

FUNDAMENTALS OF AI-AUGMENTED ENGINEERING

CHAPTER 2/LAB-2: Engineering Lab (3 Hours)



Reaction Time & Safe Stopping Distance

Can we turn a real measurement into an algorithm — and defend the decision it produces?

Taner Onsay, Ph.D. & Sebnem Onsay
Authors

August 2026 · Revision 1.0

© 2026 AI First Engineering

Problem



Model



Compute



Analyze



Validate



Improve

Today your own reaction time becomes a defended engineering decision

Reaction Time & Safe Stopping Distance

What is this lab about? Turn a real measurement — your own reaction time — into a defended engineering decision, by way of an algorithm you design, code, test, and validate:

Measure → Model → Algorithm → Code → Test → Validate → Decide
(DO › OBSERVE › RECORD › TALK TO AI › EVALUATE › DECIDE › DO ...)

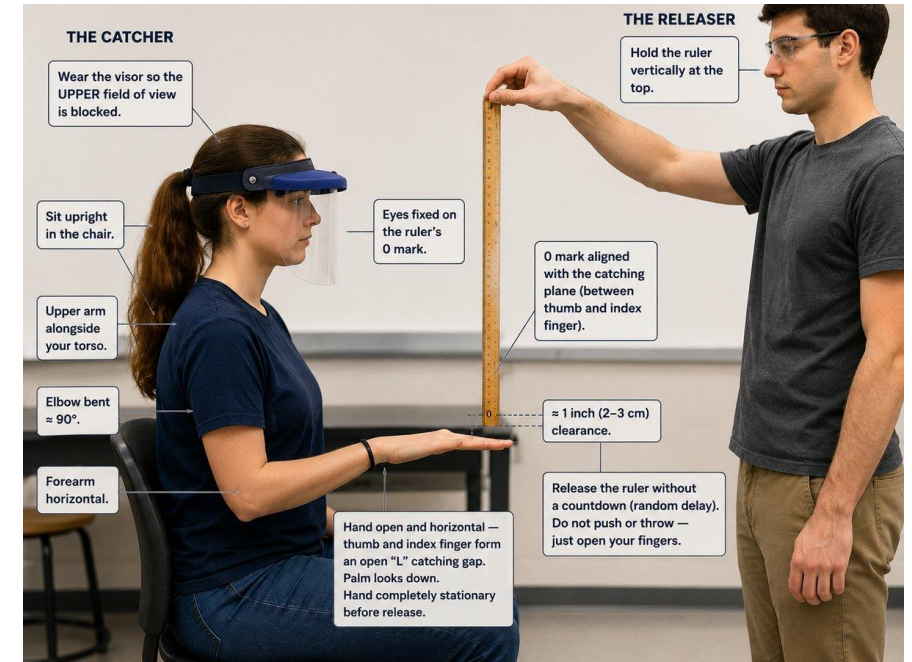
LEARNING OBJECTIVES

By the end of this lab, you will be able to:

- Measure a physical quantity using a validated model as the instrument
- Express an engineering procedure as pseudocode, then as Python
- Build functions and a WHILE search with a defensible stopping condition
- Use AI as a programming coach — while keeping the algorithm yours
- Test deliberately and debug from evidence — expected vs. actual
- Validate independently and defend a maximum safe approach speed

TOOLS YOU WILL USE

- **Wooden ruler + visor** — measure your own reaction time
- **Excel** — make the repeated calculation visible
- **Python / Jupyter** — functions and the WHILE search
- **AI assistant** — programming coach and critic, never the source of the logic
- **Your engineering judgment** — decide what you trust

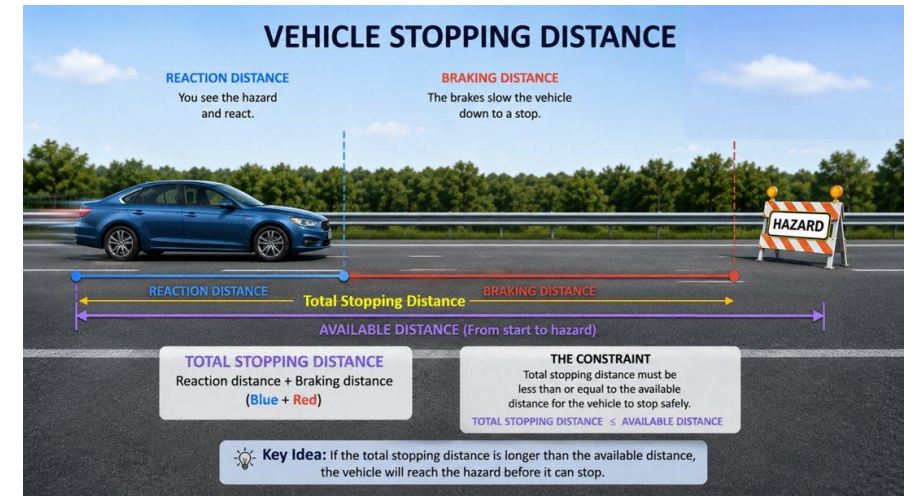




What you will do (step-by-step):

- 1 Readiness quiz — then verify Excel, Python, AI workspace
- 2 Measure your reaction time — 5 valid ruler-drop trials each
- 3 Convert catch distance $\rightarrow$ reaction time (Chapter 1 model $\cdot$ cm $\rightarrow$ m!)
- 4 Meet the stopping-distance problem — commit P1–P3 first
- 5 Excel — run the search by hand $\cdot$ find the SAFE/UNSAFE bracket
- 6 Write pseudocode — then hand it to AI for critique, not code
- 7 Python — three functions, then the WHILE search
- 8 Test the battery $\cdot$ BREAK THE AI $\cdot$ debug from evidence
- 9 Validate four ways $\cdot$ wet road $\cdot$ make the engineering decision
- 10 Complete the record $\rightarrow$ report handoff $\cdot$ final review before submission

Would you confidently put your name on your final result? *You will — literally — in the report's approval section.*





THINK › PSEUDOCODE › AI › CODE · RUNS ≠ CORRECT

In Chapter 2 the AI role advances — from tutor to programming / coding companion — while you remain the lead engineer: the algorithm, the assumptions, and the decision stay yours.

THE RECURRING LOOP — INSIDE PROBLEM › MODEL › COMPUTE › ANALYZE › VALIDATE › IMPROVE



... and back to DO — the conversation evolves as the experiment evolves.



You, not AI: design the algorithm, decide the boundary rule, verdict every suggestion — ACCEPT · MODIFY · REJECT — approve the result.



AI, for you: syntax help, pseudocode critique, translation, error explanation, test ideas — on the logic you provide.

AI assists. Humans observe, verify, decide, and approve.

Never ask AI to invent what you did. Tell AI what you actually did.



