Journal of Interdisciplinary Engineering, Computing and Automation
Vol. 01, Issue 01, July–December 2026, pp. 13-24

Inference Latency Is a Phase Margin Problem: Closed-Loop Evaluation of Learned Estimators in Feedback Control

Dr. Ramakrishna Kumar

Deputy Head / Assistant Professor, Electrical and Communication Engineering, College of Engineering, National University of Science and Technology, Muscat, Oman.

Email: ramakrishnakumar@nu.edu.om

ABSTRACT

Learned estimators are increasingly placed inside feedback loops to denoise or reconstruct sensor signals. Machine-learning practice evaluates such a component on estimation error measured offline. Control practice recognises that any element in the loop contributes phase lag. These two evaluations reward opposite things, because a more capable model generally takes longer to evaluate. We simulated a second-order positioning stage (natural frequency 5.00 rad/s, damping ratio 0.240) under PID control at a 2.0 ms sample period, with a feedback-path estimator characterised by a noise reduction factor and an inference latency. We computed the analytical delay budget from the loop phase margin, then swept latency and accuracy independently, measuring tracking error, overshoot, steady-state jitter and actuator chatter. The delay-free loop had a crossover frequency of 19.2 rad/s and a phase margin of 49.6 degrees, giving a delay budget of 45.1 ms, or 22.5 control samples. Relative to a raw sensor, a zero-latency ideal estimator reduced steady-state jitter from 0.00674 to 0.00200 and actuator chatter from 44.1 to 8.2. When accuracy was purchased with latency, benefits saturated while costs accelerated: a 10-sample estimator cutting noise by 80 per cent reduced chatter to 11.3 but raised overshoot from 10.1 to 32.2 per cent, and a 20-sample estimator cutting noise by 90 per cent raised steady-state jitter to 0.4593, 68 times worse than the raw sensor it replaced. Across the accuracy-latency grid, tracking error varied by a factor of 22 along the latency axis and by less than 2 per cent along the accuracy axis. Closed-loop performance was governed almost entirely by latency and almost not at all by estimation accuracy. Estimators intended for feedback should be specified by a delay budget derived from the loop, and evaluated in closed loop rather than offline.

Keywords: control systems; machine learning; inference latency; phase margin; hardware-software co-design; intelligent systems.

1. INTRODUCTION

Learned components are entering control loops. Neural estimators reconstruct states that are expensive to measure, learned filters denoise signals from cheap sensors, and vision models supply feedback that no conventional transducer provides [1-3]. The engineering appeal is clear: a model trained on data can extract information that a hand-designed filter cannot.

These components are developed and selected using the evaluation practice of machine learning. A candidate model is judged on estimation error against held-out data, and the

selection rule is that lower error is better. Capacity is added until the error stops improving or the budget runs out.

A control engineer looks at the same component differently. Anything inserted into a feedback path takes time, and time in a loop is phase lag. A pure delay contributes phase proportional to frequency, so at the crossover frequency where the loop gain passes unity it consumes phase margin directly. When the margin is gone the loop oscillates, and no property of the estimator's accuracy prevents this.

The two views reward opposite things, because within a model family accuracy is generally bought with computation, and computation is time [19-21]. A model selection procedure that optimises offline error will, all else equal, select the slowest acceptable model. A loop stability requirement selects the fastest. The engineering question is where between those the design should sit, and that question is not answerable inside either discipline alone.

The interaction is recognised in principle. Delay margin is standard control theory, and the real-time systems literature has long treated latency as a first-class constraint [4-6,22]. What is less often done is to measure, for a specific loop, how the two effects compare in magnitude — whether a realistic accuracy gain can offset a realistic latency cost, or whether one dominates. That is what we set out to quantify.

1.1 Contributions

We quantify closed-loop performance of a PID-controlled second-order plant as a joint function of estimator noise reduction and inference latency; we compare the empirical instability point against the analytical delay budget derived from the loop's phase margin; we show that the two axes contribute to closed-loop error by very different magnitudes; and we give a design rule that expresses the admissible inference latency as a property of the loop rather than of the model.

2. SYSTEM AND METHOD

2.1 Plant and Controller

The plant is a second-order positioning stage with mass, damping and stiffness giving a natural frequency of 5.00 rad/s and a damping ratio of 0.240 [4,25]:

$$G(s)=\frac{1}{m s^2+cs+k} \tag{1}$$

It is controlled by a PID law with a filtered derivative term [27], discretised at a sample period of 2.0 ms, with actuator saturation at ±300 units. Gains were tuned on the delay-free loop and then held fixed throughout, so that every difference reported is attributable to the estimator and not to retuning.

