

FUNDAMENTALS OF AI-AUGMENTED ENGINEERING: A CONVERSATIONAL APPROACH

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

© 2026 AI First Engineering

Problem

›

Model

›

Compute

›

Analyze

›

Validate

›

Improve

The engineering workflow you will use throughout this book

by Taner & Sebnem Onsay,

CHAPTER - 5

PART 2 · ≈ 35 MIN

Newton in Action — Fast When It Works

Three steps by hand, then Excel, then Python. Two steps where bisection needed eleven — and a stopping rule for a method that has no walls.

2

Problem



Model



Compute



Analyze



Validate



Improve



Newton-Raphson: evaluate, read the slope, jump, repeat

newton — PSEUDOCODE

```

GET  $x_0$                 # a justified start
GET tol                # from the requirement (Ch. 3)
REPEAT
    SET shortfall =  $f(x_n)$ 
    SET slope     =  $f'(x_n)$ 
    SET correction = - shortfall / slope
    SET  $x_{n+1}$  =  $x_n$  + correction
    # follow the tangent to zero
UNTIL |correction| ≤ tol
    # the last move was too small to matter
CHECK  $f(x_{n+1}) \approx 0$  and  $x_{n+1}$  is physical
REPORT  $x_{n+1}$ 

```

CHAPTERS 2 AND 4, REASSEMBLED

REPEAT / UNTIL — a condition decides (your heater loop). $f(x_n)$ — your stopping-distance function, unchanged. tol — owned by the requirement, as always. Only $f'(x_n)$ is new: one line of calculus feeding the loop.

WHAT IS GENUINELY NEW

The state is **ONE** point that jumps — no pair, no walls. The **VALUE** of f decides the size and direction of every move. And the stop is the size of the last move, not a width you can point at.

Not one new statement — one new piece of information: the slope. Bisection asked f only “which side?” Newton asks f “how far, and how steep?”

Problem



Model



Compute



Analyze



Validate

3



Improve



Three Newton steps, by hand

n	x_n (m/s)	$f(x_n) = d_stop - 50$ (m)	$f'(x_n) = 1 + x_n/7$	correction $-f/f'$ (m/s)	x_{n+1} (m/s)
0	20.0	-1.4286	3.8571	+0.3704	20.3704
1	you	...	...	...	...
2	you	...	...	...	...

start: $x_0 = 20.0$ · $d_stop(x) = x + x^2/14$ · slope $1 + x/7$ · everything here is Chapter 2 arithmetic plus one derivative

HOW TO RUN A ROW

- 1 · $f(x_n) = d_stop(x_n) - 50$
- 2 · $f'(x_n) = 1 + x_n / 7$
- 3 · $correction = -f / f'$
- 4 · $x_{n+1} = x_n + correction$

The arithmetic is Chapter 2. The judgment — is the correction small enough to stop? — is Chapter 5. That judgment is the exam question.

PREDICT first: after your two rows, how big is the correction — bigger, the same, or much smaller than 0.37?

Deliberately AI-free. Three rows on paper now buy you the whole method — and the exam.

Problem



Model



Compute



Analyze



Validate



Improve



The correction after three steps: smaller than a hair

n	x_n (m/s)	$f(x_n)$ (m)	$f'(x_n)$	correction (m/s)	x_{n+1} (m/s)
0	20.0	-1.4286	3.8571	+0.3704	20.3704
1	20.3704	+0.0098	3.9101	-0.0025	20.36786
2	20.36786	+0.0000004	3.9097	-0.0000001	20.367864

report: 20.368 m/s · (the reference 20.367864 — Newton has it to six decimals after two steps)

WHAT TWO STEPS BOUGHT

20.367864 — the reference root to six decimals. Bisection from [20.0, 20.5] needed five halvings for ± 0.008 ; from [0, 40], eleven for ± 0.01 .

TWO KINDS OF EVIDENCE

Bisection's $\pm$ came from two walls you could point at. Newton's shrinking correction is evidence the iterates have SETTLED — not a bound. Chapter 3 said it first: change $\neq$ error.

