

UNIT 1

Solution of Nonlinear Equations

(10 Hours) — Numerical Methods (CAC252)

1.1 Introduction

Numerical methods are techniques that turn a mathematical problem into a sequence of arithmetic operations (addition, subtraction, multiplication and division) that a computer can carry out. Many equations met in science and engineering cannot be solved by a closed formula, for example $x^3 - 2x - 5 = 0$ or $x = \cos x$. For such problems we look for an **approximate** solution whose error we can estimate and control.

A **root** (or zero) of an equation $f(x) = 0$ is a number α such that $f(\alpha) = 0$. Geometrically it is a point where the graph of $y = f(x)$ crosses the x -axis. Almost every method in this unit is **iterative**: it starts from one or two initial guesses and produces a sequence $x_0, x_1, x_2, \dots$ that (hopefully) converges to α .

1.1.1 Types of Equation

Type	Description	Example
Algebraic (polynomial)	$f(x)$ is a polynomial in x with a finite number of terms	$x^3 - 2x - 5 = 0$
Transcendental	$f(x)$ contains trigonometric, exponential, logarithmic terms	$x - \cos x = 0, e^x - 3x = 0$
Linear	Highest power of the unknown is 1	$3x - 7 = 0$
Nonlinear	Unknown appears with power other than 1, or inside a function	$x^2 - 5 = 0, \sin x = x/2$
System of nonlinear equations	Two or more simultaneous nonlinear equations in several unknowns	$x^2 + y^2 = 4, x^2 - y^2 = 1$

Locating a root. Before iterating we need an initial interval or guess. The **Intermediate Value Theorem** states: if f is continuous on $[a, b]$ and $f(a) \cdot f(b) < 0$, then f has at least one root in (a, b) . A simple table of values of $f(x)$ at $x = 0, 1, 2, \dots$ is usually enough to find a sign change.

Note: For a polynomial of degree n there are exactly n roots (real or complex). Descartes' rule says the number of positive real roots is at most the number of sign changes in the coefficients of $f(x)$.

1.2 Errors in Computing

No numerical answer is exact. Understanding where errors come from tells us how much to trust a result.

1.2.1 Sources of Error

- **Inherent (input) error** — error already present in the data, e.g. from measurement.
- **Round-off error** — caused by representing numbers with a finite number of digits (e.g. $1/3$ stored as 0.3333). It accumulates over many operations.
- **Truncation error** — caused by replacing an infinite process by a finite one, e.g. cutting a Taylor series after a few terms or stopping an iteration early.
- **Blunder / human error** — wrong formula, mistyped data or programming mistake.

1.2.2 Measures of Error

Let x be the exact (true) value and x^* the approximate value.

Measure	Formula	Meaning
Absolute error	$E_a = x - x^* $	Size of the error in the units of x
Relative error	$E_r = x - x^* / x $	Error compared with the size of the value
Percentage error	$E_p = E_r \times 100 \%$	Relative error expressed as a percent
Error bound	$ x - x^* \leq \Delta x$	A guaranteed upper limit for the absolute error

Significant digits. The digits of a number that carry meaning, starting from the first non-zero digit. 0.003050 has four significant digits (3, 0, 5, 0). A number is correct to n significant digits if its relative error is at most 5×10^{-n} .

Example 1.1

Find the absolute, relative and percentage errors if $\pi = 3.14159265$ is approximated by $22/7$.

Solution

$$22/7 = 3.14285714$$

$$\text{Absolute error } E_a = |3.14159265 - 3.14285714| = 0.00126449$$

$$\text{Relative error } E_r = E_a / \pi = 0.00126449 / 3.14159265 = 0.000402 (\approx 4.025 \times 10^{-4})$$

$$\text{Percentage error } E_p = 0.04025 \%$$

1.2.3 Propagation of Errors

If x and y have absolute errors Δx and Δy , then in the worst case:

- Sum / difference: $\Delta(x \pm y) \leq \Delta x + \Delta y$
- Product: $\Delta(xy)/|xy| \approx \Delta x/|x| + \Delta y/|y|$ (relative errors add)
- Quotient: $\Delta(x/y)/|x/y| \approx \Delta x/|x| + \Delta y/|y|$
- General function: $\Delta f \approx |\partial f/\partial x| \cdot \Delta x + |\partial f/\partial y| \cdot \Delta y$

Note: Subtracting two nearly equal numbers loses significant digits (loss of significance). For example, $\sqrt{10001} - 100 \approx 0.00499988$ computed with 6-digit arithmetic gives 0.0050, i.e. only 2 correct digits. Rewrite as $1/(\sqrt{10001} + 100)$ to avoid the cancellation.

1.3 The Bisection Method

The bisection method (interval halving) is the simplest bracketing method. It relies on the Intermediate Value Theorem: if $f(a)$ and $f(b)$ have opposite signs, a root lies between a and b . We repeatedly halve the interval and keep the half in which the sign change persists.

Algorithm

- Choose a and b with $f(a) \cdot f(b) < 0$ and a tolerance ϵ .
- Compute the midpoint $c = (a + b)/2$.
- If $f(c) = 0$ or $(b - a)/2 < \epsilon$, stop: c is the root.
- If $f(a) \cdot f(c) < 0$ set $b = c$, otherwise set $a = c$. Go to step 2.

Error bound and iteration count. After n steps the interval has length $(b - a)/2^n$, so the midpoint is within $(b - a)/2^{n+1}$ of the root. To achieve accuracy ϵ we need

$$n \geq \lceil \log(b - a) - \log \epsilon \rceil / \log 2$$

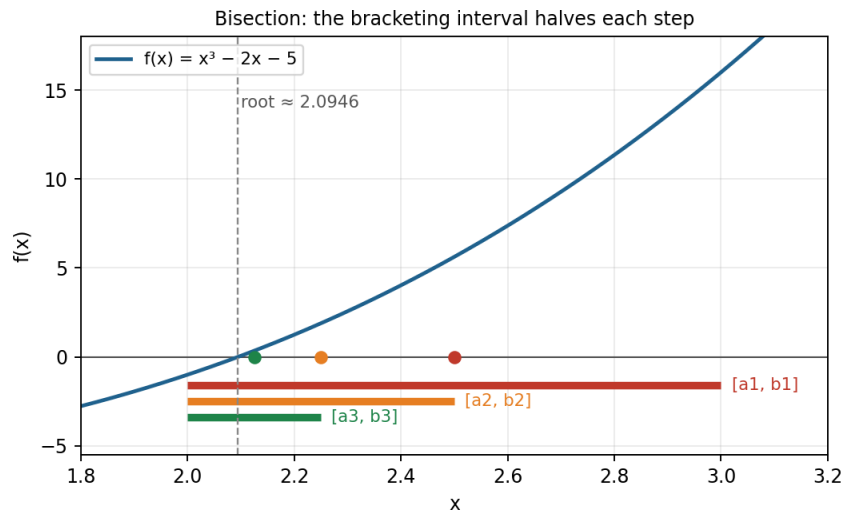


Figure 1.1 — Bisection on $f(x) = x^3 - 2x - 5$: the bracket $[a, b]$ is halved at every step.

Example 1.2

Find a root of $f(x) = x^3 - 2x - 5 = 0$ in $[2, 3]$ by the bisection method (10 iterations). How many iterations guarantee an error below 0.001?

Solution

$f(2) = 8 - 4 - 5 = -1 < 0$ and $f(3) = 27 - 6 - 5 = 16 > 0$, so a root lies in $(2, 3)$.

Iterations needed: $n \geq \log_2(1/0.001) = 9.966$, so $n = 10$.

n	a	b	$c = (a+b)/2$	$f(c)$	Sign
1	2.0000	3.0000	2.5000	5.6250	+
2	2.0000	2.5000	2.2500	1.8906	+
3	2.0000	2.2500	2.1250	0.3457	+
4	2.0000	2.1250	2.0625	-0.3513	-
5	2.0625	2.1250	2.0938	-0.0089	-
6	2.0938	2.1250	2.1094	0.1668	+
7	2.0938	2.1094	2.1016	0.0786	+
8	2.0938	2.1016	2.0977	0.0347	+
9	2.0938	2.0977	2.0957	0.0129	+
10	2.0938	2.0957	2.0947	0.0020	+

After 10 iterations the midpoint is $c = 2.0947$. The true root is 2.0945515, so the root is 2.09 (to 3 s.f.).

