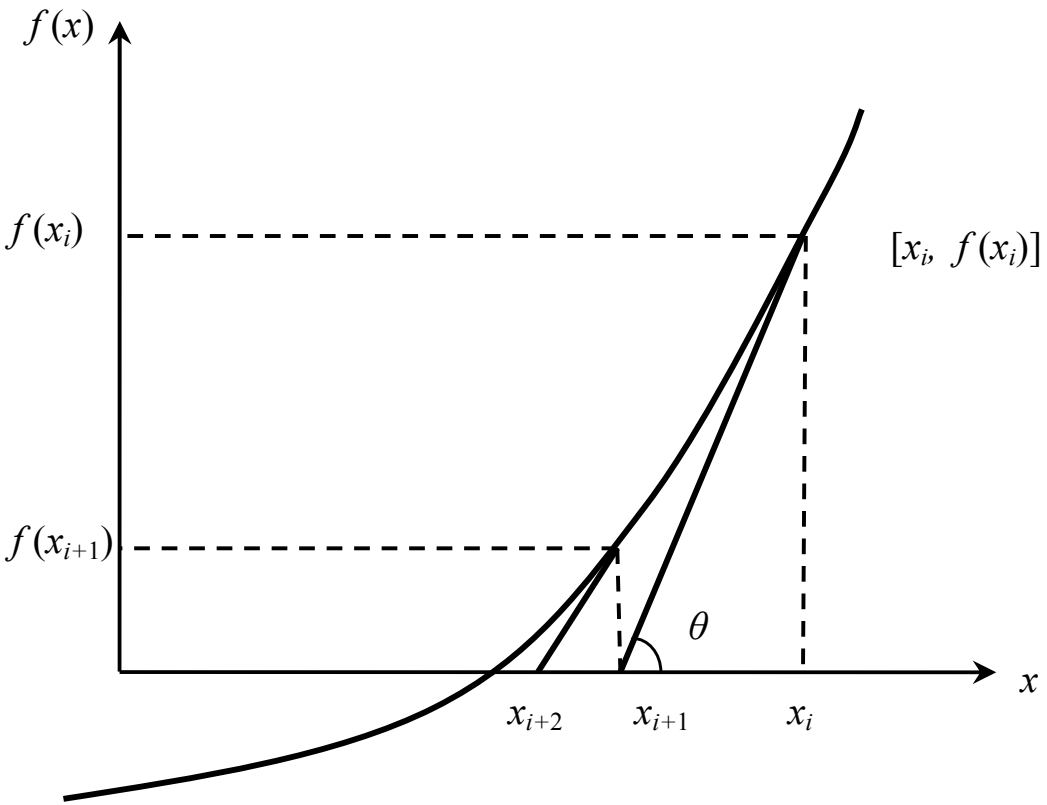
The Newton-Raphson Method

- The bisection method belongs to a class of solutions that use **bracketing** (i.e. using two guesses, an upper and a lower bound) to find the solution.
  - These methods always converge because they zero in on a solution by repeatedly reducing the interval to which the solution belongs.

- By contrast, the Newton-Raphson method is an **open method** that uses a single initial guess to search for the solution.
  - A solution is not guaranteed to be found, but if it exists it can be computed much faster than bracketing methods.

Geometric Derivation

- Starting with an initial guess of the root, x_i, extending the tangent to the curve at $f(x_i)$ to the point x_{i+1} that intersects the x-axis provides a better estimate of the root.



Geometric Derivation (2)

- The slope of a function at $x = x_i$ is:

$$f'(x_i) = \tan(\theta) = \frac{\Delta y}{\Delta x} = \frac{f(x_i) - 0}{x_i - x_{i+1}}$$

- This produces the Newton-Raphson formula:

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

- Applying the Newton-Raphon formula iteratively enables finding the root of a non-linear function $f(x) = 0$ to a given **tolerance** because, depending on the computer's numerical precision (bits used to represent numbers), it may not be possible to find an exact root.

