

FUNDAMENTALS OF AI-AUGMENTED ENGINEERING: A CONVERSATIONAL APPROACH

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

© 2026 AI First Engineering



The engineering workflow you will use throughout this book

CHAPTER - 6

PART 4 · ≈ 33 MIN

Automate It, Guard It

How far can a straight line be trusted? What changes when the request leaves the measured range? Then Python does the four estimates at once — and checks itself — and you debug a program that runs and is wrong.

4

Problem

›

Model

›

Compute

›

Analyze

›

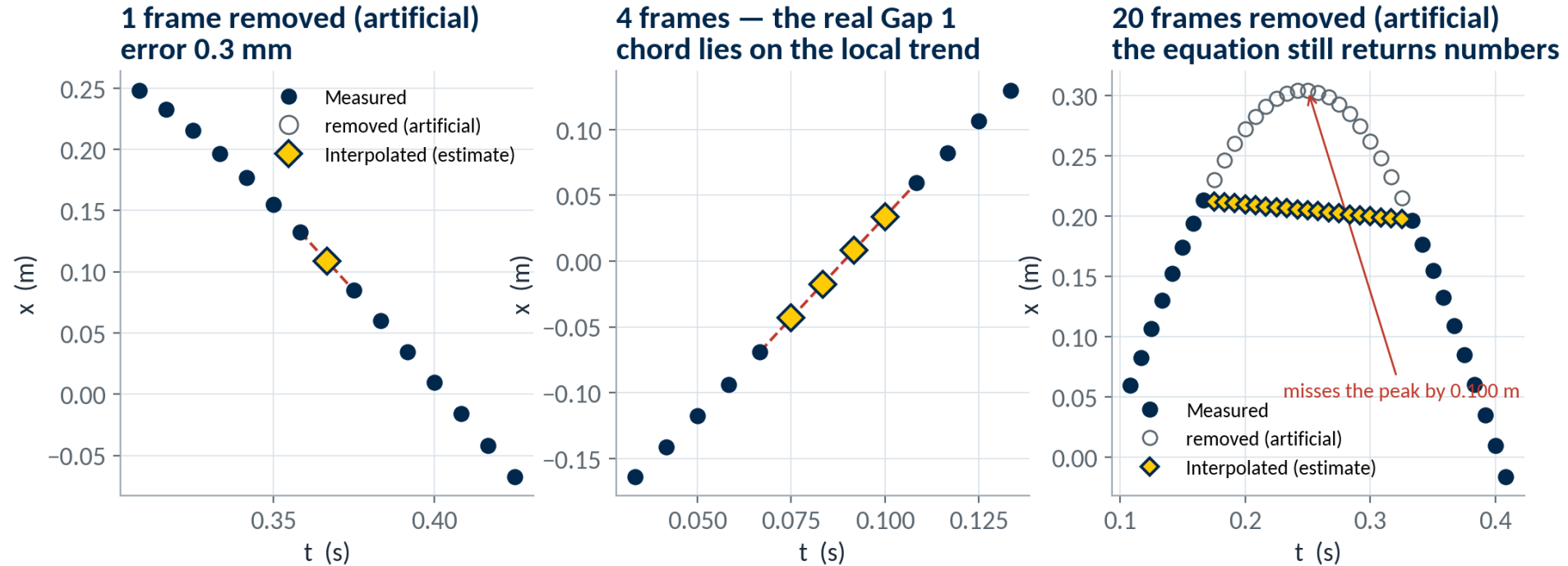
Validate

›

Improve



How far can we trust the straight line?



ONE frame removed (artificial) from a smooth stretch: the chord misses by about 0.3 mm — below the scatter of the measurements.

FOUR frames — the real Gap 1: the chord lies on the local trend of x . Accepted in Part 3 with evidence.

TWENTY frames removed (artificial) across the first peak: the chord misses the peak by ≈ 0.100 m — about 33 %. The equation did not fail; the assumption did.

Problem



Model



Compute

**Analyze**

Validate



Improve



More digits do not create knowledge

$x(0.100 \text{ s}) =$

0.033894688 m

the authentic result, printed with every digit Python offers

ASK

Does displaying nine decimals make this estimate more certain than -0.0432 m ?

Where did the certainty of the last six digits come from — the measurements, or the calculator?

COMPUTATIONAL PRECISION

How many digits the arithmetic produces. Unlimited, and free. The straight-line formula will happily give twelve.

ENGINEERING CONFIDENCE

How many digits the EVIDENCE supports: local behaviour, size of the gap, scatter of the measurements. For Gap 1, x is known to about a millimetre — three decimals, not nine.