Two model runs, two slopes — six decimals. Read the corrections: $0.37 \rightarrow 0.0025 \rightarrow 0.0000001$. Each step roughly DOUBLES the number of correct digits. That is why Newton is fast — near a root, on a smooth f .

Problem



Model

**Compute**

Analyze



Validate

5



Improve



Can Excel express the SAME Newton algorithm?

You just ran three steps by hand. Now make the algorithm visible before we automate it in Python.

n	x _n	d _{stop}	f(x _n)	f'(x _n)	-f / f'	x _(n+1)	corr	≤ tol?	
0	20.0	48.571	-1.429	3.857	+0.370	20.370	0.370	NO	
1	?	?	?	?	?	?	?	?	
2	?	?	?	?	?	?	?	?	

DESIGN THE LOGIC

Which cells are calculations?
Which cell contains the decision?
Which single value must be carried to the next row?

THE KEY QUESTION

Can the spreadsheet tell you when to STOP — without you deciding by eye?

ALGORITHM FIRST.

Excel is not the method. Python is not the method. The Newton logic is the method.

Problem



Model



Compute



Analyze



Validate



Improve



The spreadsheet version of the Newton step

VISIBLE COMPUTATION

SHORTFALL	$f = d_stop(x) - 50$
SLOPE	$f' = 1 + x / 7$
CORRECTION	$\Delta x = -f / f'$
NEXT x	$x_next = x + \Delta x$
STOP TEST	IF($ABS(\Delta x) \leq tol$, "STOP", "again")

WHAT EXCEL ADDS

- Every row preserves the evidence.
- The correction column shows convergence as it happens.
- The STOP decision is exposed — not hidden in code.
- Copying the row creates repetition.
- A slope typed wrong is visible in its own column.

HAND · EXCEL · PYTHON

SAME ALGORITHM, NEW PLATFORM.

The computation changes form; the engineering logic does not.

Problem



Model



Compute



Analyze



Validate



Improve



newton.py — the sprint, automated

Newton-Raphson

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

```

● ● ● newton.py — Chapter 2's stopping_distance(), unchanged

def f(v):
    return stopping_distance(v, 1.0, 7.0) - 50.0
def fprime(v):
    # the slope — differentiate d_stop
    return 1.0 + v / 7.0
v = 20.0
tol = 0.01
n = 0
while True:
    correction = -f(v) / fprime(v) # shortfall / sensitivity
    v = v + correction
    n = n + 1
    print(n, v, f(v), correction)
    if abs(correction) <= tol:
        # the last move was tiny
        break
    assert n < 50, 'no convergence' # a guard, never a proof
print(n, 'steps:', v, ' f =', f(v))
2 steps: 20.367864481531598 f = 4.485341946747212e-07

```

READ IT LIKE AN ENGINEER

fprime is Part 1 in code — one line of calculus. The loop stops when the **CORRECTION** is smaller than tol, never on a count. The assert is a guard against a runaway loop — it refuses to spin forever; it certifies nothing. And v is the only state: no pair, no walls.

INSPECT THE OUTPUT

Two steps; f at the answer is 4×10^{-7} m. The requirement asked for ± 0.01 ; the evidence supports 20.3679. Report either — but SAY which, and why.

PREDICT → RUN → INSPECT — as always. The output shown is the real output of this exact program.

Problem



Model



Compute



Analyze



Validate

8



Improve



When is the correction small enough?

THE STOPPING RULE, WRITTEN DOWN

stop when $|x_{n+1} - x_n| \leq \text{tol}$ and then CHECK $|f(x_{n+1})|$ small in engineering units, x_{n+1} physical

step n	0	1	2	3
$ \text{correction} $ (m/s) from $x_0 = 20.0$	0.37	0.0025	0.0000001	0

correct digits roughly double each step — near the root, on a smooth curve

CHAPTER 3's RULE, QUALIFIED

Stop when the evidence meets the requirement — Chapter 3's rule. But Newton's evidence is a shrinking CHANGE, not a bound: it estimates the remaining movement, it does not prove it.

THE REQUIREMENT STILL CHOOSES tol

Need ± 0.01 m/s? Stop when the move is below 0.01 — then confirm $f(x^*) \approx 0$ and x^* is physical. Never a fixed count. Never “residual 10^{-12} so it must be exact.”