AI as coding companion — who does what



HUMAN TEAM — LEAD ENGINEER

- Own the problem, the model, the algorithm
- Perform the ruler experiment and measurements
- Write the pseudocode before AI sees it
- Decide the boundary rule ($\leq$ or $<$)
- Verdict every AI suggestion: ACCEPT · MODIFY · REJECT
- Test, validate, and interpret the evidence
- Approve the final engineering decision



AI CODING COMPANION

- Explain syntax and error messages
- Critique your pseudocode — logic, units, edge cases
- Translate your pseudocode into Python
- Suggest debugging strategies and tests
- Propose additional test cases
- Help organize functions and the record
- Draft report text from your evidence

AI assists with the code. The algorithm — and the responsibility — remain yours.



Create your AI workspace — the engineering record

ONE WORKSPACE · “LAB-2 — REACTION TIME & STOPPING DISTANCE” (Workspace means new PROJECT in ChatGPT or Claude)

- Lab instructions + report template (Word docs or pdfs)
- Trial table · every catch distance
- Assumptions · predictions P1–P3 (committed)
- Excel table + recorded bracket
- Pseudocode versions · AI critiques · your verdicts
- Python code · tests · BREAK THE AI findings
- Validation evidence · decisions · conclusions

The record documents what happened.

The report communicates what matters.

Feed the record as you go — don't reconstruct it at 2:45.

STARTER PROMPTS

PROMPT-1

“Here are the corresponding lecture notes, lab instructions and the lab report template. Just read them for later reference and wait for my command.” (drop or attach PDF versions of all 3 documents right into this chat)

PROMPT-2 *“You are our AI coding companion for the LAB-2 Reaction Time & Safe Stopping Distance Lab. We are the lead engineers. Coach us on syntax, critique our pseudocode when we ask, translate only what we hand you, and suggest tests — but never change our engineering assumptions, our boundary rule, or our units without flagging the change explicitly. Never invent measurements, results, or conclusions. This workspace is the engineering record for our final report.”*

Any capable AI platform works — a project/workspace feature (e.g., ChatGPT Projects) is convenient, but one persistent conversation is enough.



Engineering first, software second

0:00–0:10		Phase 0 · Readiness quiz + verify Individual quiz — concepts, not syntax · Excel, Python, AI workspace proven working.
0:10–0:35		Part 1 · Ruler-drop experiment Measure every member's reaction time — 5 valid trials each, roles rotating.
0:35–0:45		Part 2 · Catch distance → reaction time Chapter 1 model inverted: $t_r = \sqrt{2d/g} \cdot \text{cm} \rightarrow \text{m!}$ · one hand-checked trial each.
0:45–0:55		Part 3 · The stopping-distance problem Givens and assumptions recorded · predictions P1–P3 committed in writing.
0:55–1:15		Part 4 · Excel — make repetition visible Run the search by hand, row by row · find and record the SAFE/UNSAFE bracket.
1:15–1:30		Part 5 · Pseudocode → AI critique Your algorithm on paper first · AI critiques; you verdict ACCEPT · MODIFY · REJECT.
1:30–2:00		Part 6 · Python — functions + WHILE Three testable functions, then the search · translate YOUR pseudocode.
2:00–2:25		Part 7 · Test · BREAK THE AI · debug Expected before actual · attack your own program · evidence-first debugging.
2:25–2:50		Part 8 · Validate · wet road · decide Four checks + independence · $a = 3.5 \text{ m/s}^2$ rerun · the decision statement.
2:50–3:00		Part 9 · Record → report handoff Evidence complete · missing items listed · human review — then sign.



Teams & logistics — everything runs in parallel

EACH STATION'S KIT — IDENTICAL AND INDEPENDENT



Wooden metric ruler

~30 cm, legible markings from 0 · optional string-release rigging



Chair — the catch zone

Placed clear of desks, bags, and walkways. Only catcher and releaser inside.



Visor-style face shield

Blocks the catcher's upward view — anticipation is the enemy of the measurement.



Team laptops

Excel · Python/Jupyter · AI assistant access · at least one working per team

TEAM RULES

- Rotate roles every trial set: catcher · releaser · recorder.
- Everyone gets measured; everyone records.
- One engineering record per team — all members named, individual data attributed. Team number, members, date, station at the top NOW.
- Claim a station and stay clear of other teams' catch zones.

EXPERIMENTAL DESIGN: each student, five valid trials → one representative reaction time per member. Individual data — team decision.



Safety & conduct — respect the measurement



SAFE STATION

- Clear catch zone — chair away from desks, bags, walkways
- Only the catcher and releaser inside the zone
- Dropped means dropped — never lunge or dive for a falling ruler
- A missed catch is an invalid trial, repeated at no cost
- Release means release — no throw, push, or string-pull



HONEST DATA

- No countdowns, no rhythms — anticipation contaminates the data
- Reaction-time data is personal — recorded honestly, never edited
- Variation is the phenomenon, not a scoreboard — no teasing
- No distracting other teams' catchers
- Failed trials, bugs, and rejected AI advice belong IN the record

Honest records beat pretty records. The journey is the evidence.



Readiness check — concepts, not syntax



READINESS QUIZ

- ☐ 5 questions · 5 minutes
- ☐ Individual · closed notes · no AI
- ☐ Collected before rulers come out



EXCEL + PYTHON

- ☐ Excel opens and computes
- ☐ Notebook runs a test cell
- ☐ You can interrupt the kernel



AI WORKSPACE

- ☐ “LAB-2” workspace created
- ☐ Starter prompts pasted
- ☐ Instructions + template uploaded



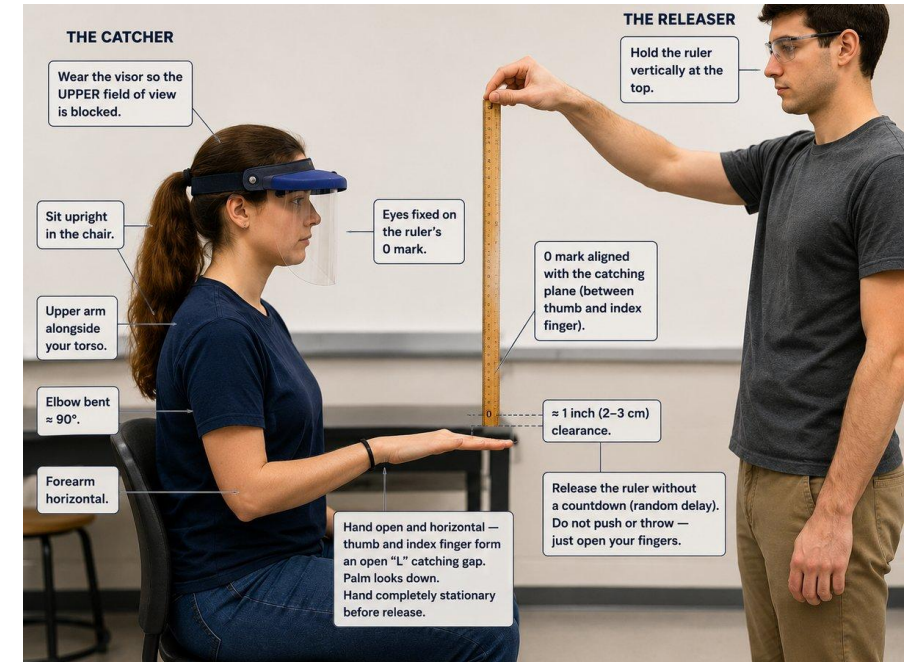
If something is broken

- Ask your instructor/lab facilitator for help — don't burn 40 minutes alone
- Paste the exact error into AI and ask what it means (often faster than you'd think)
- Keep your team moving on one working machine meanwhile