Note: Bisection always converges (if f is continuous and the bracket is valid), but slowly — it gains about one binary digit (0.3 decimal digits) per iteration. Its order of convergence is 1 (linear).

1.4 The Method of False Position (Regula Falsi)

False position also keeps a bracket $[a, b]$, but instead of the midpoint it uses the point where the **chord** joining $(a, f(a))$ and $(b, f(b))$ crosses the x-axis. This uses the function values and usually converges faster than bisection.

Equation of the chord and its x-intercept:

$$c = [a \cdot f(b) - b \cdot f(a)] / [f(b) - f(a)] = a - f(a) \cdot (b - a) / [f(b) - f(a)]$$

Algorithm

- Choose a, b with $f(a) \cdot f(b) < 0$.
- Compute $c = [a \cdot f(b) - b \cdot f(a)] / [f(b) - f(a)]$.
- If $|f(c)| < \epsilon$ (or successive c values agree to the required accuracy), stop.
- If $f(a) \cdot f(c) < 0$ set $b = c$, else set $a = c$. Repeat from step 2.

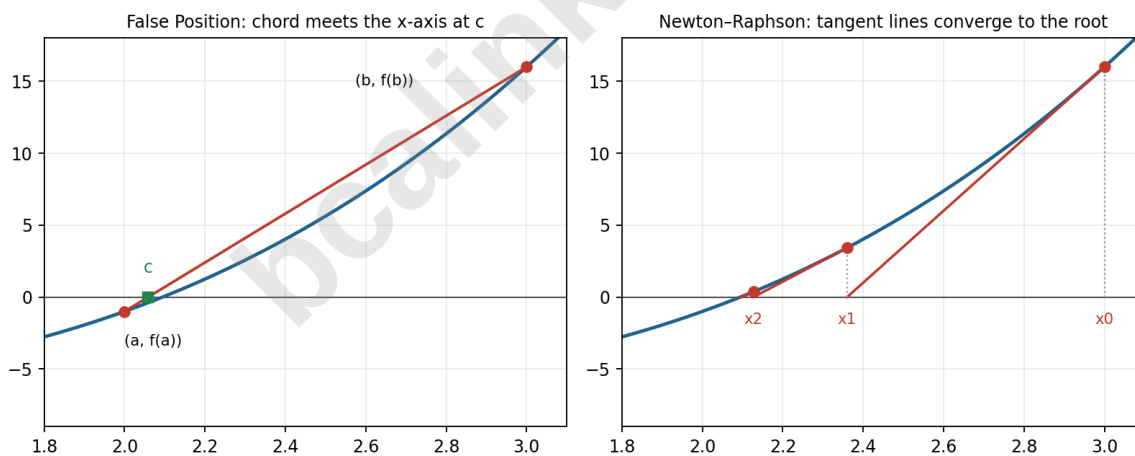


Figure 1.2 — Left: false position (chord). Right: Newton–Raphson (tangents) starting at $x_0 = 3$.

Example 1.3

Solve $x^3 - 2x - 5 = 0$ in $[2, 3]$ by the method of false position (6 iterations).

Solution

Using $c = [a \cdot f(b) - b \cdot f(a)] / [f(b) - f(a)]$, with $f(2) = -1$, $f(3) = 16$:

$$c_1 = (2 \cdot 16 - 3 \cdot (-1)) / (16 - (-1)) = 35/17 = 2.0588$$

n	a	b	c	f(c)
1	2.00000	3.00000	2.05882	-0.39080
2	2.05882	3.00000	2.08126	-0.14720
3	2.08126	3.00000	2.08964	-0.05468
4	2.08964	3.00000	2.09274	-0.02020
5	2.09274	3.00000	2.09388	-0.00745
6	2.09388	3.00000	2.09431	-0.00275

After 6 iterations $c = 2.09431$, already correct to about 3 decimal places (true root 2.09455).

