

FUNDAMENTALS OF AI-AUGMENTED ENGINEERING: A CONVERSATIONAL APPROACH

Taner Onsay, Ph.D. & Sebnem Onsay
Authors
September 2026 · Revision 1.0

© 2026 AI First Engineering



The engineering workflow you will use throughout this book
by Taner & Sebnem Onsay,

CHAPTER - 3

PART 5 · ≈ 45 MIN

Accurate Enough? Validate & Decide

You can measure error, control resolution, and read convergence. One question remains — the one the engineer is paid to answer.

5

Problem



Model



Compute



Analyze



Validate



Improve



Four tools — one decision left

ERROR

Absolute — how far, in units. Relative / percent — how large against the reference scale. Context decides whether it matters.

RESOLUTION

This search's error is bounded by about one step — a property of this algorithm. Finer costs more.

CONVERGENCE

Shrinking successive differences — evidence consistent with convergence, never proof by itself.

TOLERANCE

An engineering specification. A person chooses it; a requirement justifies it.

One question pays for all four tools: is THIS answer good enough for THIS decision? Two cases follow — neither has an automatic answer.

Problem



Model



Compute



Analyze



Validate



Improve



Both methods meet the spec. Which do you buy?

METHOD A

20.36 m/s

$E_{\text{abs}} \approx 0.008 \text{ m/s}$

2,037 model evaluations

METHOD B

20.3 m/s

$E_{\text{abs}} \approx 0.068 \text{ m/s}$

204 model evaluations

SPEC
 ± 0.2
m/s

Argue it with the toolkit — error, resolution, convergence, tolerance. Then defend your purchase order.

Reveal — after the argument: both methods satisfy the stated $\pm 0.2 \text{ m/s}$ requirement. Method B is sufficient at far fewer evaluations — for THIS decision, Method A's extra resolution buys no engineering benefit. **“More accurate” is not automatically “better engineering.”**



The louder error — your reaction time returns

NUMERICAL ERROR — STEP 0.1

0.068 m/s

Step 0.1 → 0.001: 100× finer resolution, for ≈ 100× the model evaluations (204 → 20,368). The machine will oblige as long as you keep paying.

MEASUREMENT UNCERTAINTY — YOUR t_r , ± 0.1 s

≈ ± 0.5 m/s

t_r 0.9 → 20.90 / 1.0 → 20.37 / 1.1 → 19.86 m/s.
Roughly seven times the numerical error — before the search even starts.

One budget, two options: run the search at step = 0.001 — or collect additional ruler-drop trials to better characterize your reaction-time input and its variability. Which do you fund, and why?

Reveal — after the argument: sharpening the smallest error on the table is effort spent looking busy. Rank the errors; spend where the decision gains most. **Evidence + Physics + Computation + Uncertainty + Judgment.**

Problem



Model



Compute



Analyze



Validate



Improve



Eight questions every numerical answer must survive

1

What is the **approximation**?

2

Compared with what **reference** — honestly named?

3

How large is the **error** — absolute and relative?

4

What sets the **resolution**?

5

Convincing **convergence** evidence — beyond a grid coincidence?

6

What does more accuracy **cost**?

7

What accuracy does the **DECISION** require?

8

What independent evidence exists?

Questions 1–7 are this chapter's tools. Question 8 is Chapter 2's validation ladder — hand check, independent method, analytical result, physical sense — still holding everything up.

Problem



Model



Compute



Analyze



Validate



Improve



EXAMPLE: Can You Build Your Own $\sin(x)$?

Use a trusted library value to investigate an approximation you write yourself.

TRUSTED REFERENCE

Python library
`math.sin(x)`

compare



YOUR NUMERICAL FUNCTION

`own_sin(x)`
Taylor-series approximation

QUESTION

How many terms are enough
— and how will the code
know when to stop?