*All nine green? CHECKPOINT ① — the TA reviews quiz answers before equipment comes out.
Experienced with another IDE? Fine — as long as you can run and modify a notebook and support yourself.*

The standardized setup — identical at every station

- 1 Catcher: sit upright · visor blocks your upward view
- 2 Upper arm along torso · elbow $\approx 90^\circ$ · forearm horizontal
- 3 Hand open — thumb and index form an “L” · palm down · perfectly still
- 4 Eyes fixed on the ruler's 0 mark — never on the releaser
- 5 Releaser: hold the ruler vertically ABOVE the catching hand
- 6 Align the 0 mark exactly with the catching plane
- 7 Keep ~ 1 inch (2–3 cm) clearance — the ruler never touches the hand
- 8 NO countdown · vary the delay 1–5 s, unpredictably · simply let go
- 9 String release (if fitted): hold the loop, not the ruler — open your fingers
- 10 When the ruler drops: close the hand and grab it in the air — never chase it



Why so strict? Every rule removes one source of error — that is what a standardized procedure is FOR. An anticipated catch measures prediction, not reaction.



Five valid trials per student

1

Predict your own catch distance first

Before trial 1, write a guessed catch distance in the record. After trial 5, compare — was your self-model calibrated?

2

Run trials until FIVE are valid

VALID: caught in the air. Read the catch distance at the TOP edge of the grip, in cm, to the nearest 0.5 cm.

3

Log every trial — invalid ones too

INVALID: missed, anticipated, hand moved early, bad release. Record it and repeat — never invent a number for a miss. Rotate roles after each student's set.



TALK TO YOUR AI TEAMMATE “Here is our trial table so far — log it, and flag any student whose five values look inconsistent.”

TRIAL NOTES

Typical catch distances: 10–25 cm. If someone consistently misses, move the starting gap slightly closer — and note it. Invalid-trial tallies stay in the record; they are data about the procedure.

Also record: who caught, who released, and anything odd about each trial.



ENGINEER DECIDES validity is called at the station by the humans who watched the trial — never edited afterward.

Problem ›

Model ›

Compute ›

Analyze ›

Validate ›

Improve



From catch distance to reaction time



The Chapter 1 model — inverted

$d = \frac{1}{2} g t^2$ becomes $t_r = \sqrt{2d/g}$ — free fall, pointed backwards



Convert **BEFORE** the equation:
centimeters → METERS



Hand-check **ONE** of your own trials — on paper



Then automate: `reaction_time(d_cm)` — your first function

cm → m inside, seconds out — must reproduce your hand check

$$t_r = \sqrt{2d/g} \quad 18.0 \text{ cm} = 0.180 \text{ m} \rightarrow 0.19 \text{ s}$$

THE UNIT GATE — cm → m FIRST

- Feeding 18 (cm) instead of 0.18 (m) gives $t_r \approx 1.9 \text{ s}$ — ten times too slow.
- $\sqrt{100} = 10$: the cm mistake is exactly a factor of ten in time.
- The same mistake will try to sneak into your Python later. Let it fail here, on paper, where it is cheap.
- While the ruler falls freely, you are reacting — its fall distance is a clock.

Typical results: 0.14–0.25 s. Outside that range? Check the units first.

Problem



Model



Compute



Analyze



Validate



Improve



Every member measured — one table, no discards

Student	T1 (cm)	T2	T3	T4	T5	Invalid	t_r avg (s)
Maya	18.0	21.5	16.5	19.0	20.0	1	0.20

One row per team member — five valid catch distances, the invalid tally, and the average. Copy this table into your engineering record.

t_r per trial = $\sqrt{2d/g}$, d in METERS · student t_r = five-trial average · DEFAULT team t_r = mean of the members' averages. Another representative value is allowed ONLY if the choice is recorded and justified.

Record ALL trials — valid and invalid. An inconvenient measurement is still a measurement; deleting it is how bias is born.



TALK TO YOUR AI TEAMMATE

“Here is our trial table and the output from our `reaction_time()` function. Check our calculations, units, and organization. Flag anything inconsistent — but do not replace our computation or reinterpret the data.”



ENGINEER DECIDES

recompute one trial by hand — if AI's arithmetic fails there, nothing downstream is trusted.

CHECKPOINT ② — trial table · hand check · working `reaction_time()` · team representative t_r + rationale.

Problem



Model



Compute



Analyze



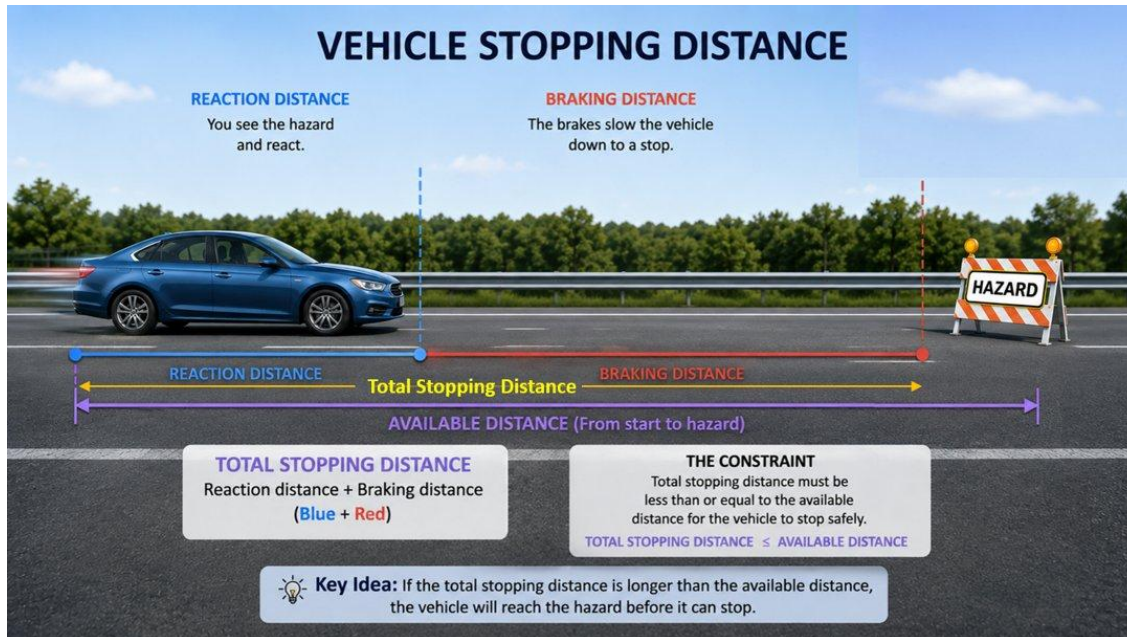
Validate



Improve



A vehicle, a hazard, 50 m of road — and your reaction time



FIND: the maximum safe approach speed v_{max} (m/s)

...at which the vehicle still stops within $d_{\text{avail}} = 50 \text{ m}$ — using YOUR measured reaction time.