Taylor Series Derivation

- Points on a differentiable function $f(x)$ can be estimated using a Taylor series expansion:

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

- We can estimate the function using the first two terms of the series:

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

- Because we are seeking $f(x) = 0$ we can assume that:

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

- This gives us the Newton-Raphson formula:

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

Newton-Raphson Algorithm

1. Evaluate $f'(x)$ symbolically.
2. Use an initial estimate of x_i and use it to estimate the new value of the root:
$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$
3. Find the absolute relative approximate error $|\epsilon_a|$:
$$|\epsilon_a| = \left| \frac{x_{i+1} - x_i}{x_{i+1}} \right| \times 100$$
4. Compare the absolute relative approximate error with a pre-specified relative error tolerance ϵ_s.
   - If $|\epsilon_a| > \epsilon_s$ go to Step 2.
   - Or, if the number of iterations is less than a maximum limit go to Step 2.
   - Otherwise end algorithm.

Newton-Raphson Algorithm (2)

Input: a function, its derivative, an initial guess, an iteration limit, and a tolerance
Output: a point for which the function has small value.

RUN_NEWTON$(f, f', x0, N, tol)$

(1) Let $x \leftarrow x0, n \leftarrow 0$.
(2) **while** $n \leq N$
(3) Let $fx \leftarrow f(x)$.
(4) **if** $|fx| < tol$
(5) **return** x.
(6) Let $fpx \leftarrow f'(x)$.
(7) **if** $|fpx| < tol$
(8) Warn "$f'(x)$ is small; giving up."
(9) **return** x.
(10) Let $x \leftarrow x - fx/fpx$.
(11) Let $n \leftarrow n + 1$.

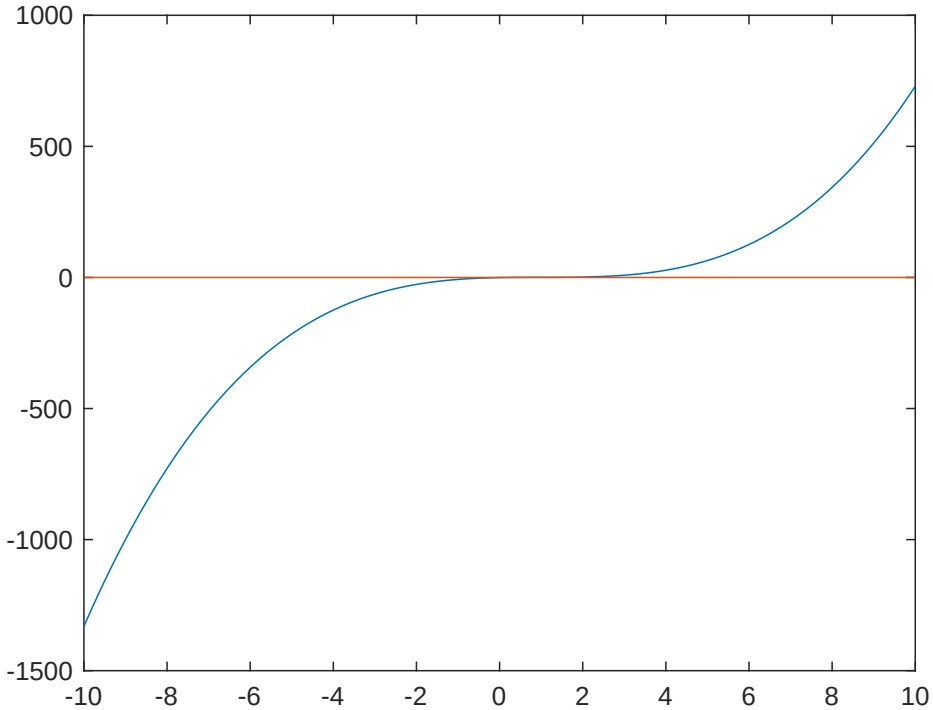
Problems with the Newton-Raphson Method

- The Newton-Raphson method suffers from a number of problems:
  - Divergence at inflection points
  - Division by zero
  - Oscillations near local maxima/minima
  - Root jumping

Divergence at inflection points

- An inflection point in a function $f(x)$ is a point where the concavity of the function changes (e.g. downwards to upwards, or vice versa). In other words, it is the point where $f''(x) = 0$.

$$f(x) = (x-1)^3 + 0.512$$



- In this function the inflection point occurs when $x = 1$.