2.2 The Estimator Model

Rather than train a specific network, we characterise the estimator by the two properties that determine its effect on the loop. The first is a noise reduction factor α, the proportion of sensor noise that survives estimation, so that α = 1 is the raw sensor and α = 0.1 a tenfold reduction. The second is an inference latency L, expressed in whole control samples.

```
meas     = x + noise
estimate = x + (meas - x) * alpha      # accuracy
fed_back = estimate delayed by L samples # latency
```

This abstraction is deliberate. It covers any estimator whose net effect is to attenuate measurement noise at a computational cost [23,24], and it lets the two mechanisms be varied independently, which no real model permits. Its limitation is that it assumes an unbiased estimator with no dynamics of its own, discussed in Section 5. Figure 1 shows the loop architecture.

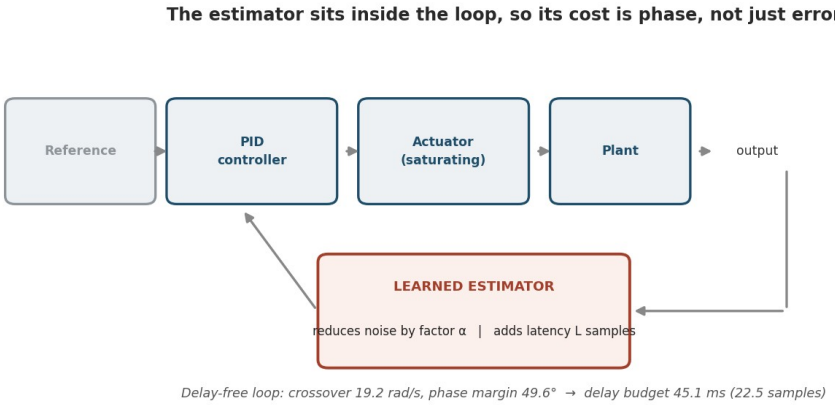


Fig.1: Loop architecture. The estimator contributes both noise attenuation and phase lag.

2.3 The Delay Budget

Before simulating, the admissible latency follows from classical margin analysis [4,26]. A pure delay τ contributes phase lag ωτ radians at frequency ω. Setting this equal to the phase margin at the gain crossover frequency ωc gives the delay at which the margin is exhausted:

$$\tau_{max} = \frac{\varphi_m}{\omega_c} \tag{2}$$

For this loop, the delay-free crossover is 19.2 rad/s with a phase margin of 49.6 degrees, giving τ_{max} = 45.07 ms, or 22.5 control samples. This single division is the design constraint that the rest of the paper tests.

2.4 Scenario and Metrics

Each run applies a unit step reference at 0.05 s and a step load disturbance at 2.0 s over a 4 s horizon. Four metrics are recorded: integral squared error over the step window; percentage overshoot; steady-state jitter, the standard deviation of the output over a settled window before the disturbance; and actuator chatter, the root mean square of the sample-to-sample control increment.

The last two are the metrics on which a good estimator should pay. Sensor noise propagates through the derivative term into the actuator, causing wear and acoustic noise without improving tracking, and attenuating it is the main practical reason to filter a signal at all.

Parameters are given in Table 1. All are illustrative and represent a lightly damped mechanical stage rather than a specific product. The simulation was implemented in Python using NumPy, SciPy and Matplotlib [28-30].

Table 1: System parameters.

Parameter	Value	Note
Natural frequency	5.00 rad/s	Lightly damped positioning stage
Damping ratio	0.240	Oscillatory without control
Control sample period	2.0 ms	500 Hz loop rate
Controller	PID with filtered derivative	Tuned on the delay-free loop, then fixed
Actuator	Saturating at ±300 units	Present in all runs
Gain crossover	19.2 rad/s	Computed from the open-loop transfer function
Phase margin	49.6°	Delay-free loop
Delay budget τmax	45.07 ms (22.5 samples)	From Equation 2

3. RESULTS

3.1 What a Free Estimator Would Buy

Table 2 establishes the upper bound. Against the raw sensor, an estimator that attenuates noise by 90 per cent at zero latency reduces steady-state jitter from 0.00674 to 0.00200, a factor of 3.4, and actuator chatter from 44.1 to 8.2, a factor of 5.4.

Tracking error and overshoot barely move (0.0537 to 0.0542, and 10.09 to 9.77 per cent). This is expected and worth stating: denoising a sensor improves regulation and actuator life, not command following, because the noise is zero-mean and the loop already rejects it in the mean.

These figures are the entire prize available from a perfect estimator. Everything that follows asks how much of it survives the cost of computing one.

Table 2: The bound: raw sensor against a zero-latency ideal estimator.