The model: $d_{\text{stop}} = v \cdot t_r + v^2/(2a)$

Dry road: $a = 7.0 \text{ m/s}^2$. The boundary call — is $d_{\text{stop}} = d_{\text{avail}}$ safe? — is YOURS. Write it down.

Assumptions go in the record — including this one:

TALK TO YOUR AI TEAMMATE “Log our problem inputs and assumptions: $t_r = ______$ s, $a = 7.0 \text{ m/s}^2$, $d_{\text{avail}} = 50 \text{ m}$, boundary rule: $______$.”

ENGINEER DECIDES using ruler t_r in a driving model is a modeling choice — state it; revisit in Part 8.



Predict — in writing, right now

1

P1 — Your $v_{\max}$, a committed number

In m/s AND km/h (or mph). Wrong is fine; uncommitted is not.

2

P2 — Which distance is larger at $v_{\max}$?

Reaction or braking — one word plus one sentence of reasoning.

3

P3 — Wet road: braking capability halves

Does $v_{\max}$ drop by less than half, exactly half, or more than half?
Circle one — you will check it in Part 8.



TALK TO YOUR AI TEAMMATE “Here are our committed predictions P1–P3 — log them. Do not evaluate them yet.”

WHY PREDICT FIRST?

A prediction made after seeing the data isn't a prediction — it's a story. Committing first is what makes the comparison in Part C honest, and it's how real engineering validation works.

Predictions go in the record BEFORE Phase 4 opens Excel — and are never edited afterward.



ENGINEER DECIDES the numbers you commit are yours — AI logs them, you own them.

Problem ›

Model ›

Compute ›

Analyze ›

Validate ›

Improve



Excel — watch yourself run the algorithm

Speed v (m/s)	Reaction v·t_r (m)	Braking $v^2/(2a)$ (m)	Total d_stop (m)	Safe? (≤ 50)
5	=A2*t_r	=A2^2/(2*a)	=B2+C2	=D2<=d_avail
10	...	...	...	...
15	...	...	...	...
: step 5, then refine	...	...	...	flip point?

Build the row for ONE speed and check it by hand · fill down v = 5, 10, 15, 20, 25, 30 · find where SAFE flips to UNSAFE · refine between those rows with step 1, then 0.5.

Record the bracket: “v_max is between ____ and ____.” Compare with P1 now. Struggling teams: starter sheet OK — but write the four formulas yourselves.

Constants live in NAMED cells — t_r, a, d_avail — so formulas read like the model. Never retype numbers.



TALK TO YOUR AI TEAMMATE “Do not build this sheet for us — we drive. Check that our four formulas reference the constants block correctly.”



ENGINEER DECIDES as you fill row after row, say out loud what you are repeating — calculate, compare, decide, repeat, stop. That sentence is your algorithm.

Phase 5 does not invent anything new — it writes your sentence down precisely.





You are the loop — the algorithm already exists in your sheet

✓ Constants block — t_r , a , d_{avail}

INPUTS

Named values with units, referenced by every formula.

✓ Row formulas — $v \cdot t_r$, $v^2/(2a)$, the sum

PROCESS

The same calculation, repeated on every row.

✓ The Safe? column — $d_{\text{stop}} \leq d_{\text{avail}}$

DECISION

A condition. The engineering lives in $\leq$ vs $<$.

✓ Fill-down, row after row

REPETITION

You were the loop — by hand.

🔗 You stopped adding rows at the flip

STOPPING CRITERION

Repetition ended when a condition changed — state exactly when.

⚠ The bracket you recorded

OUTPUT

The number the search reports — with units.

“I am repeating essentially the same calculation over and over. Can I tell the computer to do this?” Nothing new is invented in Phase 5 — it is only written precisely.



Pseudocode — your algorithm, on paper first

SKELETON — SHAPE ONLY

```
GET    inputs, with units — t_r [s] · a [m/s2] · d_avail  
[m] · step [m/s]  
SET    v = 0  
WHILE ( ...the next speed is still safe... ):  
    ...  
REPORT ...the last safe speed, with units...
```

The two classic gaps: no explicit stopping criterion — “loop until unsafe” of WHAT? —

and testing v instead of $v + \text{step}$, which reports the first UNSAFE speed.

YOUR PSEUDOCODE MUST CONTAIN:

- ☐ Named inputs, with units
- ☐ The stopping-distance calculation, as a formula
- ☐ The safety decision, as a comparison
- ☐ A WHILE with an explicit stopping criterion
- ☐ The boundary call: is $d_{\text{stop}} = d_{\text{avail}}$ safe? Your choice, written
- ☐ The reported output — what number, what units
- ☐ ...all finished BEFORE any AI conversation

THINK › PSEUDOCODE › AI › CODE — you are at step 2. AI has seen nothing yet.



Hand AI your pseudocode — for critique, not replacement



1 · PROMPT

“Here is my pseudocode. Do not write Python yet. Critique the logic — unclear steps, missing cases, unit risks, my stopping criterion. Ask if my intent is ambiguous.”

If AI writes code anyway: “I asked for critique only.”



2 · JUDGE

For each AI point: is it right, for YOUR problem? Verdict every one — ACCEPT · MODIFY · REJECT — with a reason.

AI critiques; YOU edit the pseudocode.



3 · RECORD

Copy in: the suggestion (verbatim), your verdict, and WHY. If every point was correct and accepted, write that — with your verification.

Then CHECKPOINT ③ — one evaluated suggestion, minimum.

Watch for the silent swap — AI critiques sometimes “improve” your assumptions: a different boundary rule, an added brake-lag term, changed units. Any change to the PROBLEM is yours to authorize, explicitly.

Do not manufacture a rejection — the requirement is critical evaluation, not disagreement.



Python — the tasks get names first

WORKING METHOD — TRANSLATE YOUR OWN PSEUDOCODE

- AI coaches syntax; the logic stays yours
- One function at a time — test each against a hand value
- `reaction_time()` must reproduce your Part 2 hand check
- `stopping_distance(20, 1.0, 7.0)` must return 48.57 m
- `is_safe()` must encode YOUR recorded boundary rule
- Read every AI-suggested line until you can explain it to a teammate
- Unexplainable code does not enter your program

Three meaningful functions is right-sized.