Taylor series for sine

$$\sin(x) \approx x - x^3/3! + x^5/5! - x^7/7! + x^9/9! - \dots$$

Start with $x = 1.0$ rad. Each loop adds ONE more term. **PREDICT:** As terms are added, what should happen to the error?

REFERENCE ≠ “EXACT.” For this experiment, `math.sin(x)` is the trusted benchmark used to judge our own approximation.

Problem



Model



Compute



Analyze

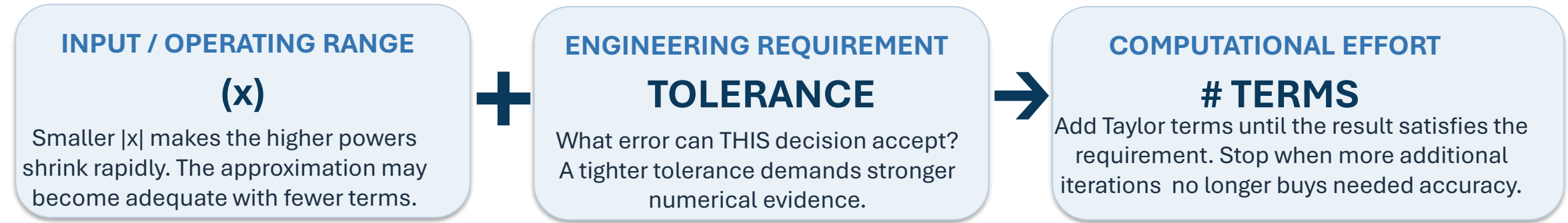


Validate



Improve

How Many Taylor Terms Do We Need? It Depends.

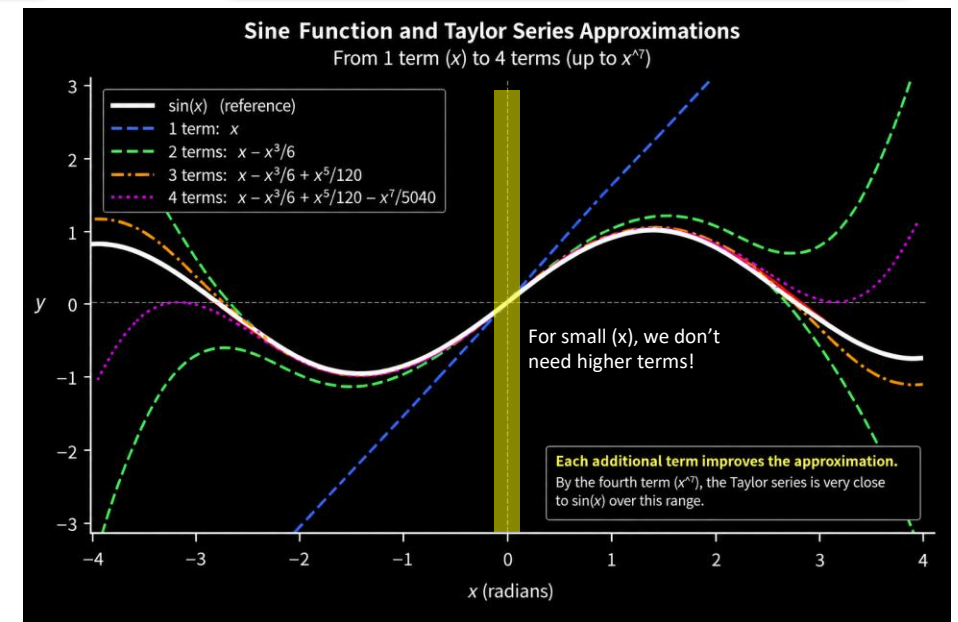


The algorithm does not choose the required accuracy — the engineering application does.

A FAMILIAR EXAMPLE

$$\sin(x) \approx x$$

For small x , the first Taylor term may already satisfy a modest engineering tolerance. For larger x — or a tighter tolerance — more terms are required.