Note: In false position one end-point often stays fixed (here $b = 3$), so the interval does not shrink to zero; convergence is linear but usually much faster than bisection. The stopping test must therefore use $|f(c)|$ or $|c_n - c_{n-1}|$, not the interval length.

1.5 Newton–Raphson Method

Newton–Raphson uses the **tangent line** at the current estimate. If x_n is an approximation to the root, the tangent at $(x_n, f(x_n))$ is $y = f(x_n) + f'(x_n)(x - x_n)$. Its x -intercept is the next approximation. Setting $y = 0$ gives

$$x_{n+1} = x_n - f(x_n) / f'(x_n), n = 0, 1, 2, \dots$$

The formula can also be derived from the Taylor series $f(x + h) \approx f(x) + h f'(x)$: setting $f(x + h) = 0$ gives $h = -f(x)/f'(x)$.

Algorithm

- Choose x_0 close to the root and a tolerance ϵ .
- Compute $f(x_n)$ and $f'(x_n)$. If $f'(x_n) = 0$, stop (method fails).
- Compute $x_{n+1} = x_n - f(x_n)/f'(x_n)$.
- If $|x_{n+1} - x_n| < \epsilon$, stop; otherwise repeat.

Example 1.4

Find the root of $f(x) = x^3 - 2x - 5 = 0$ near $x_0 = 2$ using Newton–Raphson. Here $f'(x) = 3x^2 - 2$.

Solution

$$x_{n+1} = x_n - (x_n^3 - 2x_n - 5) / (3x_n^2 - 2). f(2) = -1, f'(2) = 10, \text{ so } x_1 = 2 - (-1)/10 = 2.1.$$

n	x_n	$f(x_n)$	$f'(x_n)$	x_{n+1}
0	2.000000	-1.000000	10.0000	2.100000
1	2.100000	0.061000	11.2300	2.094568
2	2.094568	0.000186	11.1616	2.094551
3	2.094551	0.000000	11.1614	2.094551

The root is 2.094551 (correct to 6 decimal places after 4 iterations, and the values agree from $n = 2$).

Example 1.5 — Square root by Newton's method

Compute $\sqrt{2}$ by solving $f(x) = x^2 - 2 = 0$ with $x_0 = 1.5$.

Here $x_{n+1} = x_n - (x_n^2 - 2)/(2x_n) = \frac{1}{2}(x_n + 2/x_n)$ (the ancient Babylonian formula).

n	x_n	x_{n+1}
0	1.50000000	1.41666667
1	1.41666667	1.41421569
2	1.41421569	1.41421356
3	1.41421356	1.41421356

The number of correct digits roughly doubles at each step — the signature of quadratic convergence.

Convergence and Failure of Newton–Raphson

If f , f' , f'' are continuous, $f'(\alpha) \neq 0$ and x_0 is close enough to α , the error $e_n = \alpha - x_n$ satisfies $|e_{n+1}| \approx |f''(\alpha) / (2 f'(\alpha))| \cdot e_n^2$. The order of convergence is therefore **2** (quadratic). The method may fail when:

- $f'(x_n) = 0$ or is very small — the tangent is horizontal or nearly so and the next point is far away;
- the starting value is far from the root, or near a local extremum, so the iterates jump to another root or diverge;
- the iterates cycle, e.g. for $f(x) = x^3 - 2x + 2$ with $x_0 = 0$ ($0 \rightarrow 1 \rightarrow 0 \rightarrow \dots$);
- the root is multiple ($f'(\alpha) = 0$): convergence is only linear.

Note: A sufficient condition for convergence on an interval I : $|f(x) \cdot f''(x)| < [f'(x)]^2$ for all x in I .