Configuration	ISE	Overshoot (%)	Steady-state jitter	Actuator chatter
Raw sensor, no estimator	0.0537	10.09	0.00674	44.08
Ideal estimator, 90% noise cut, zero latency	0.0542	9.77	0.00200	8.17
Improvement factor	**0.99×**	**—**	**3.37×**	**5.40×**

3.2 What Accuracy Bought with Latency Actually Delivers

Table 3 and Figure 2 replace the free estimator with four realistic ones, in which greater noise attenuation costs proportionally more inference time. The progression is not what offline evaluation would predict.

The light model (2 samples, 40 per cent noise cut) reduced chatter from 44.1 to 27.1 and raised overshoot from 10.09 to 12.53 per cent. The medium model (5 samples, 65 per cent) reduced chatter further, to 16.7, and raised overshoot to 17.89 per cent. The heavy model (10 samples, 80 per cent) reached chatter 11.3 at an overshoot of 32.21 per cent — more than three times the baseline.

The very heavy model (20 samples, 90 per cent noise cut) is the instructive case. Its offline accuracy is the best of the four and equals that of the ideal estimator in Table 2. In the loop it produced an overshoot of 76.0 per cent, a sustained oscillation visible in Figure 2, an

integral squared error 9.8 times the baseline, and a steady-state jitter of 0.4593 — 68 times worse than the raw sensor it was installed to improve.

The metric the estimator was designed to optimise therefore reversed. Offline, its error was the lowest of any candidate; in closed loop, the quantity that error was supposed to improve was worse than using no estimator at all. This is the central result and it is not a marginal effect. Figure 3 compares steady-state jitter, actuator chatter and overshoot across the configurations.

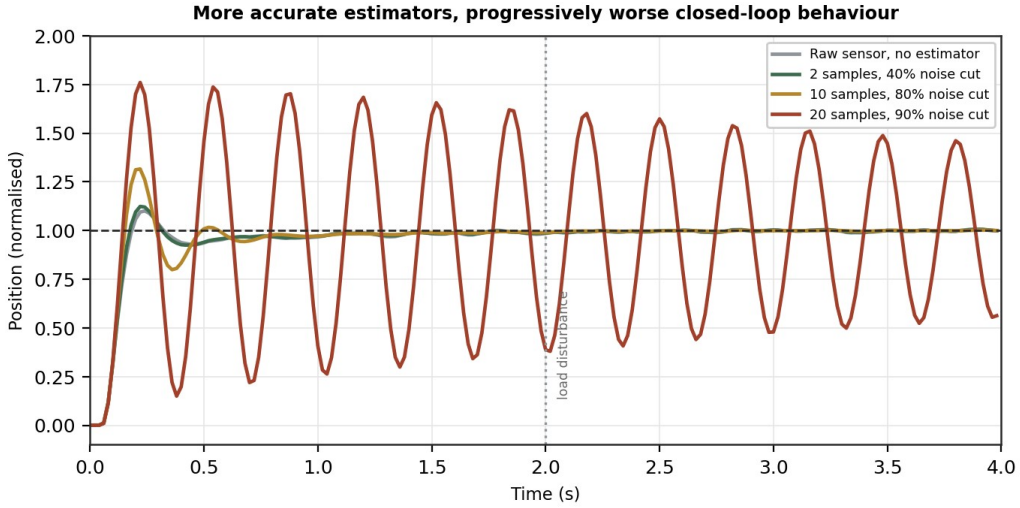


Fig.2: Closed-loop step and disturbance response for four estimator configurations.

Table 3: Realistic estimators, accuracy purchased with latency.

Configuration	Latency (samples)	Noise cut (%)	ISE	Overshoot (%)	Steady-state jitter
Raw sensor	**0**	**0**	**0.0537**	**10.09**	**0.00674**
Light model	2	40	0.0535	12.53	0.00458
Medium model	5	65	0.0541	17.89	0.00330
Heavy model	10	80	0.0607	32.21	0.00269
Very heavy model	**20**	**90**	**0.5281**	**76.00**	**0.45933**

Controller gains identical in every row. Latency in control samples of 2.0 ms.

3.3 Latency Dominates the Grid

Figure 4 and Table 4 vary the two parameters independently across a grid. Reading across a row — fixed accuracy, increasing latency — integral squared error rises from 0.0545 at 2 samples to 1.1361 at 22 samples, a factor of 21.

Reading down a column — fixed latency, increasing accuracy — the change is negligible. At 10 samples of latency, moving from the raw sensor to a 90 per cent noise cut changes error from 0.0610 to 0.0607, a difference of 0.4 per cent.

Tracking performance in this loop is therefore a function of the column and not of the row. An engineer choosing among candidate models on offline error is optimising the axis that barely matters, using an evaluation that does not measure the axis that does.