Three named, testable engineering tasks.

Don't over-engineer — decomposition is why Part 7's debugging will be fast.

THE CODE

```
def reaction_time(d_cm, g=9.81):  
    d_m = d_cm / 100          # cm -> m  
    return (2 * d_m / g) ** 0.5  
  
def stopping_distance(v, t_r, a):  
    return v * t_r + v**2 / (2 * a)  
  
def is_safe(d_stop, d_avail):  
    return d_stop <= d_avail    # your boundary call  
  
# unit test the physics BEFORE any loop:  
print(stopping_distance(20, 1.0, 7.0))  
# 48.57142857142857    ·    expect 48.57 m
```

Same signatures as the lecture — call `stopping_distance(v, t_r, a)` that way everywhere, including inside the loop.



The WHILE loop replaces your fill-down

THE SEARCH — AUTOMATED

```
t_r      = 0.20    # YOUR measured value (s)
a        = 7.0     # dry road (m/s²)
d_avail  = 50.0    # available distance (m)
v, step  = 0.0, 0.5
while is_safe(stopping_distance(v + step, t_r, a), d_avail):
    v = v + step
print("Max safe speed:", v, "m/s")
```

**If the loop hangs: interrupt the kernel (you saw this in lecture),
then inspect the condition —**

testing v instead of $v + \text{step}$, or an inverted inequality, is the classic.

BEFORE YOU RUN IT — WRITE DOWN:

- ☐ Your expected output — the Excel bracket knows the neighborhood
- ☐ Why this loop must terminate — say it to your team
- ☐ What the loop tests: the NEXT speed — v ends as the last safe one
- ☐ Which line encodes your boundary rule
- ☐ Then run — compare Python vs Excel bracket vs prediction P1
- ☐ All three must tell one consistent story
- ☐ Any disagreement is evidence — chase it with the Part 7 method

With $t_r \approx 0.20$ s expect v_{max} near 25 m/s — higher than the lecture's $t_r = 1.0$ s example. Your number *SHOULD* differ.



The minimum test battery — expected values first

Test	Inputs	Expected — write it BEFORE running	Probes for
Known case	$v=20, t_r=1.0, a=7$	$d_{\text{stop}} = 48.57 \text{ m}$ (hand calc)	implementation slips
Boundary	inputs landing $d_{\text{stop}} = d_{\text{avail}}$ exactly	safe or unsafe — per YOUR recorded rule	$\leq$ vs $<$ confusion
Edge	$v = 0 \cdot d_{\text{avail}} = 0 \cdot \text{step} = 10$	$0 \text{ m} \cdot v_{\text{max}} = 0 \cdot \text{coarse but sane answer}$	hidden assumptions
Nonsense probe	your t_r with d in cm (uncorrected)	absurd $t_r \approx 10\times$ too big — is it caught?	unit discipline
Monotonic check	$v = 10, 20, 30$ in sequence	d_{stop} strictly increasing	broken physics

Log every test in the record: *inputs · expected · actual · verdict*. A failed test is a finding, not an embarrassment.

Constructing the boundary case — inputs landing d_{stop} exactly on d_{avail} — is itself good thinking: solve backwards, or tune d_{avail} .

Expected before actual — the TA may ask to see the expected column filled while actual is still empty.



TALK TO YOUR AI TEAMMATE

“Propose extra tests for this program.” Adopt any that probe something this battery misses — and say what it probes.



ENGINEER DECIDES

RUNS $\neq$ CORRECT has four levels — running was only level 1. This battery is how you climb the other three.

A failed test here is a gift: the Part 7 debugging method is next.

Problem



Model



Compute



Analyze



Validate



Improve



Part 7 · 2:00–2:25 · Attack your own program — then grade the AI



Attack — pick at least three

Feed km/h where m/s belongs · land exactly on the boundary · flip the sign of a · set step = 5 · provoke an infinite loop · try a negative or nonsense input.



Re-read the AI conversation

Did AI silently change an assumption you never authorized — t_r, units, the boundary rule? Verdict what you find ›



Document every attack

What you ran, what happened, what you concluded — “survived the attack” is a finding too, if you show HOW you verified it.

VERDICT WHAT YOU FIND:



ACCEPT — with your reason



MODIFY — then retest



REJECT — with evidence



VERIFIED — also a result

BREAK THE AI:



Debug from evidence — never from vibes



1 · EVIDENCE

Pin the failing case: inputs, expected (from your hand calculation), actual (what the program printed). No explanations yet.

***Expected 48.6 — actual 68.6.
That gap is data.***



2 · LOCALIZE

Test `stopping_distance()` alone against the hand value — physics or loop? Print `v` each iteration — watch the search walk. Check the last iteration by hand.

Off-by-one lives at the stop.



3 · DIAGNOSE → FIX → RETEST

“For $v = 20$, $t_r = 1.0$, $a = 7$ I expect 48.6 m; my program returns 68.6 m. Here are my pseudocode and code. Help me identify possible causes.”

AI's diagnosis is a hypothesis — confirm it with a test before you edit.

Never just “why is my code wrong?” — that outsources both the evidence and the judgment. Give the test case + expected + actual + the smallest relevant code section.

68.6 – 48.6 = 20 m — exactly the reaction distance, counted twice. A good known case doesn't just detect the bug; it points at it.



Validate four ways — then argue about independence



Hand-calculated known case — $v = 20$ on paper vs the program

CATCHES CODE SLIPS

Must match to rounding — but it shares the model and the assumptions.



Excel vs Python — same speeds, same distances, same flip point

IMPLEMENTATION CHECK

Two calculators, one model — both can be wrong together.



Analytical limit — solve $d_{\text{avail}} = v \cdot t_r + v^2/(2a)$ exactly

METHOD-INDEPENDENT

A genuinely different method (next slide) — still the same physical model.



Physical reasonableness

MODEL-EXTERNAL

d_{stop} grows with v ? Wet road lowers v_{max} ?
Magnitudes plausible against published stopping-distance context?



Your ruler t_r inside a driving model

STATED ASSUMPTION

A simple hand reaction stands in for a full driver response — a limitation to state, not a bug.



A second AI confirming the first

EVIDENCE, NOT PROOF

Two AIs can repeat the same model choice — and the same error.

Independence is not binary. Always ask: what assumptions, data, or methods do two checks share?



The analytical limit — a disagreement worth keeping

step (m/s)	search v_max (m/s)	gap to v_exact (m/s)	loop iterations
0.5	fill in	fill in	fill in
0.1	fill in	fill in	fill in
0.01	fill in	fill in	fill in

$v_{exact} = a \cdot (-t_r + \sqrt{t_r^2 + 2 \cdot d_{avail} / a})$ — provided; use it, derivation optional. One Python line, YOUR t_r .