DO NOT ASK: “How many terms does sine need?” **ASK:** “How many terms does THIS application need to meet ITS tolerance?”



A Loop Adds Terms Until the Error Meets the Tolerance

First pass: use `math.sin(x)` to make convergence and stopping visible.

Error we can measure

$$E_{\text{abs}} = | \text{reference} - \text{approximation} |$$

Stopping specification

STOP when $E_{\text{abs}} < \text{tolerance}$

WHAT THE NAMES MEAN

reference $\rightarrow$ `math.sin(x)`

approximation $\rightarrow$ our running Taylor sum

tolerance $\rightarrow$ accuracy requirement

WATCH THE LOOP: term $\rightarrow$ approximation $\rightarrow$
error $\rightarrow$ test tolerance $\rightarrow$ repeat.

`pass1_reference_guided.py`

```
import math
```

```
def own_sin_reference(x, tol):
```

```
    reference = math.sin(x)
```

```
    approx = 0.0
```

```
    n = 0
```

```
    errors = []
```

```
    while True:
```

```
        term = (-1)**n * x**(2*n + 1) / math.factorial(2*n + 1)
```

```
        approx = approx + term
```

```
        error = abs(reference - approx)
```

```
        errors.append(error)
```

```
        print(n + 1, approx, error)
```

```
        if error < tol:
```

```
            break
```

```
        n = n + 1
```

```
    return approx, errors
```