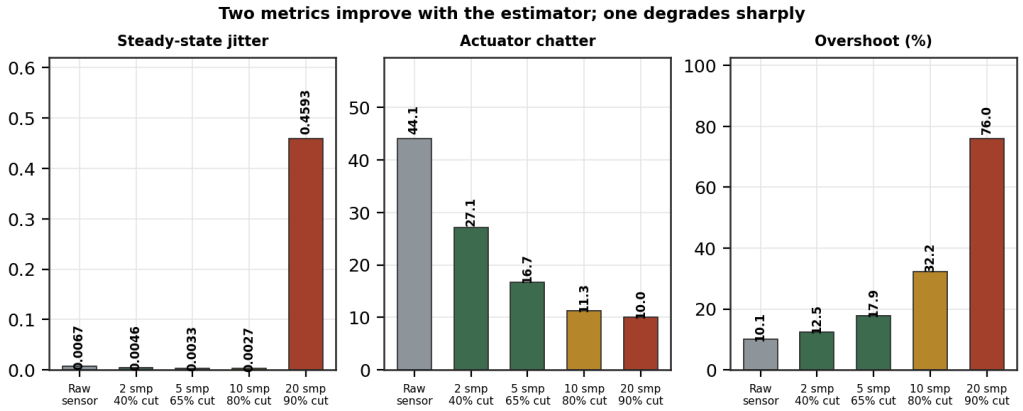


Fig.3: Steady-state jitter, actuator chatter and overshoot across configurations.

Table 5 makes the point in the form a designer would use. At every latency tested, the best achievable improvement over the raw sensor was between 0.0 and 1.3 per cent on tracking error — and at the largest latency the best available noise factor was 1.00, meaning the optimal estimator was no estimator.

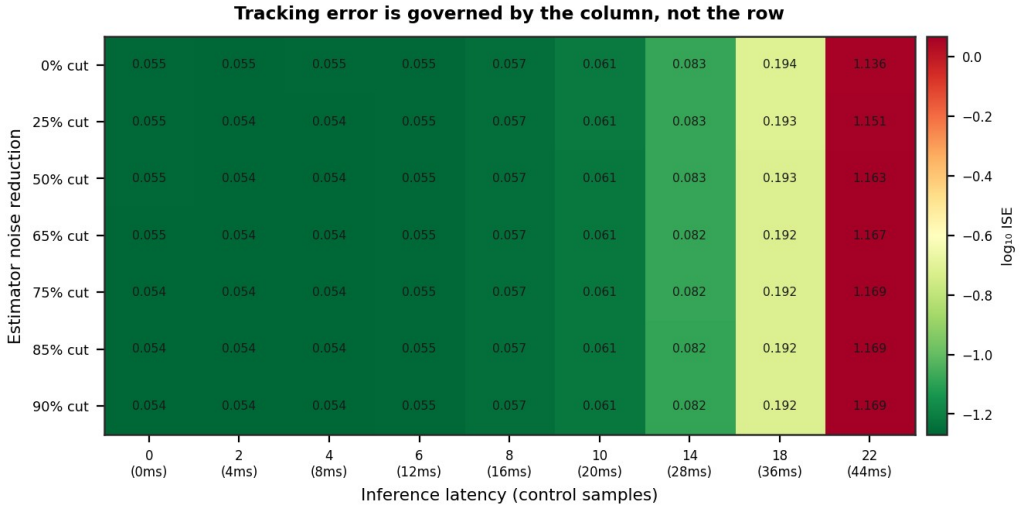


Fig.4: Integral squared error across the accuracy-latency grid.

Table 4: Integral squared error across the grid (latency in samples).

Noise cut	0 smp	2 smp	4 smp	6 smp	8 smp	10 smp	14 smp	18 smp	22 smp
0%	0.0550	0.0545	0.0546	0.0551	0.0570	0.0610	0.0831	0.1935	1.1361
25%	0.0548	0.0543	0.0544	0.0550	0.0569	0.0609	0.0829	0.1933	1.1512
50%	0.0546	0.0542	0.0543	0.0549	0.0569	0.0608	0.0826	0.1927	1.1629
65%	0.0545	0.0541	0.0542	0.0549	0.0568	0.0608	0.0824	0.1924	1.1674
75%	0.0544	0.0540	0.0541	0.0548	0.0568	0.0607	0.0822	0.1922	1.1692
85%	0.0544	0.0540	0.0540	0.0548	0.0568	0.0607	0.0821	0.1920	1.1691
90%	0.0543	0.0539	0.0540	0.0548	0.0568	0.0607	0.0820	0.1919	1.1689

Rows differ by less than 2%; columns differ by more than an order of magnitude.

Table 5: Best achievable at each latency.