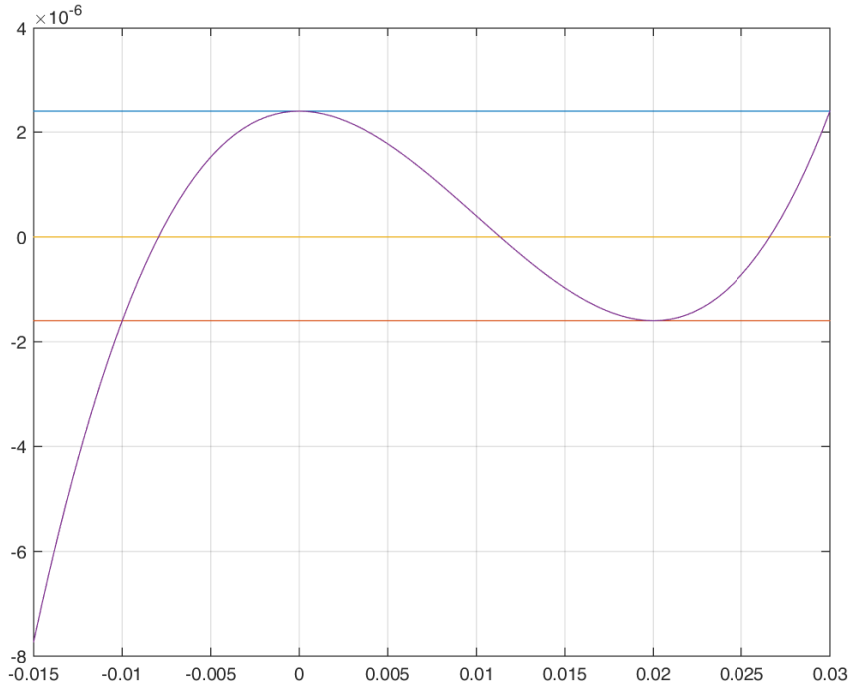
Divergence at inflection points (2)

- As the estimated root gets closer to the inflection point ($x = 1$), the Newton-Raphson algorithm may produce diverging results. However, it may recover and produce the correct result.

Iteration	x	f(x)	f'(x)	x-f(x)/f'(x)
0	5.000000	64.512000	48.000000	3.656000
1	3.656000	19.248316	21.163008	2.746474
2	2.746474	5.839041	9.150509	2.108363
3	2.108363	1.873587	3.685402	1.599982
4	1.599982	0.727980	1.079935	0.925885
5	0.925885	0.511593	0.016479	-30.119181
6	-30.119181	-30135.410443	2905.210339	-19.746297
7	-19.746297	-8928.877763	1291.226532	-12.831261
8	-12.831261	-2645.460677	573.911365	-8.221733
9	-8.221733	-783.707482	255.121077	-5.149829
10	-5.149829	-232.076958	113.461185	-3.104398
11	-3.104398	-68.631053	50.538260	-1.746397
12	-1.746397	-20.203230	22.628083	-0.853558
13	-0.853558	-5.856225	10.307030	-0.285380
14	-0.285380	-1.611707	4.956606	0.039783
15	0.039783	-0.373335	2.766047	0.174754
16	0.174754	-0.050018	2.043093	0.199236
17	0.199236	-0.001469	1.923671	0.199999
18	0.199999	-0.000001	1.920004	0.200000
19	0.200000	0.000000	1.920000	0.200000