Newton earns digits fast — and cannot certify them. Convergence evidence first, validation always: $f(x^*) \approx 0$, x^* inside the physical world, and the model still needs checking.

Chapter 4 could schedule its accuracy before running. Chapter 5 cannot — it can only watch the corrections collapse and then verify.

Problem



Model



Compute



Analyze



Validate

9



Improve



PREDICT

**No head start this time: $v_0 = 0$ — a parked car.
And $v_0 = 40$ m/s — a car that cannot stop.**

Bisection from $[0, 40]$ needed 11 halvings for ± 0.01 . Starting Newton from EITHER end — with no bracket, only a start — how many steps to a correction below 0.01? Commit: 2? 6? 20? 200?

Newton has no walls to lean on. Whatever it does from 0 and from 40, it does because of the SHAPE of f — write down what you expect, then we run both.

Problem



Model



Compute



Analyze



Validate

10



Improve



Eleven halvings — or two steps

THE HUNT · CHAPTER 4

11

halvings from $[0, 40]$ — guaranteed
 ± 0.0098



THE SPRINT · TODAY

2

steps from $20.0 \cdot 5$ from $0 \cdot 4$ from 40
— evidence, not a bound

THE PRICE OF INFORMATION

Every Newton step costs one model run PLUS one slope. Two steps = 2 runs + 2 slopes. Eleven halvings = 11 runs + 2 endpoint checks. Count both columns honestly.

required accuracy	bisection halvings $[0, 40]$	Newton steps from 20.0
± 0.01	11	2
± 0.001	15	3
± 0.0001	18	3

READ THE SCALING — THE REAL LESSON

Each extra decimal costs bisection +3 or 4 halvings. It costs Newton almost nothing — near the root each step roughly doubles the correct digits. But bisection's column is a **GUARANTEE**; Newton's is an **OBSERVATION**.

Newton counted to $|\text{correction}| \leq \text{tol}$; from $v_0 = 0$: $5 \cdot 6 \cdot 6$ — the start matters more than the tolerance

The sprint buys accuracy with information — the slope — not with effort. (These counts belong to THIS f and THESE starts. What generalizes: digits double per step near a root — and no bound comes with them.)

Problem



Model



Compute



Analyze



Validate

11



Improve



What each method asked of you — and what it gave back

BISECTION ASKED FOR

- **A certified bracket**
- **Two signs**
- **Continuity**

Never how big f was, na crude cage from physics was enough. It gave back CONTAINMENT: root $\pm$ half-width, on a schedule known before running. Slow and sure — by design. ever a slope, never a good guess —

NEWTON ASKED FOR

- **one start**
- **f AND f' at every step**

A justified start, the derivative of your model, a domain check after every jump. It gave back SPEED: six decimals in two steps — no walls, no bound, only a collapsing correction. Fast when it works — Part 3 shows when it does not.

AND BOTH ASKED FOR THE MODEL

Same f , same 50 m of road, same t_r and a . The method changed; the physics did not — and neither method can tell you whether the physics is right. That is still validation's job.

Speed was bought with information. The slope told Newton where to jump. Part 3 asks what happens when the slope lies.

Problem



Model



Compute

**Analyze**

Validate



Improve



AI proposes the start, the slope, the method — you attack all three



AI MAY...

differentiate your model — and show the derivative
implement the Newton loop you specified — with its guard
check your hand steps against its own
propose a starting point — with a reason
run bisection and Newton side by side and tabulate the cost
construct a case where Newton fails — for you to diagnose



THE ENGINEER MUST...

own f — and check f' at one point you can compute
justify the start from the physics, not from habit
set tol from the requirement — and refuse a fixed count
confirm every iterate stayed inside the physical domain
validate the root: $f(x^*) \approx 0$, the known case, the physics
choose the method — and defend the choice, and the decision

The analyst becomes a critic you argue with. Ask it to break your run before the lab does. AI assists. Human verifies. Engineer decides.

Problem ›

Model ›

Compute ›

Analyze ›

Validate ›

Improve