Latency (samples)	Latency (ms)	Best noise factor	Best ISE	Improvement over raw sensor (%)
0	0.0	0.10	0.0543	1.2
2	4.0	0.10	0.0539	1.0
4	8.0	0.10	0.0540	1.1
6	12.0	0.10	0.0548	0.5
8	16.0	0.10	0.0568	0.4
10	20.0	0.10	0.0607	0.4
14	28.0	0.10	0.0820	1.3
18	36.0	0.10	0.1919	0.8
22	**44.0**	**1.00**	**1.1361**	**0.0**

3.4 The Analytical Budget Predicts the Collapse

Figure 5 sweeps latency at fixed accuracy and plots tracking error on a logarithmic scale against the delay budget computed in Section 2.3.

Error is essentially flat to about 6 samples, rises gently to 10, and then climbs steeply. By 22 samples it is 22 times the value at 2 samples. The analytical budget of 22.5 samples falls exactly in this region of collapse.

The agreement is worth stating plainly because it is the practical payoff. The delay budget is a single division on quantities a control engineer already has, and it located the failure region without any simulation. A team selecting an inference architecture does not need to run this study; it needs to compute φ_m/ω_c for its own loop and treat the result as a hard specification.

We note one qualification. Within the range tested the loop degraded severely but did not formally diverge, because actuator saturation bounded the oscillation. Saturation converts what would be instability into a large sustained limit cycle, which is arguably worse operationally than a clean divergence because it may be mistaken for poor tuning.

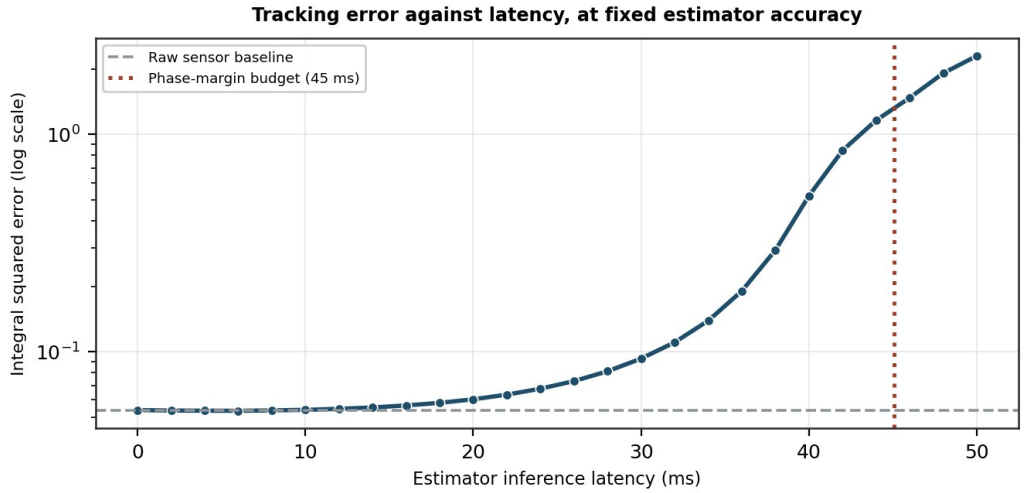


Fig.5: Tracking error against latency at fixed accuracy, with the analytical delay budget marked.

3.5 Diminishing Benefit against Accelerating Cost

Figure 6 plots what each estimator buys against what it costs. Chatter reduction saturates: moving from the heavy to the very heavy model improved chatter only from 11.3 to 10.0, while overshoot rose from 32.2 to 76.0 per cent.

The shape of the two curves is the design argument. Noise attenuation has diminishing returns by construction — each further reduction removes a smaller absolute quantity of noise — while phase lag accumulates linearly in latency and its consequences grow faster than linearly as the margin approaches zero. A diminishing benefit set against an accelerating cost has an interior optimum, and in this loop that optimum was at low latency and modest accuracy.

Table 6 summarises which metrics respond to which axis, and it is the table we would give to a team deciding where to place a learned component.