Division by zero

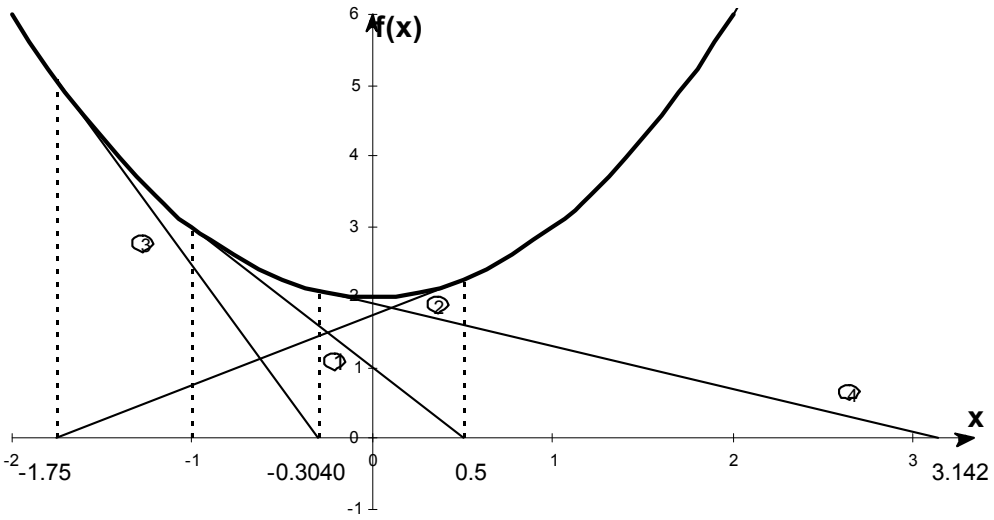
$$f(x) = x^3 - 0.03x^2 + 2.4 \times 10^{-6}$$



- For this function $f'(x) = 3x^2 - 0.06x = 0$ at $x = 0$ and $x = 0.02$.
- Estimating roots at or near these points can result in division by zero or very large estimates where Newton-Raphson does not converge.

Oscillations near local maxima/minima

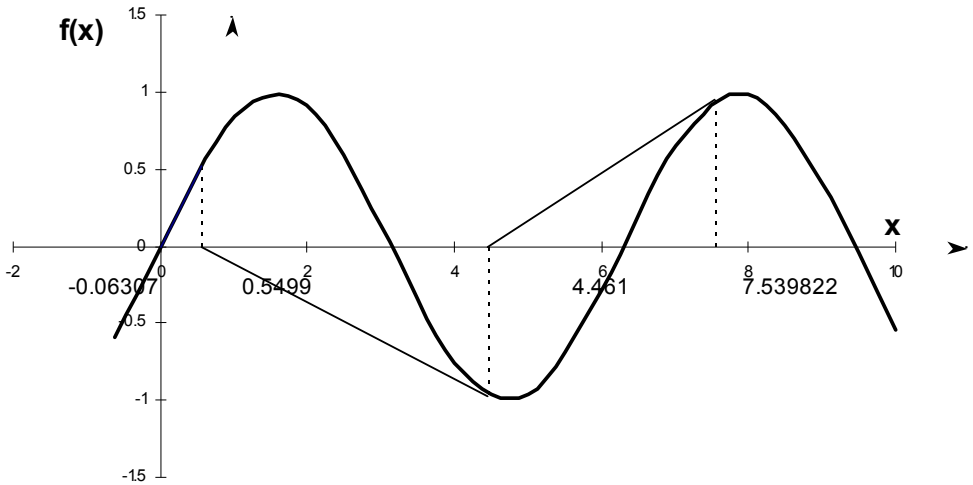
- The Newton-Raphson method may produce results that converge upon and oscillate around local maxima/minima, and that may diverge away from the root.
- The function $f(x) = x^2 + 2$ does not have (real) roots. Applying the Newtop Raphson method results in oscillations around the (global) minimum at $x = 0$.