Count iterations with one added line in the loop. Smaller steps → closer answers → more work — resolution vs effort; the theory arrives in Chapter 3.

They won't match exactly — the true limit sits between two tested speeds; the search reports the last safe step below it. The algorithm is not wrong: it is exact to its resolution.



TALK TO YOUR AI TEAMMATE “Explain in plain language why a smaller step gives a closer answer but more iterations.” Grade the explanation: accurate or not?



ENGINEER DECIDES in lecture, the same search with $t_r = 1.0$ s gave 20.0 m/s against the analytical 20.37 — the exact question Chapter 3 opens from.

Vocabulary today: step size, resolution. Tolerance and convergence get their theory later — no error theory now.

Problem



Model



Compute



Analyze



Validate



Improve



The road turns wet

1

New condition: $a = 7.0 \rightarrow 3.5 \text{ m/s}^2$

Braking capability halved. Same road length, same driver, same reaction time.

2

BEFORE rerunning — revisit P3

Less than half, exactly half, or more than half? Commit again — then run.

3

One edited value — the decomposition dividend

Because a is a parameter, the wet-road study is a single change, not a rewrite. You just ran your first parameter study.

EXPLAIN IT PHYSICALLY

Braking distance doubled — but reaction distance did not change. So v_{max} falls by LESS than half. Does your output agree? Can every teammate explain why, without the algebra?

Record both scenarios side by side: dry v_{max} , wet v_{max} , and the physical explanation of the difference.



TALK TO YOUR AI TEAMMATE “Here are our dry and wet results — log them side by side. Do not explain the ratio; we will.”



ENGINEER DECIDES if your P3 prediction said “exactly half,” the miss is the lesson. Own it.

Problem ›

Model ›

Compute ›

Analyze ›

Validate ›

Improve



Make — and defend — the engineering decision

“Under the stated assumptions ($t_r = \underline{\hspace{1cm}}$ s from our ruler measurement, $a_{\text{dry}} = 7.0 \text{ m/s}^2$, $a_{\text{wet}} = 3.5 \text{ m/s}^2$, $d_{\text{avail}} = 50 \text{ m}$, boundary rule: $\underline{\hspace{1cm}}$), the maximum safe approach speed is $\underline{\hspace{1cm}}$ m/s ($\approx \underline{\hspace{1cm}}$ km/h) for the dry case and $\underline{\hspace{1cm}}$ m/s ($\approx \underline{\hspace{1cm}}$ km/h) for the wet case.”

YOUR DEFENSE — one line of evidence on each rule:

The four lines: ① hand-checked known case ② Excel $\leftrightarrow$ Python agreement ③ analytical limit + step-size gap ④ physical reasonableness + stated limitations.

An engineering answer is a NUMBER + ASSUMPTIONS + EVIDENCE + JUDGMENT. A bare number is a guess wearing a lab coat.

Problem



Model



Compute



Analyze



Validate



Improve



Leave with the record complete — the report follows



EVIDENCE OF DOING

Trial table — every member
Hand-checked conversion
Representative t_r + rationale
Predictions P1–P3 — unedited
Excel table + recorded bracket



EVIDENCE OF THINKING

Pseudocode + AI verdicts
Code · tests · attacks · bugs
Validation ×4 + step table



DECISIONS RECORDED

Boundary rule + reason
Representative-t_r choice
Dry & wet, side by side
Decision statement complete



BEFORE YOU LEAVE

Missing report items listed
Assembly assigned — who, by when
Final review happens at submission
Stations reset — ruler, visor, chair
CHECKPOINT ④ — TA sign-off

Final formatting may be finished after lab — a complete, honest record tonight beats a polished report missing evidence.

Problem



Model



Compute



Analyze



Validate



Improve



Final report · human review & approval

“Before submission, we reviewed the completed report against our original engineering record — measurements, calculations, observations, code, plots, tests, validation evidence, and conclusions. We accept responsibility for the submitted engineering work.”

Human reviewed and approved.

EVERY TEAM MEMBER SIGNS — BEFORE
SUBMISSION (digital signature: create
once, copy-paste)

The recurring question, now literal: ***Would you confidently put your name on this result?***

AI assists. Humans observe. Humans verify. Humans decide. Humans approve.

Problem ›

Model ›

Compute ›

Analyze ›

Validate ›

Improve



One concise report — built from your record



FRAME IT

- 1 Problem / objective
- 2 Models — t_r & d_{stop}
- 3 Assumptions
- 4 Experimental setup + photo
- 5 Reaction-time data



GROUND IT

- 6 Hand check / unit check
- 7 Inputs + boundary rule
- 8 Predictions P1–P3



BUILD IT

- 9 Excel search + bracket
- 10–11 Pseudocode → AI critique
- 12 Python — functions + WHILE
- 13 Test battery — exp. vs act.



OWN IT

- 14 BREAK THE AI + debugging
- 15 Validation $\times 4$ + step size
- 16 Wet-road parameter study
- 17–18 Decision + limitations
- 19 Human review & approval — signed

A few honest sentences per section. Tables, plots, and photos carry the evidence — prose carries the judgment.

Problem



Model



Compute



Analyze



Validate



Improve

WHAT JUST HAPPENED

**You measured yourself with a model you validated,
turned the measurement into an algorithm you own,
coded it with an AI companion — on your terms,
tested it, broke it, validated it — and signed the decision
it produced.**

MODEL → INSTRUMENT → ALGORITHM → DECISION. You searched by stepping until a condition flipped. The lecture reference case ($t_r = 1.0$ s) gave search = 20.0 m/s against the analytical 20.37 — Chapter 3 begins from that disagreement: step size, resolution, controlled numerical error. Your own step-size table showed the same phenomenon at a different speed.

Problem



Model



Compute



Analyze



Validate



Improve

FUNDAMENTALS OF AI-AUGMENTED ENGINEERING

CHAPTER 2 / LAB-2

Engineering Report Rubric

Reaction Time & Safe Stopping Distance — Main LAB-2 Team Report

Suggested Assessment · 40 points

Used by the instructor and, where applicable, by an AI grading agent. Assessment rewards experimental evidence, algorithm ownership, testing, validation, and human responsibility.