Chapter 3 asked which digits you would sign your name to. The same question, now for an estimate: report the digits the evidence supports, and label the number as an estimate.

Problem



Model



Compute



Analyze



Validate



Improve



ACCEPT / MODIFY / REJECT — judge each case

case	local behaviour	missing interval vs the system's time scale	your verdict + evidence
A · x, one frame, smooth stretch	nearly straight	1/120 s — tiny fraction of a revolution	
B · x, Gap 1 (four frames)	straightest part of the curve (zero crossing)	1/30 s $\approx$ 6 % of a revolution	
C · y, Gap 1 (four frames)	bottom of the arc (extremum)	1/30 s $\approx$ 6 % of a revolution	
D · x, twenty frames across a peak	the curve turns a corner inside the gap	1/6 s $\approx$ 26 % of a revolution	
E · x, four frames, plus the r(t) check passed	straightest part; independent physics agrees	$\approx$ 6 % of a revolution	

THE RESPONSE FORMAT

DECISION — one word. EVIDENCE — what you looked at: the neighbours, the local shape, the fraction of a cycle, any independent check. No confidence scores. No frame-count rule.

TWO CASES SHARE EVERY NUMBER

B and C have the same experiment, the same frames, the same formula and the same sampling rate. If your verdicts differ, you have understood the chapter. If they agree, look again.

Expected: A ACCEPT · B ACCEPT · C MODIFY · D REJECT (or MODIFY with a better local model / more data) · E ACCEPT — with the evidence stated.

Problem



Model



Compute

**Analyze**

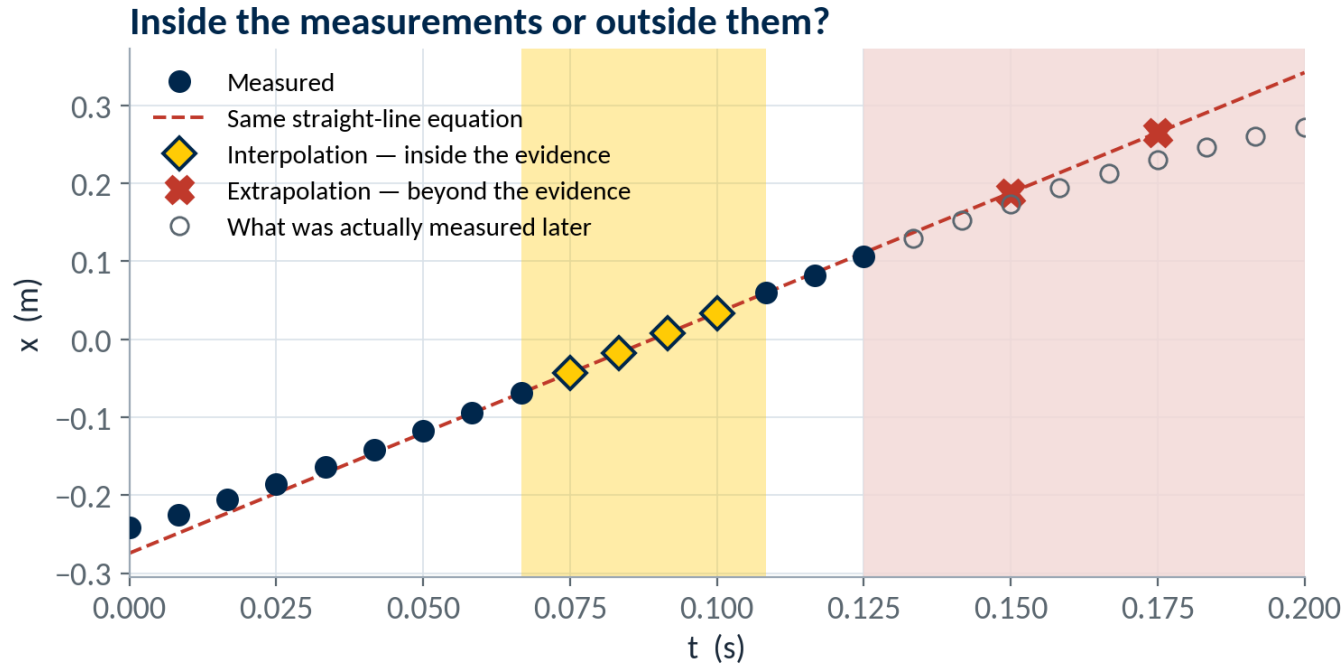
Validate



Improve



Interpolation vs extrapolation — same equation, different claim



INTERPOLATION

A requested time BETWEEN two measurements. The estimate is fenced in by evidence on both sides.