1.6 Solution of a System of Nonlinear Equations

Consider two equations in two unknowns: $f_1(x, y) = 0$ and $f_2(x, y) = 0$. Newton's method extends naturally using the **Jacobian matrix** J of partial derivatives. From the Taylor expansion, the corrections $\Delta x, \Delta y$ at the current point (x_n, y_n) satisfy the linear system

$$J \cdot [\Delta x, \Delta y]^T = -[f_1, f_2]^T, \text{ where } J = [[\partial f_1/\partial x, \partial f_1/\partial y], [\partial f_2/\partial x, \partial f_2/\partial y]]$$

and the new estimate is $x_{n+1} = x_n + \Delta x, y_{n+1} = y_n + \Delta y$. For two equations the corrections are (by Cramer's rule):

$$\Delta x = (-f_1 \cdot \partial f_2 / \partial y + f_2 \cdot \partial f_1 / \partial y) / D, \Delta y = (-f_2 \cdot \partial f_1 / \partial x + f_1 \cdot \partial f_2 / \partial x) / D, D = \partial f_1 / \partial x \cdot \partial f_2 / \partial y - \partial f_1 / \partial y \cdot \partial f_2 / \partial x$$

Example 1.6

Solve $f_1 = x^2 + y^2 - 4 = 0, f_2 = x^2 - y^2 - 1 = 0$ starting from $(x_0, y_0) = (1.5, 1.2)$.

Jacobian: $J = [[2x, 2y], [2x, -2y]], D = -8xy$.

Solution

At each step solve $J \cdot \Delta = -F$ and update. Adding the equations gives $x^2 = 2.5$ and $y^2 = 1.5$, so the exact solution is $x = \sqrt{2.5} = 1.581139, y = \sqrt{1.5} = 1.224745$.

n	x_n	y_n	f_1	f_2	x_{n+1}	y_{n+1}
0	1.500000	1.200000	-0.310000	-0.190000	1.583333	1.225000
1	1.583333	1.225000	0.007569	0.006319	1.581140	1.224745
2	1.581140	1.224745	0.000005	0.000005	1.581139	1.224745
3	1.581139	1.224745	0.000000	0.000000	1.581139	1.224745

Note: Newton's method for systems needs a good starting point and the Jacobian must be non-singular ($D \neq 0$). Each iteration requires solving a linear system, which is the cost of quadratic convergence.

1.7 Fixed Point Iteration and Convergence

Rewrite $f(x) = 0$ in the equivalent form $x = g(x)$. A number α with $\alpha = g(\alpha)$ is a **fixed point** of g . Starting from x_0 , generate $x_{n+1} = g(x_n)$. Geometrically the iterates are the intersections of $y = g(x)$ with the line $y = x$ (a “cobweb”).

Convergence theorem. Suppose g and g' are continuous on an interval I containing the fixed point α and g maps I into itself. If $|g'(x)| \leq k < 1$ for all x in I , then the iteration converges to α for any x_0 in I , and the error satisfies $|x_n - \alpha| \leq k^n |x_0 - \alpha|$ and $|x_n - \alpha| \leq [k^n / (1 - k)] |x_1 - x_0|$.

Near the root $|e_{n+1}| \approx |g'(\alpha)| \cdot |e_n|$, so the order of convergence is **1 (linear)**; the smaller $|g'(\alpha)|$, the faster the convergence. If $g'(\alpha) = 0$ the convergence is at least quadratic (Newton's method is the fixed point iteration with $g(x) = x - f(x)/f'(x)$, for which $g'(\alpha) = 0$).