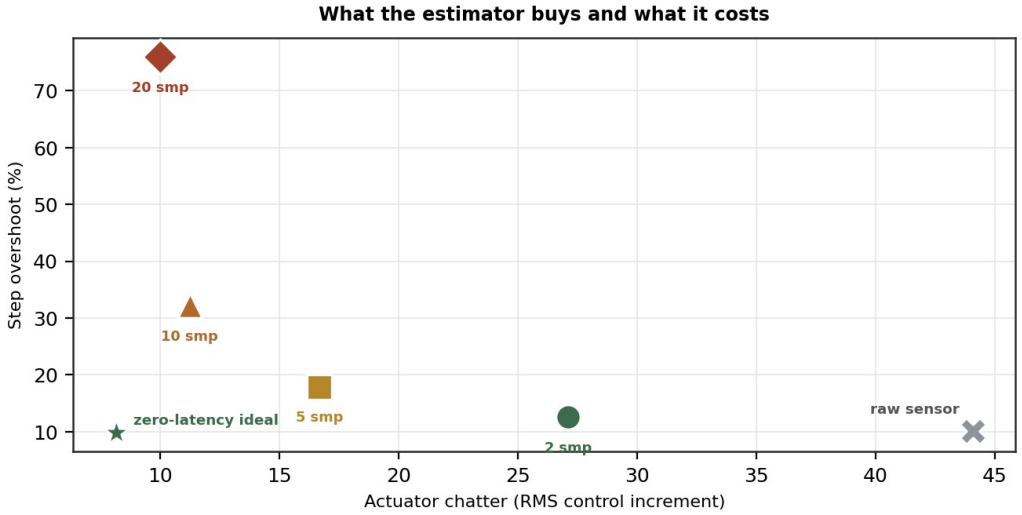


Fig.6: Actuator chatter against overshoot for each configuration.

Table 6: Which axis governs which metric.

Metric	Responds to accuracy	Responds to latency	Design implication
Tracking error (ISE)	Negligibly	Strongly	Set by the delay budget
Overshoot	Negligibly	Strongly	Set by the delay budget
Steady-state jitter	Strongly, until the loop degrades	Strongly once margin is consumed	Benefit reverses at high latency
Actuator chatter	Strongly, with saturation	Weakly	The one axis accuracy genuinely buys

4. DISCUSSION

4.1 Principal Findings

Four findings follow. A zero-latency ideal estimator reduced actuator chatter by a factor of 5.4 and steady-state jitter by a factor of 3.4, which bounds the available benefit. Across the accuracy-latency grid, tracking error varied by more than an order of magnitude along the latency axis and by under 2 per cent along the accuracy axis. The most accurate estimator tested degraded steady-state jitter to 68 times the raw-sensor value, reversing the quantity it was designed to improve. And the analytical delay budget φ_m/ω_c located the collapse without simulation.

The implication for practice is a division of labour rather than a prohibition. Learned components are valuable and the benefit measured here is real; it is simply bounded, and it is available only to components fast enough to sit inside the loop. Heavier models belong outside the feedback path — in supervision, fault detection, gain scheduling or trajectory generation — where their latency is not phase lag.

4.2 Relation to Existing Work

Delay margin and the phase cost of computation are standard results in control theory [4,7], and the real-time and networked-control literatures treat sampling and delay as first-class design variables [5,6,8]. Learning-based estimation and control are active fields with substantial results on stability guarantees for learned components [9-11], and co-design of computation and control has been studied under the heading of anytime or resource-aware control [12,13].

What we believe this study adds is an empirical magnitude comparison for a concrete loop: not that latency matters, which is known, but that in this system it dominated accuracy by more than two orders of magnitude in its effect on closed-loop error. That ratio is the quantity a design team needs and it is not obtainable from either literature separately.

4.3 Design Guidance

Figure 7 and Table 7 set out the recommendations. Two deserve comment.

Compute the delay budget first and treat it as a specification. φ_m/ω_c requires only the open-loop transfer function, takes a minute, and converts an open-ended model-selection problem into a bounded one: the question becomes which is the most accurate model that fits inside the budget, which is a well-posed engineering question with a defensible answer.

Specify worst-case latency, not mean. A loop experiences the delay of each individual inference, not the average across a benchmark. An architecture with good mean latency and a long tail — common where memory allocation, cache behaviour or shared accelerators are involved — will intermittently exceed the budget, producing transient oscillation that is difficult to diagnose precisely because it is intermittent [16-18].