EXTRAPOLATION

A requested time BEYOND the last measurement. The same straight-line equation still computes — and now assumes the trend continues where nothing was measured. A stronger claim.

The formula cannot tell the difference. You can — by checking where t lies. That check becomes the first line of the Python program in five minutes.

On the plot: the red crosses are what the Gap 1 line predicts at $t = 0.150$ s and 0.175 s; the open circles are what Tracker actually measured there. The wheel had already turned.

Problem



Model



Compute

**Analyze**

Validate



Improve



TRANSITION

Four missing values were manageable in Excel. What about many gaps — or twenty thousand rows?



The full Tracker record already has three gaps. A longer video, or a hundred videos, would have hundreds. Copying a formula by hand stops being a method.

Automation earns its place only after the method is understood. You understand this one: you did it on paper and in a spreadsheet, and you know when it breaks.





Algorithm before code — build the logic together

INPUT	the two bounding measurements (t_1, x_1) , (t_2, x_2) and the list of missing times
FIND	the slope from the bounding pair — the local straight-line model
CHECK	is the requested time inside $[t_1, t_2]$? If not, this would be extrapolation — warn, skip
CALCULATE	$x = x_1 + \text{slope} \cdot (t - t_1)$
CHECK	does the estimate lie between x_1 and x_2 ? For a straight line it must
LABEL	print the value with the word INTERPOLATED — never as a bare number
VALIDATE	compare the first value with the hand / Excel result before trusting the loop

WHAT REPEATS?

Only the part that depends on t . The slope is computed once; the loop runs CHECK → CALCULATE → CHECK → LABEL once per missing time.

WHAT IS NEW?

Nothing in the mathematics. The two CHECK steps are new — and they are the difference between a calculator and engineering code.





interpolate_gap1.py — four estimates, one loop

```
●●● interpolate_gap1.py — the whole program

# bounding measurements of Gap 1 (raw values)
t1 = 0.06666667;    x1 = -0.06883248
t2 = 0.1083333;    x2 = 0.05957638
missing_times = [0.075000, 0.083333, 0.091667, 0.100000]

slope = (x2 - x1) / (t2 - t1)    # local model, once
for t in missing_times:
    x = x1 + slope * (t - t1)    # the equation
    print(t, round(x, 5), "INTERPOLATED")

0.075 -0.04315 INTERPOLATED
0.083333 -0.01747 INTERPOLATED
0.091667 0.00821 INTERPOLATED
0.1 0.03389 INTERPOLATED
```

1 · PREDICT

Before running: what four numbers will appear, to two decimals? You have them on paper and in column B. Write them down.

2 · RUN

The printed output is the real output of this exact program.

3 · INSPECT

HAND ↔ EXCEL ↔ PYTHON: -0.04315, -0.01747, +0.00821, +0.03389. Three platforms, one algorithm, one answer.

Problem ›

Model ›

Compute ›

Analyze ›

Validate ›

Improve



Two guardrails make it engineering code

```
●●● interpolate_gap1.py — with guardrails

slope = (x2 - x1) / (t2 - t1)
for t in missing_times:
    if t < t1 or t > t2:                # guardrail 1
        print(t, "WARNING: outside measured interval")
        continue
    x = x1 + slope * (t - t1)
    if x < min(x1,x2) or x > max(x1,x2): # guardrail 2
        print(t, "WARNING: outside bounding values")
    else:
        print(t, round(x, 5), "INTERPOLATED")

# try it: missing_times = [0.075, 0.150]
0.075 -0.04315 INTERPOLATED
0.15 WARNING: outside measured interval
```

GUARDRAIL 1 · WHERE IS t ?

Inside $[t_1, t_2]$ — interpolation. Outside — extrapolation, a stronger claim the program refuses to make silently. The formula would have returned +0.188 m for $t = 0.150$ s without complaint.

GUARDRAIL 2 · WHERE IS x ?

A straight line between two points cannot leave the interval between their values. If the estimate does, the arithmetic is wrong — no physics needed.

THE MODEL IS YOURS, NOT THE TOOL'S — so the checks on it are yours to write. A guard is not a fix; it is a question asked automatically.