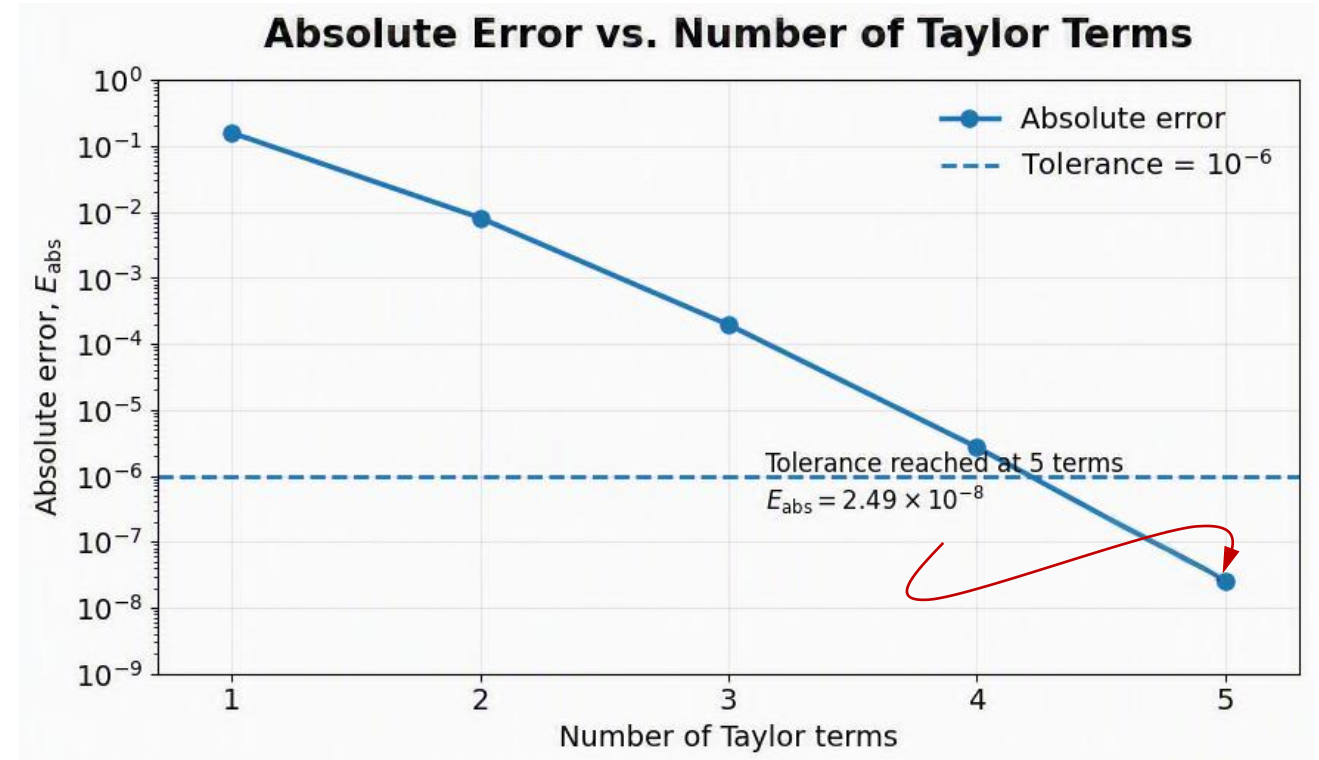
```
own_sin_reference(1.0, 1e-6)
```



Watch the Error Collapse as Terms Are Added

$x = 1.0 \text{ rad} \cdot \text{reference} = \text{math.sin}(1.0) \cdot \text{tolerance} = 10^{-6}$

Terms	own_sin approximation	E_abs
1	1.000000000	1.59×10^{-1}
2	0.833333333	8.14×10^{-3}
3	0.841666667	1.96×10^{-4}
4	0.841468254	2.73×10^{-6}
5	0.841471010	2.49×10^{-8}



CONVERGENCE EVIDENCE: the error shrinks rapidly as terms are added. **STOPPING:** the fifth term brings E_{abs} below 10^{-6} .