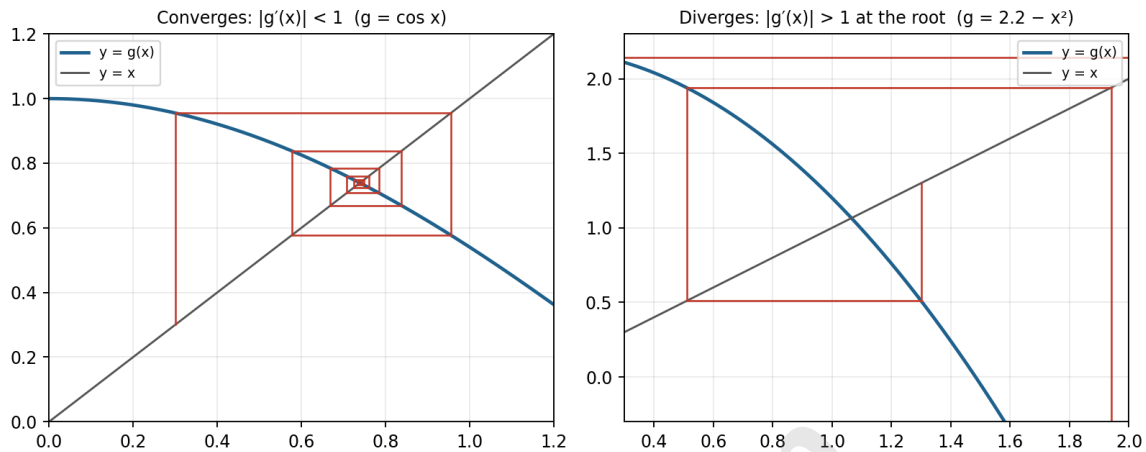


Figure 1.3 — Cobweb diagrams: convergence when $|g'| < 1$ (left) and divergence when $|g'| > 1$ (right).

Example 1.7

Find the real root of $x^3 - x - 1 = 0$ by fixed point iteration with $x_0 = 1$.

Rearranging as $x = g(x) = (x + 1)^{1/3}$ gives $g'(x) = 1 / [3 (x + 1)^{2/3}]$.

Solution

On $[1, 2]$, $|g'(x)| \leq g'(1) = 1/(3 \cdot 2^{2/3}) = 0.2100 < 1$, and g maps $[1, 2]$ into $[1.26, 1.44] \subset [1, 2]$. So the iteration converges.

n	x_n	$x_{n+1} = (x_n + 1)^{1/3}$
0	1.000000	1.259921
1	1.259921	1.312294
2	1.312294	1.322354
3	1.322354	1.324269
4	1.324269	1.324633
5	1.324633	1.324702
6	1.324702	1.324715
7	1.324715	1.324717

The iterates approach 1.324718 (the root of $x^3 - x - 1 = 0$). Choosing the other rearrangement $x = x^3 - 1$ gives $g'(x) = 3x^2$, which is about 5.3 at the root, so that iteration **diverges**. The form of

g matters!

Comparison of the Methods

Method	Needs	Order	Reliability	Cost per step
Bisection	Bracket $[a, b]$	1 (linear, factor $\frac{1}{2}$)	Always converges	1 function evaluation
False Position	Bracket $[a, b]$	1 (usually faster)	Always converges	1 function evaluation
Newton–Raphson	$x_0, f'(x)$	2 (quadratic)	Needs good $x_0, f' \neq 0$	f and f' evaluations
Fixed Point	$x = g(x), x_0$	1 (linear)	Needs $ g' < 1$	1 evaluation of g

Key Takeaways — Unit 1

- ✓ Errors: $E_a = |x - x^*|$, $E_r = E_a/|x|$, $E_p = 100 \cdot E_r$; sources are inherent, round-off and truncation.
- ✓ Bisection: $c = (a + b)/2$; needs $n \geq \log_2((b - a)/\epsilon)$ iterations; always converges, linear.
- ✓ False position: $c = [a \cdot f(b) - b \cdot f(a)] / [f(b) - f(a)]$; bracketing, usually faster than bisection.
- ✓ Newton–Raphson: $x_{n+1} = x_n - f(x_n)/f'(x_n)$; quadratic convergence; fails if $f' \approx 0$ or x_0 is poor.
- ✓ Systems: solve $J \cdot \Delta = -F$ at each step (Jacobian form of Newton's method).
- ✓ Fixed point: $x_{n+1} = g(x_n)$ converges if $|g'(x)| < 1$ near the root; convergence is linear.

— End of Unit 1 —