Problem ›

Model ›

Compute ›

Analyze ›

Validate ›

Improve



Debug a program that runs — and is wrong

```
●●● bug_1.py — runs, prints plausible numbers
```

```
slope = (x2 - x1) / (t2 + t1)      # ← ?  
x = x1 + slope * (t - t1)  
0.075 -0.06272 INTERPOLATED  
0.083333 -0.0566 INTERPOLATED  
0.1 -0.04437 INTERPOLATED
```

BUG 1 · CAUGHT BY A KNOWN CASE

Every number is inside $[x_1, x_2]$; every number is negative and increasing. Nothing looks absurd. Only the KNOWN CASE catches it: the hand / Excel value at $t = 0.075$ is -0.04315 , not -0.06272 . Then localise: the slope is 0.7338 m/s, not 3.0818 — the denominator is a sum.

```
●●● bug_2.py — runs, guardrail 2 fires
```

```
slope = (x2 - x1) / (t2 - t1)  
x = x1 + slope * t                # ← ?  
0.075 WARNING: outside bounding values  
0.083333 WARNING: outside bounding values  
# x would have been 0.16230, 0.18798, ...
```

BUG 2 · CAUGHT BY GUARDRAIL 2

The missing $(t - t_1)$ makes every estimate $+0.16$ m or more — outside the bounding values — so the guard refuses before anyone reads a plausible-looking table. Without the guard this bug also prints numbers.

Problem ›

Model ›

Compute ›

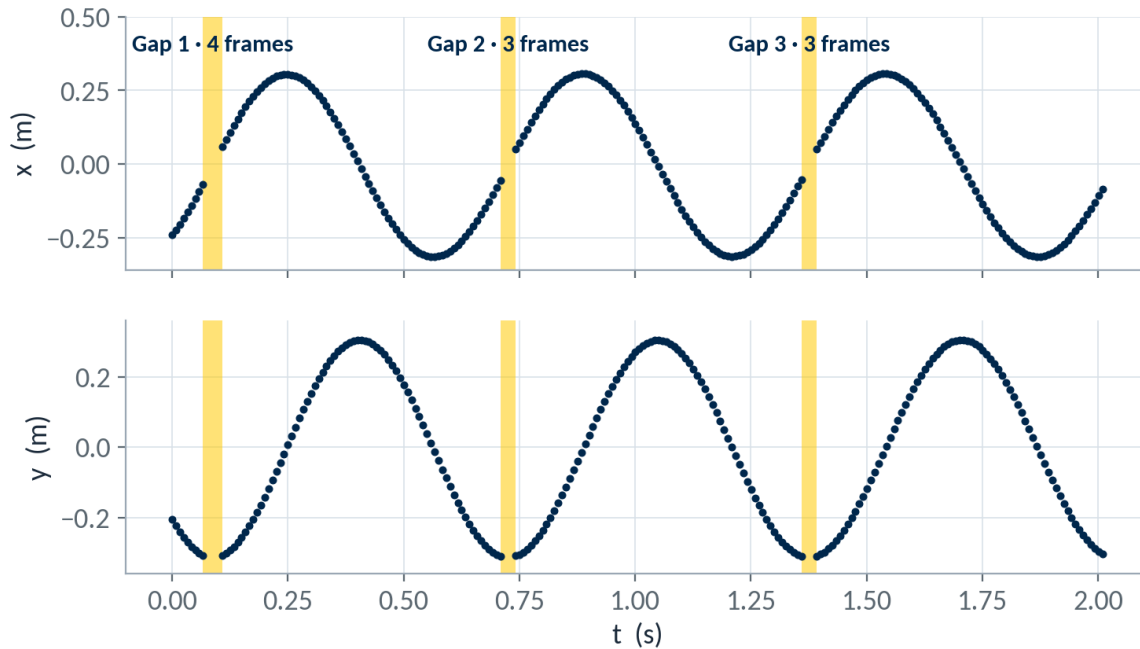
Analyze ›

Validate ›

Improve



Scaling the idea to the full dataset — conceptually



- 1 READ the record
- 2 inspect Δt between rows
- 3 detect a gap
- 4 list the missing times
- 5 estimate (interpolate)
- 6 LABEL every estimate**
- 7 validate (r check)
- 8 report

A REAL-DATA NOTE

Real experimental timestamps are not perfectly evenly spaced. Robust gap detection uses a tolerance (" Δt noticeably larger than $1/120$ s"), not exact equality. Chapter 3's tolerance idea, back again.

Automate a method only after you understand what it assumes. The loop above assumes a straight line across every gap it finds — you now know when that is, and is not, defensible.

Problem



Model



Compute

**Analyze**

Validate



Improve