Problem



Model



Compute



Analyze



Validate

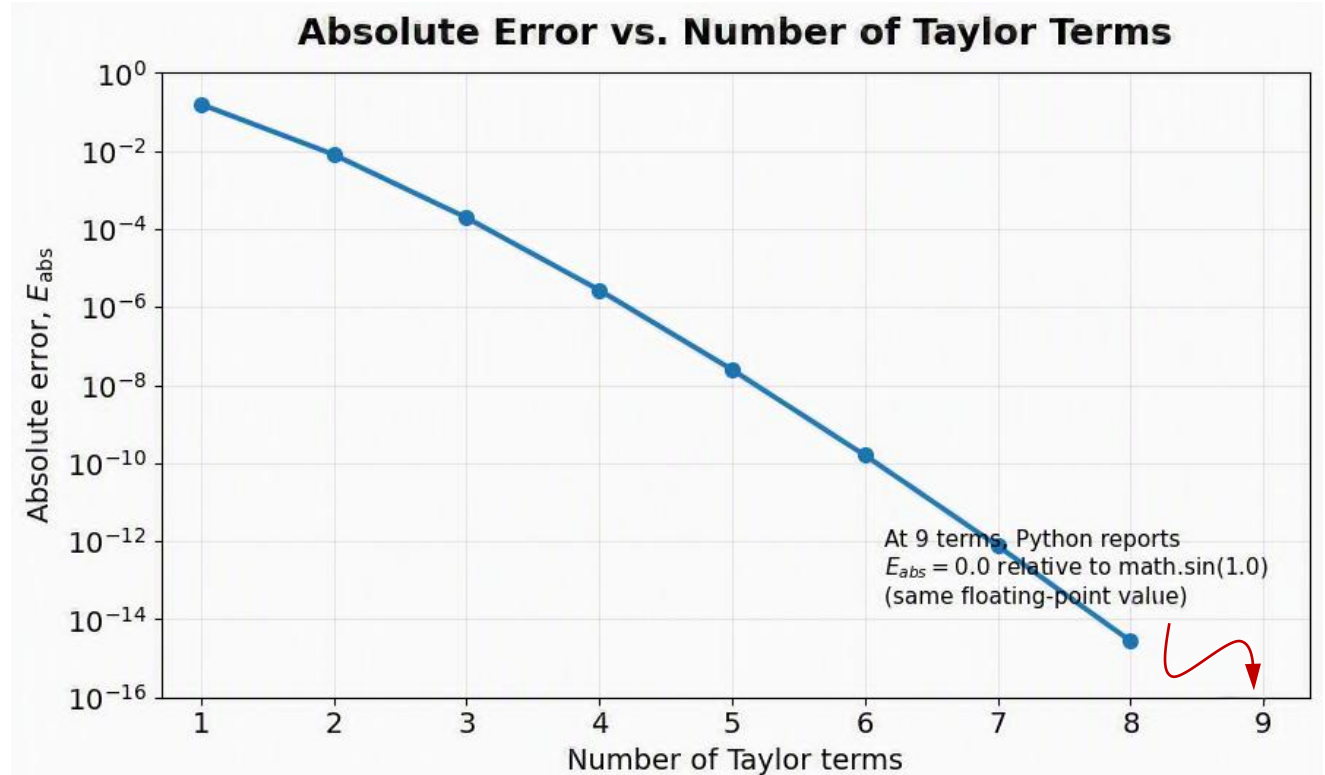


Improve



What if we decrease the tolerance? How many Taylor terms?

Tolerance	Taylor terms / iterations	Achieved absolute error
1.0E-01	2	8.14E-03
1.0E-02	2	8.14E-03
1.0E-03	3	1.96E-04
1.0E-04	4	2.73E-06
1.0E-05	4	2.73E-06
1.0E-06	5	2.49E-08
1.0E-07	5	2.49E-08
1.0E-08	6	1.60E-10
1.0E-09	6	1.60E-10
1.0E-10	7	7.62E-13
1.0E-11	7	7.62E-13
1.0E-12	7	7.62E-13
1.0E-13	8	2.78E-15
1.0E-14	8	2.78E-15
1.0E-15	9	0.0 (float match)



CONVERGENCE @ Smaller Tolerance: the error continues to shrink rapidly as terms are added. **STOPPING:** 9th term brings E_{abs} below 10^{-15} .



How Can `own_sin(x)` Stop Without Calling `math.sin(x)`?

The operational algorithm should not need to know the answer in advance.

NEW STOPPING IDEA

For this alternating Taylor series at $x = 1$ rad:
STOP when $|\text{next term}| < \text{tolerance}$.

The next omitted term gives a useful error bound for this case.

IMPORTANT DISTINCTION

Stopping criterion: used inside `own_sin()`.
Validation: compare afterward with `math.sin(x)`.

```
pass2_operational.py
```

```
import math
```

```
def own_sin(x, tol):
```

```
    approx = 0.0
```

```
    n = 0
```

```
    while True:
```

```
        term = (-1)**n * x**(2*n + 1) / math.factorial(2*n + 1)
```

```
        if abs(term) < tol:
```

```
            break
```

```
        approx = approx + term
```

```
        n = n + 1
```

```
    return approx
```

```
# VALIDATE AFTER the function is finished
```

```
x = 1.0
```

```
tol = 1e-6
```

```
y_own = own_sin(x, tol)
```

```
y_ref = math.sin(x)
```

```
error = abs(y_ref - y_own)
```

```
print(y_own, y_ref, error)
```

The library reference is now **OUTSIDE** your function — used to validate it, not to make it work.



The Chapter 3 hunt

**The code is clean. The number is plausible.
The AI says it is “accurate.” Prove that — or break it.**

In Chapter 1 you hunted a wrong formula. In Chapter 2, the algorithm itself. Now you hunt the accuracy claim — the step size it never mentioned, the resolution it cannot exceed, the convergence evidence it never gathered, the tolerance nobody chose.

Problem



Model



Compute



Analyze



Validate



Improve



Live demonstration — the class attacks the claim

THE DEMO — RIGHT NOW

We ask an AI, casually:

“Write a Python program to find the max safe speed for a car with 50 m to stop — and tell me how accurate the answer is.”