Iteration	x	f(x)	f'(x)	x-f(x)/f'(x)
0	-1.0000	3.0000	-2.0000	0.5000
1	0.5000	2.2500	1.0000	-1.7500
2	-1.7500	5.0625	-3.5000	-0.3036
3	-0.3036	2.0922	-0.6071	3.1423
4	3.1423	11.8742	6.2847	1.2529
5	1.2529	3.5698	2.5059	-0.1717
6	-0.1717	2.0295	-0.3433	5.7395
7	5.7395	34.9422	11.4791	2.6955
8	2.6955	9.2659	5.3911	0.9768
9	0.9768	2.9541	1.9536	-0.5354
10	-0.5354	2.2866	-1.0708	1.6002

Root jumping

- For functions with multiple roots the Newton-Raphson method may return a root that is farther away from an initial estimate than a closer root.
- For example, if we use an initial root estimate of 2.4π (7.5398) for the function $f(x) = sin(x)$, the Newton-Raphson algorithm will return the root at $x = 0$ instead of the closer root at $x = 2\pi$ (6.2832).



Iteration	x	f(x)	f'(x)	x-f(x)/f'(x)
0	7.539822369	0.951057	0.309017	4.462139
1	4.462138831	-0.96885	-0.24765	0.549904
2	0.549904286	0.522606	0.852575	-0.06307
3	-0.063069242	-0.06303	0.998012	8.38E-05
4	8.37574E-05	8.38E-05	1	-2E-13
5	-1.95861E-13	-2E-13	1	0

Secant Method

- One of the disadvantages of the Newton-Raphson method is that it requires knowledge of the derivative function $f'(x)$. Although symbolic manipulation programs like MATLAB make this possible, there may be times when calculating the derivative function is not possible or practical.

- The derivative function can be approximated by the slope of the secant line passing through the points $(x_{i-1}, f(x_{i-1}))$ and $(x_i, f(x_i))$:

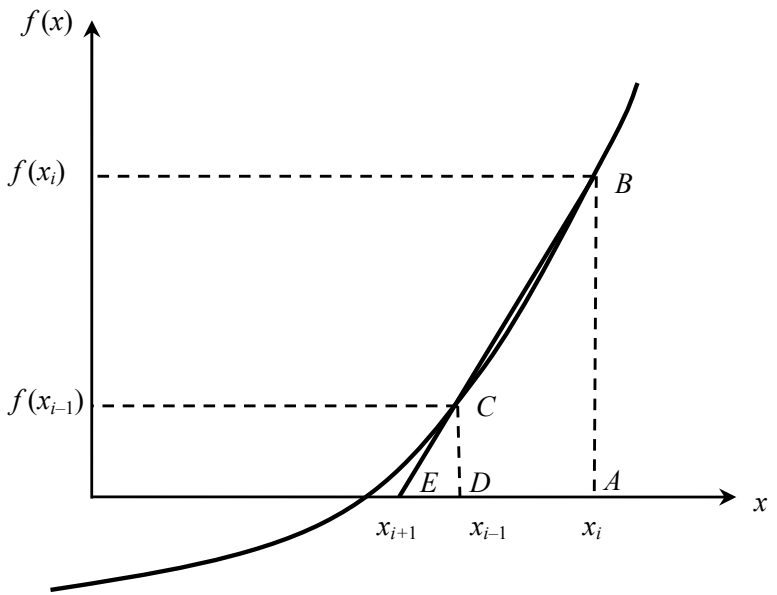
$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

Secant Method (2)

- Substituting the approximation of $f'(x_i)$ in the Newton-Raphson equation:

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

- In other words, the Secant Method estimates the root as the x-intercept of the secant line connecting two points on the function graph.



Performance of the Secant Method

- Like the bisection method, the secant method needs two initial points to estimate the root of a function. But unlike the bisection method, the two values are not used to bracket the root.
- The secant method is an open method that is not guaranteed to converge on a root.
- When it does converge, the secant method is faster than the bisection method, but slower than the Newton-Raphson method.