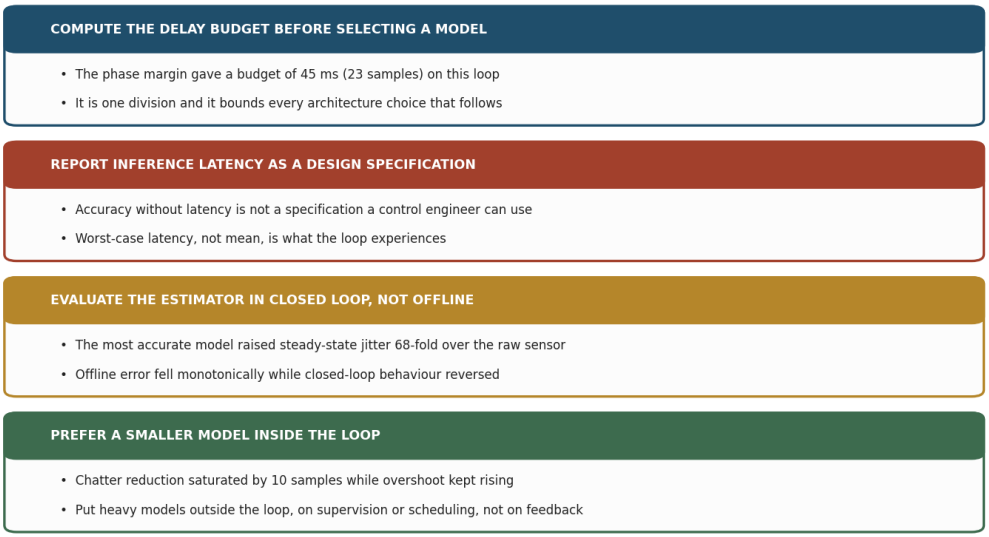


Fig.7: Proposed practice for placing learned components in feedback loops.

Table 7: Recommendations.

Recommendation	Addresses	Cost	Evidence here
Compute φ_m/ω_c before selecting a model	Model selection unbounded by the loop	One calculation	Sections 2.3, 3.4
Specify worst-case inference latency	Tail latency exceeding the budget intermittently	Profiling	Section 4.3
Evaluate the estimator in closed loop	Offline error that reverses in the loop	One simulation harness	Section 3.2
Prefer the smallest model inside the loop	Diminishing benefit against accelerating cost	Accepting lower offline accuracy	Section 3.5
Place heavy models outside the feedback path	Latency that need not be phase lag	Architectural	Section 4.1
Watch for limit cycles, not divergence	Saturation masking a margin failure as poor tuning	None	Section 3.4

5. LIMITATIONS

The estimator is modelled as an unbiased noise attenuator with a pure delay. A real learned estimator has dynamics of its own, may be biased, and may have error correlated with the state — all of which interact with the loop in ways a pure delay does not capture. In particular a model whose error grows outside its training distribution introduces a state-dependent disturbance that this abstraction excludes entirely [25], and which we would expect to be more dangerous than the effect we measured.

Accuracy and latency are varied independently here, whereas in a real model family they are coupled by architecture. Our grid therefore includes combinations no real model offers. This is deliberate — it is what isolates the two mechanisms — but the practical design space is a curve through the grid, not the whole grid.

The controller was tuned on the delay-free loop and held fixed. A designer who knows the latency in advance can retune, or use a predictor such as a Smith compensator [14,15], and would recover part of the loss. Our results describe the cost of inserting a component into an existing loop, which is the common situation, and overstate the cost where co-design is possible.

We simulated one plant, one controller structure and one sample rate. The delay budget scales with φ_m/ω_c, so a slower loop tolerates proportionally more latency and a faster one less; the qualitative conclusion transfers but the numbers do not. We also did not model quantisation, communication jitter, or multi-rate execution, each of which adds further effective delay.

Finally, actuator saturation bounded the degradation in every run, so we report severe limit cycles rather than formal instability. A loop with more actuator authority would have diverged outright at latencies where ours merely oscillated, and the boundary we report is therefore specific to this saturation limit.

6. CONCLUSION

Inserting a learned estimator into a feedback loop trades noise attenuation against phase lag. In the loop studied, the whole available benefit was a 5.4-fold reduction in actuator chatter and a 3.4-fold reduction in steady-state jitter, obtainable only at zero latency.

Against that, latency moved closed-loop tracking error by more than an order of magnitude while accuracy moved it by under 2 per cent. The most accurate estimator tested, at 20 samples of latency, produced 76 per cent overshoot and a steady-state jitter 68 times worse

than the raw sensor it replaced — the metric it was selected to optimise, reversed by the loop it was placed in.

The admissible latency is a property of the loop and not of the model, and it is available before any model is trained: φ_m/ω_c gave 45.1 ms here and correctly located the region where behaviour collapsed. Computing it first turns model selection from an open-ended search for accuracy into a bounded engineering choice, which is the form the problem should have had from the start.

DECLARATIONS

Ethics approval and consent to participate. Not applicable. This study is a simulation study using synthetic data generated by the author. It did not involve human participants, animals or personal data.

Consent for publication. Not applicable.

Data availability statement. This is a simulation study; no external or experimental datasets were used. All results were generated by a simulation script developed by the author in Python using NumPy, SciPy and Matplotlib [28-30], with fixed random seeds. Every model parameter is reported in full in Table 1, and all figures and all values in Tables 2 to 5 can be reproduced exactly from these parameters. The simulation script is available from the corresponding author upon reasonable request.

Conflict of interest. The author declares no competing interests, financial or non-financial, in relation to the subject matter of this manuscript. The author has no financial or advisory relationship with any manufacturer of control hardware, inference accelerators or motion systems.

Funding. This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors.