We run whatever it produces. It will run — and it will claim accuracy. Then the class attacks the claim, not the syntax.

What step size — and did it say?

What resolution — which digits CAN it claim?

What reference or convergence evidence exists?

E_abs, E_% — against what, exactly?

Does the decision need hundredths at all?

Even if it converges — is the MODEL validated?

Verdicts: ACCEPT · MODIFY · REJECT — with evidence. If the claim survives your attack, state exactly how you verified it. That is also engineering.



From a number → to a number + evidence

In Lab 3 you will apply this chapter's numerical-accuracy toolkit to an engineering computation — practicing how to:

- 1 Examine the numerical error in a computed result
- 2 Investigate the resolution — what the computation can claim
- 3 Look for convergence evidence as a parameter is refined
- 4 Connect a tolerance to an engineering requirement
- 5 Report the result with justified precision
- 6 Defend whether the result is accurate enough for its purpose

NUMBER → NUMBER + EVIDENCE

The laboratory will ask you to support a numerical answer with evidence — not simply produce the number.

The detailed problem, procedure, data, and deliverables will be provided with the Lab 3 instructions.

Support the answer with evidence. Producing the number is where the laboratory begins — not where it ends.

Problem



Model



Compute



Analyze



Validate



Improve

So far, we had a luxury: a reference answer

THE STOPPING-DISTANCE STUDY

20.0 m/s vs. 20.37 m/s

The analytical solution gave us a trusted reference value.

That let us isolate numerical error and study step size, resolution, convergence, tolerance, and cost in a clean setting.

REAL ENGINEERING IS MESSIER

What if the answer also depends on measurements, an estimated parameter, and an imperfect model?

Then refining the computation is only one part of the accuracy story.

It is time to test Chapter 3 against the physical world.

The numerical toolkit stays the same. The evidence is about to become physical.



When Is the Water Ready?

When will warm water cool to 40 °C — and how accurately can we predict when it happens?

PHYSICAL WORLD

Mug + warm water
Thermometer readings
Observed cooling curve
Measured 40 °C crossing



COMPARE



ERROR
REFINE
VALIDATE
DECIDE

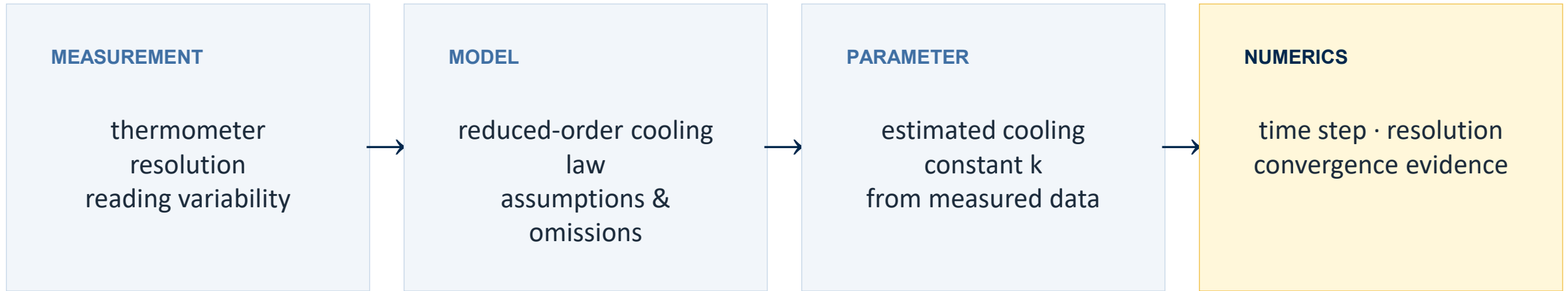
COMPUTATIONAL WORLD

Reduced-order cooling model
Estimate the cooling constant k
Predict the cooling curve
Predicted 40 °C crossing





Now there is more than one kind of “wrong”



PREDICTED 40 °C CROSSING TIME

A converged numerical answer can still disagree with the experiment.

