Bisection method

NC 07 • Module 2: Solution of nonlinear equations

Central question	How can we repeatedly halve a root bracket to obtain a reliable approximation?
Lecture focus	Bracketing, midpoint iteration, error bounds, stopping criteria and implementation.
Prerequisite	Continuity, intermediate value theorem and root isolation (NC 06).

Learning outcomes

By the end of this lecture, students should be able to...

Explain	State the conditions under which bisection guarantees convergence.
Compute	Construct iteration tables and update the bracket correctly.
Evaluate	Choose a stopping tolerance and estimate the required iteration count.

Principle of bisection

Start with a continuous function f on [a,b].

Initial bracket	Check f(a)=0 or f(b)=0 first. Otherwise require opposite endpoint signs.
Midpoint	Compute c = (a+b)/2 and evaluate f(c).
Preserve the root	Keep the half whose endpoints have opposite signs; stop immediately if f(c)=0.

Geometric interpretation

Every iteration halves the width of the bracket.

Initial interval	[1,2] contains the root of $x^3 - x - 2$.
First midpoint	c = 1.5; f(c) = −0.125. Keep [1.5,2].
Second midpoint	c = 1.75; f(c) > 0. Keep [1.5,1.75].







Decision rule

Let c be the midpoint of the current bracket.

Case 1	If f(c) = 0, return c as an exact root (in exact arithmetic).
Case 2	If f(a) and f(c) have opposite signs, set b = c.
Case 3	Otherwise set a = c. The new bracket still contains at least one root.

Algorithm: bisection

Input: continuous f, bracket [a,b], tolerance ε > 0.

Initialize	Check endpoints, confirm a < b and opposite signs.
Iterate	Compute c = a + (b−a)/2. If f(c)=0 or (b−a)/2 ≤ ε, return c.
Update	Keep the sign-changing half; repeat, with a maximum iteration safeguard.

Example 1: initial bracket

Find a positive root of $f(x) = x^3 - x - 2$.

Evaluate endpoints	$f(1) = -2$; $f(2) = 4$. Continuity guarantees a root.
Iteration 1	$c_1 = 1.5$; $f(c_1) = -0.125$. New bracket $[1.5,2]$.
Iteration 2	$c_2 = 1.75$; $f(c_2) = 1.609375$. New bracket $[1.5,1.75]$.

Example 1: iteration table

The displayed a and b are the endpoints BEFORE each iteration.

Iterations 1–3	1: [1, 2] → c=1.5; f(c)=−0.125 2: [1.5, 2] → c=1.75; f(c)=1.609375 3: [1.5, 1.75] → c=1.625; f(c)=0.666016
Iterations 4–6	4: c=1.5625; f(c)=0.252197 5: c=1.53125; f(c)=0.059113 6: c=1.515625; f(c)=−0.034054
Current bracket	After six updates: [1.515625, 1.53125].

Example 1: accuracy

Continue until the certified midpoint error is at most 0.001.

Iteration 7	$c_7 = 1.5234375$; $f(c_7) > 0$. Keep left half.
Iteration 8–9	$c_8 = 1.51953125$ (negative); $c_9 = 1.521484375$ (positive).
Iteration 10	$c_{10} = 1.5205078125$; bound = $1/1024 \approx 0.0009766$. Return c_{10}.

Error bound

Let $W = b_0 - a_0$ be the original interval width.

Before iteration k	The current bracket width is W / 2^(k−1).
Midpoint error	$\lvert c_k - r \rvert \le W / 2^k$ for a root r in the current bracket.
After k updates	New width = W / 2^k; its midpoint has error at most W / 2^(k+1).

Required iteration count

Choose an absolute error tolerance ε > 0.

Derive	W / 2^k ≤ ε implies k ≥ $\log_2(W/\varepsilon)$.
Formula	k = max(1, ceiling($\log_2(W/\varepsilon)$)) midpoint evaluations suffice.
Example	W = 1, ε = 0.0001: k = ceiling($\log_2(10000)$) = 14.

Example 2: transcendental root

Solve $f(x) = \cos x - x$ on $[0,1]$ (radians).

First two steps	$c_1 = 0.5$, $f(c_1) \approx 0.377583$; $c_2 = 0.75$, $f(c_2) \approx -0.018311$.
Next two steps	$c_3 = 0.625$, $f(c_3) \approx 0.185963$; $c_4 = 0.6875$, $f(c_4) \approx 0.085335$.
Result	At $k=10$, $c_{10} = 0.7392578125$, with certified error ≤ 0.0009766.

Stopping criteria

Distinguish interval tolerance from residual tolerance.

Absolute x-error	Stop when half the current bracket width is $\leq \varepsilon$. This gives a certified bound.
Residual	A small $\lvert f(c)\rvert$ is useful but does not by itself guarantee a small root error.
Floating point	Stop with a warning if the midpoint equals an endpoint before tolerance is met.

Convergence and cost

Bisection is a robust bracketing method.

Guaranteed progress	For a continuous sign-changing bracket, nested intervals shrink to a root.
Rate	The interval-width bound decreases geometrically by a factor of 1/2 per update.
Cost	One new function evaluation per iteration if endpoint values are cached; $O(\log_2(W/\varepsilon))$ iterations for W/ε large.

Python implementation: setup

The following code assumes continuity on [a,b].

Function header

```
def bisect(f, a, b, tol=1e-6, limit=1000):
```

Validation

```
if not (a < b and tol > 0): raise ValueError("interval/tolerance")
fa, fb = f(a), f(b)
```

Endpoints and signs

```
if fa == 0: return a
if fb == 0: return b
if (fa < 0) == (fb < 0): raise ValueError("no bracket")
```

Python implementation: loop

Continue inside the function defined on the preceding slide.

Compute midpoint

```
for _ in range(limit):
    c = a + (b-a)/2; fc = f(c)
    if fc == 0 or (b-a)/2 <= tol: return c
```

Update bracket

```
    if c == a or c == b: raise ArithmeticError("stagnation")
    if (fa < 0) != (fc < 0): b = c
    else: a, fa = c, fc
```

Finish and test

```
  raise RuntimeError("iteration limit")
# Outside function: bisect(lambda x: x**3-x-2, 1, 2)
```

Practice questions

Attempt these before reviewing the solutions.

Q1

For $f(x)=x^2-5$ on $[2,3]$, compute the first three midpoints and updated brackets.

Q2

How many midpoint evaluations guarantee error $\leq 10^{-5}$ if the initial width is 2?

Q3

Why does a sign change across a discontinuity not guarantee a root?

Solutions and recap

Next: NC 08 — Regula Falsi (False Position) Method.

A1	2.5 → [2,2.5]; 2.25 → [2,2.25]; 2.125 → [2.125,2.25].
A2 / A3	Ceiling($\log_2(200000)$) = 18. Continuity is required; 1/x changes sign across zero but has no root.
Takeaway	Maintain a valid bracket, halve it, and stop using the certified midpoint error bound.