Criterion	What full credit looks like	Pts
Problem, models, assumptions & inputs	Clear objective; reaction-time and stopping-distance models stated; assumptions include the ruler-reaction-time modeling choice; inputs (t_r, a, d_avail, step) carry units; boundary rule recorded.	4
Reaction-time experiment & data	Standardized procedure followed; complete trial table for every member including the invalid tally; one hand-checked cm → m conversion; representative t_r recorded with its rationale.	6
Committed predictions	P1–P3 recorded before any computation and left unedited; the wet-road prediction is revisited and re-committed before the rerun.	3
Excel search & bracket	Constants live in referenced cells; one row hand-checked; the SAFE/UNSAFE flip found; the bracket refined and recorded.	4

Continued

RUBRIC for the Lab Report

Pseudocode & AI critique	Complete pseudocode (inputs with units, formula, decision, WHILE, explicit stopping criterion, boundary call) written before AI; at least one AI suggestion evaluated ACCEPT / MODIFY / REJECT with reasoning; unauthorized assumption changes caught.	6
Python implementation & tests	Three functions with the lecture signatures; the WHILE search runs and matches the Excel bracket; the test battery is complete with expected values written before actual results.	6
BREAK THE AI & debugging evidence	At least three attacks run and documented; any bug traced evidence → hypothesis → fix → retest; “survived” findings verified, not staged.	4
Validation & wet-road study	Four checks performed and their independence discussed; step-size table completed and interpreted as resolution; wet-road result predicted first and explained physically.	4
Final decision, limitations & signed approval	Decision statement complete with assumptions and cited evidence; limitations stated (incl. ruler vs driver reaction time); report concise; human review signed by every member.	3
TOTAL		40

Problem

>

Model

>

Compute

>

Analyze

>

Validate

>

Improve

CHAPTER - 2

Lab 2 · ≈ 25 MIN

Lab Extension Challenge:

(Individual Assignment)

Students can do it at their convenience (individually) but remind them it is part of the Lab-2 assessment.

Problem



Model



Compute



Analyze



Validate



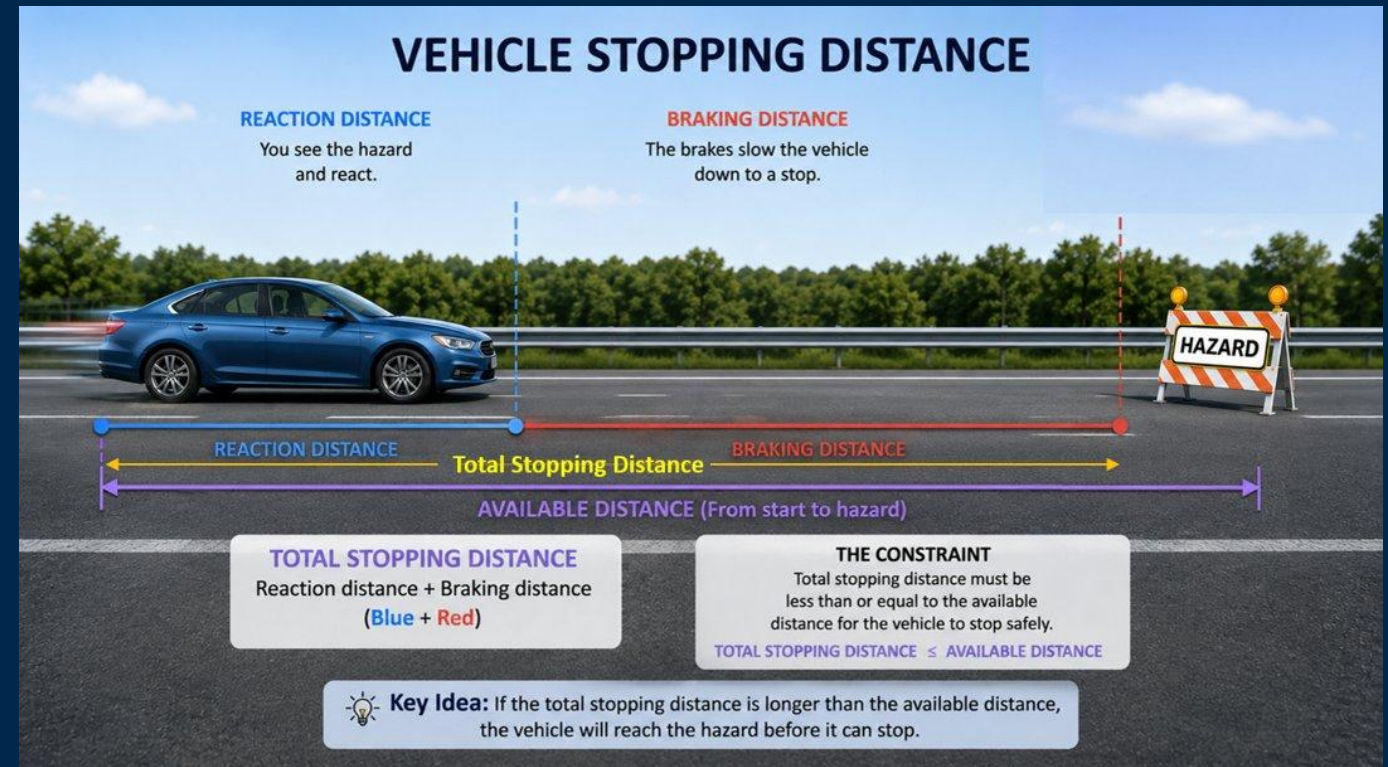
Improve



How Sensitive Is Safe Speed to Reaction Time?

Does a 20% longer reaction time cost 20% of the safe speed? Predict first — then compute.

An extension of the Lab-2 stopping-distance study. You already built the functions, the search, and the validation method — now one parameter puts your intuition to the test.



Problem

>

Model

>

Compute

>

Analyze

>

Validate

>

Improve



An assessment mini-challenge — not another lab report

1

Designed as part of the Lab-2 assessment

The rubric is on the last slide of this section.

2

During lab or on your own

Complete the analysis in the laboratory period or outside it — your choice. Your Lab-2 functions and this deck are all you need.

3

Submit a separate short report

Download and use Chp2-Lab2-LabExtension-Challenge-ReactionTimeSensitivity-Report-Template.docx

THE ENGINEERING QUESTION

If reaction time increases by 20%, does maximum safe speed decrease by 20% — and what does the answer tell us about the model?

Same workflow as the whole lab:

**Predict → Compute → Plot → Explain →
Ask AI → Critique → Validate → Decide**



ENGINEER DECIDES
decide, and approve.

the question is not how many cases you ran — it is what the trend means and whether you can defend it. AI assists. Humans observe, verify,

Problem

›

Model

›

Compute

›

Analyze

›

Validate

›

Improve



Commit to a prediction

1

The 20% question

If reaction time increases by 20%, does maximum safe speed decrease by 20%, less than 20%, or more than 20%? Circle one.

2

Sketch the shape

Will $v_{\max}$ vs t_r be a straight line, or curved? Which way?

3

Why?

Briefly explain the physical reasoning behind your prediction.

4

Where does YOUR t_r sit?

Against the 1.00 s lecture driving-response reference — will your ruler-measured t_r give a higher or lower $v_{\max}$?

THE MODEL

$$d_{\text{avail}} = v \cdot t_r + \frac{v^2}{(2a)}$$

$$a = 7.0 \text{ m/s}^2 \cdot d_{\text{avail}} = 50 \text{ m} \text{ — held fixed}$$

Look closely: t_r multiplies v in one term only.