Author's contribution. Dr. Ramakrishna Kumar is the sole author of this work and was responsible for the conception and design of the study, formulation of the delay-budget analysis, implementation of the simulation, analysis and verification of all reported values, development of the design guidance, and drafting, revision and final approval of the manuscript. The author is accountable for all aspects of the work.

Acknowledgements. None declared.

REFERENCES

1. Hewing L, Wabersich KP, Menner M, Zeilinger MN. Learning-based model predictive control: toward safe learning in control. Annu Rev Control Robot Auton Syst. 2020;3:269–96.
2. Brunton SL, Noack BR, Koumoutsakos P. Machine learning for fluid mechanics. Annu Rev Fluid Mech. 2020;52:477–508.
3. Levine S, Finn C, Darrell T, Abbeel P. End-to-end training of deep visuomotor policies. J Mach Learn Res. 2016;17(39):1–40.
4. Astrom KJ, Murray RM. Feedback systems: an introduction for scientists and engineers. 2nd ed. Princeton: Princeton University Press; 2021.
5. Franklin GF, Powell JD, Workman ML. Digital control of dynamic systems. 3rd ed. Menlo Park: Addison-Wesley; 1998.
6. Zhang W, Branicky MS, Phillips SM. Stability of networked control systems. IEEE Control Syst Mag. 2001;21(1):84–99.
7. Zhong QC. Robust control of time-delay systems. London: Springer; 2006.

8. Hespanha JP, Naghshtabrizi P, Xu Y. A survey of recent results in networked control systems. Proc IEEE. 2007;95(1):138–62.
9. Recht B. A tour of reinforcement learning: the view from continuous control. Annu Rev Control Robot Auton Syst. 2019;2:253–79.
10. Dean S, Mania H, Matni N, Recht B, Tu S. On the sample complexity of the linear quadratic regulator. Found Comput Math. 2020;20(4):633–79.
11. Berkenkamp F, Turchetta M, Schoellig AP, Krause A. Safe model-based reinforcement learning with stability guarantees. Adv Neural Inf Process Syst. 2017;30:908–18.
12. Bini E, Cervin A. Delay-aware period assignment in control systems. Proc IEEE Real-Time Syst Symp. 2008;291–300.
13. Cervin A, Henriksson D, Lincoln B, Eker J, Arzen KE. How does control timing affect performance? Analysis and simulation of timing using Jitterbug and TrueTime. IEEE Control Syst Mag. 2003;23(3):16–30.
14. Smith OJM. A controller to overcome dead time. ISA J. 1959;6(2):28–33.
15. Palmor ZJ. Time-delay compensation: Smith predictor and its modifications. In: The Control Handbook. Boca Raton: CRC Press; 1996. p. 224–37.
16. Nilsson J, Bernhardsson B, Wittenmark B. Stochastic analysis and control of real-time systems with random time delays. Automatica. 1998;34(1):57–64.
17. Gupta RA, Chow MY. Networked control system: overview and research trends. IEEE Trans Ind Electron. 2010;57(7):2527–35.
18. Han S, Zhu X, Mok AK, Chen D, Nixon M. Reliable and real-time communication in industrial wireless mesh networks. Proc IEEE Real-Time Technol Appl Symp. 2011;3–12.
19. Sze V, Chen YH, Yang TJ, Emer JS. Efficient processing of deep neural networks: a tutorial and survey. Proc IEEE. 2017;105(12):2295–329.
20. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, et al. MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861. 2017.
21. Reddi VJ, Cheng C, Kanter D, Mattson P, Schmuelling G, Wu CJ, et al. MLPerf inference benchmark. Proc ACM/IEEE Int Symp Comput Archit. 2020;446–59.
22. Lee EA. Cyber physical systems: design challenges. Proc IEEE Int Symp Object Compon Serv Real-Time Distrib Comput. 2008;363–69.
23. Kalman RE. A new approach to linear filtering and prediction problems. J Basic Eng. 1960;82(1):35–45.
24. Simon D. Optimal state estimation: Kalman, H-infinity, and nonlinear approaches. Hoboken: Wiley; 2006.
25. Ljung L. System identification: theory for the user. 2nd ed. Upper Saddle River: Prentice Hall; 1999.
26. Skogestad S, Postlethwaite I. Multivariable feedback control: analysis and design. 2nd ed. Chichester: Wiley; 2005.
27. Aastrom KJ, Hagglund T. Advanced PID control. Research Triangle Park: ISA; 2006.
28. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat Methods. 2020;17(3):261–72.
29. Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, et al. Array programming with NumPy. Nature. 2020;585(7825):357–62.
30. Hunter JD. Matplotlib: a 2D graphics environment. Comput Sci Eng. 2007;9(3):90–95.