CONVERGED $\neq$ CORRECT

Refining the time step cannot repair poor measurements or an inadequate model.



Explore / Practice · Prepare for Lab 3

EXPLORE / PRACTICE — SHORT, PREDICTION-FIRST

- 1 · **By hand:** a predicted 40 °C crossing time and a measured crossing time are supplied. Compute the absolute and percent error — and state the justified digits.
 - 2 · **Predict, then refine:** reduce the cooling prediction's time step. Predict what will change BEFORE running — then run, compare successive crossing times, and reconcile.
 - 3 · **Grade an AI accuracy claim:** “The water will reach 40 °C at 18.347 min, so the prediction is highly accurate.” Three sentences: what is supported, what is overclaimed, what evidence is missing.
 - 4 · Read a **stopping criterion** for the cooling prediction. State it as an engineering sentence — explain what the tolerance means physically, and who should choose it.
- 2 · **Predict, then refine:** a provided computation has one numerical parameter. Predict what refining it will change BEFORE running — then run and reconcile.
 - 3 · **Grade an AI accuracy claim:** an AI reports “ $v_{\text{max}} = 20.367864366808018$ m/s (exact)”. Three sentences: what is right, what is overclaimed, what evidence is missing.
 - 4 · Read a **stopping criterion** in a provided refinement loop. State it as an engineering sentence — and explain what the tolerance means, and who should choose it.

BE READY TO

Use Excel for a short numerical table

Run a provided Python cell in your notebook environment

Compute absolute and relative / percent error by hand

Explain resolution, convergence evidence, and tolerance in words

Justify the precision of a reported result

These practice items now point directly into Lab 3: predict the 40 °C crossing, quantify error, refine the time step, examine convergence, and defend the tolerance.

What must be true before the laboratory

**A NUMERICAL ANSWER IS INCOMPLETE
UNTIL YOU KNOW ITS ACCURACY**

CONVERGED $\neq$ CORRECT

MORE DIGITS $\neq$ MORE KNOWLEDGE

The engineer measures the error, understands the resolution, examines the convergence, justifies the tolerance, weighs the other uncertainties — and decides whether the answer is good enough for its purpose.

Ahead: your search WALKED to the answer — 2,037 evaluations to earn two decimal places. Chapter 4 asks the question this chapter earned: can we reach the required accuracy more intelligently? The same road returns, written a new way: $\mathbf{f(v) = 0}$.

Problem

›

Model

›

Compute

›

Analyze

›

Validate

›

Improve

AI = Coding Companion → Emerging Engineering Analyst: “Help me understand what the numerical results are telling me — not just produce more numbers.”



What we established in this Chapter Numerical Accuracy & the Emerging Analyst:

Every numerical answer is an approximation whose accuracy must be assessed.

Step size sets resolution; refinement costs computation.

Convergence is evidence — agreement is not proof.

Tolerance is a specification someone must own.

CONVERGED ≠ CORRECT · MORE DIGITS ≠ MORE KNOWLEDGE.



AI COMPANION PROMPT: Convergence-Evidence Prompt:

“Here are my results for several step sizes. Do not tell me which answer to accept yet. Help me examine how the solution changes as the step size decreases. Ask me what accuracy the engineering decision actually requires, and help me decide what additional evidence I should check.”



ENGINEER'S CHECK BOX:

Reporting a number? Report its error and units too.

Two resolutions agree? Evidence, not proof — check a third.

A tolerance in the code? You chose it — be ready to justify it.

Converged? Not yet correct — validate the model.



AI COMPANION PROMPT: Go One Step Deeper Prompt:

“Create a water-cooling scenario where numerical error, thermometer uncertainty, and model uncertainty compete. Do not resolve it for me — question me until I can defend which source of error the engineer should spend effort reducing first, and why.”