So what — exactly — should a longer reaction time change?

✓ **ENGINEER DECIDES** *record your prediction NOW — no revising after you compute (correctness is not graded).*

Problem

>

Model

>

Compute

>

Analyze

>

Validate

>

Improve



Five reaction times against one model

Case	t _r (s)	v _{max} (m/s)	v _{max} (km/h)	Δ vs previous case
Your ruler t _r *	—			
Simple-reaction reference	0.20			
Intermediate response case	0.50			
Lecture driving-response reference	1.00			
Extended-response case	1.50			

HELD FIXED

$$a = 7.0 \text{ m/s}^2$$
$$d_{\text{avail}} = 50 \text{ m}$$

step $\leq 0.1 \text{ m/s}$ · your rule
*t_r ≈ 0.20 already?
...use 0.25 instead

THEN — ONE PICTURE

Plot v_{max} vs t_r — Excel or Python, your choice. Five points are enough to show the shape. Before reading on: does your committed prediction survive contact with the plot?

✓ **ENGINEER DECIDES** *reuse your Lab-2 functions — each case is one parameter change. If you are rewriting code, stop and ask why.*

Problem

>

Model

>

Compute

>

Analyze

>

Validate

>

Improve



Interrogate the pattern — don't assume one

THE 20% CHECK

Add one run: $t_r = 1.20$ s. From 1.00 $\rightarrow$ 1.20 s (+20%), what happened to $v_{\max}$ — in percent?

Compute the actual percentage — write it down next to your prediction.

THE TREND

As t_r grows, $v_{\max}$ falls — but how? Equal amounts per step, or changing amounts?

Read it off your five points — the plot, not your expectation.

WHY NOT PROPORTIONAL?

Reaction distance $v \cdot t_r$ scales with t_r — braking distance $v^2/(2a)$ doesn't care.

Explain the sub-proportional drop from the two terms.

THE RULER ASSUMPTION

Your ruler t_r vs the 1.00 s lecture driving-response reference — how much $v_{\max}$ separates them?

What does that gap say about using a ruler catch to model driving?

✓ **ENGINEER DECIDES** *use your computed evidence — not what you expected to see. The data decide what the pattern is.*

Problem

›

Model

›

Compute

›

Analyze

›

Validate

›

Improve



Ask AI for the explanation — then grade it

SUGGESTED PROMPT

“In the model $d_{stop} = v \cdot t_r + v^2/(2a)$ with a fixed available distance, maximum safe speed drops by less than the percentage increase in reaction time. Explain why, in plain engineering language, using the two distance terms.”

Any capable AI assistant (your coding companion). Record a concise summary of its explanation. Do NOT copy the full answer.

NOW CRITIQUE THE AI RESPONSE

- 1 Is the explanation consistent with the two-term model?
- 2 Does it match YOUR computed numbers — check one case against it?
- 3 Did it introduce any assumption you did not state (brake lag, different a , changed boundary)?
- 4 Verdict: ACCEPT · MODIFY · REJECT — with a reason.
- 5 What one additional computation would test its claim?

 **THE KEY DISTINCTION** *a fluent explanation is not necessarily a correct explanation. Your five computed points outrank its prose.*

 **ENGINEER DECIDES** *AI explains; engineers verify the explanation against computed evidence.*

Problem

›

Model

›

Compute

›

Analyze

›

Validate

›

Improve



Validate one case — then make the call

1 INDEPENDENT CHECK

Validate at least one case without the search:

$$v_{\text{exact}} = a \cdot (-t_r + \sqrt{t_r^2 + 2 \cdot d_{\text{avail}} / a}).$$

One line of Python or a calculator. Does it agree with your search to within the step size? If not — which do you trust, and why?

2 ENGINEERING JUDGMENT

State a defensible conclusion: how sensitive is safe speed to reaction time — and is the relationship proportional?

What does the gap between your ruler t_r and the longer driver-response reference mean for how these results should be used?

You are NOT expected to build a better driver model — identifying the assumption and its consequence is the deliverable.

Same functions → new parameter → new insight — that is what decomposition buys

Problem

›

Model

›

Compute

›

Analyze

›

Validate

›

Improve



One compact report — 10 points

LAB EXTENSION CHALLENGE REPORT

separate from your main Lab-2 report · ½–1 page

- 1 Committed prediction — the 20% question + shape
- 2 Model / given conditions (a and d_{avail} fixed)
- 3 Completed five-case results table
- 4 v_{max} vs t_r plot
- 5 The 20% check + trend interpretation
- 6 AI explanation + your **ACCEPT / MODIFY / REJECT**
- 7 Independent analytical check of one case
- 8 Final judgment — incl. the ruler-t_r assumption

No formal Intro / Methods / Results / Discussion structure required.

Recommended Challenge Rubric	Points
Prediction and physical reasoning	1
Computation / five-case results table	3
Trend interpretation / the 20% check	2
AI use AND critical evaluation	2
Independent check / validation	1
Final engineering judgment	1
TOTAL :	10

Serious enough to take seriously — small enough that Lab-2 still belongs to the main lab.

Problem

>

Model

>

Compute

>

Analyze

>

Validate

>

Improve

RUBRIC for the Lab Extension Report

© 2026 AI First Engineering · Fundamentals of AI-Augmented Engineering · Chapter 2 / LAB-2

FUNDAMENTALS OF AI-AUGMENTED ENGINEERING

CHAPTER 2 / LAB-2

Lab Extension Challenge Rubric

How Sensitive Is Safe Speed to Reaction Time? · Individual Assignment

Suggested Assessment · 10 points

Separate from the main LAB-2 team report. One compact individual challenge report. Assessment rewards the committed prediction, computed evidence, critical evaluation of AI, and engineering judgment.

Criterion	What full credit looks like	Pts
Prediction and physical reasoning	The 20% question and the expected shape of v_{max} vs t_r are committed before computing, with brief defensible physical reasoning. Correctness of the prediction itself is not graded.	1
Computation / five-case results table	Five-case table is complete with reaction times, maximum safe speeds in m/s and km/h; values are traceable to the student's own Lab-2 functions with the stated fixed conditions.	3
Trend interpretation / the 20% check	The +20% reaction-time case is computed and compared against the prediction; the sub-proportional trend is explained from the two distance terms.	2
AI use AND critical evaluation	The AI explanation is summarized concisely and graded ACCEPT / MODIFY / REJECT against the model and the computed numbers, with a reason.	2
Independent check / validation	At least one case is validated with the analytical formula and reconciled with the search to within the step size.	1
Final engineering judgment	A defensible sensitivity conclusion is stated, including what the ruler- t_r assumption means for how the results may be used.	1
TOTAL		10

Problem



Model



Compute



Analyze



Validate



Improve