Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Data Driven Estimation of Series-RLC Parameters

Kunal Kashyap
Indian Institute of Technology Roorkee
Mayank Goswami
mayankfph@iitr.ac.in

Indian Institute of Technology Roorkee

Research Article

Keywords: RLC circuits, parameter estima- tion, neural networks, MLP, nonlinear least squares, noise augmentation, transient response

Posted Date: November 10th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11652840/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Data Driven Estimation of Series-RLC Parameters

Kunal Kashyap[1] and Mayank Goswami,[1,2]

[1]Department of Physics, IIT Roorkee
[2]MF School of AI and DS, IIT Roorkee
mayank.goswami@ph.iitr.ac.in

***Abstract*—Estimating circuit parameters requires balancing accuracy, measurement requirements, and processing cost. This study compares multiple multilayer perceptrons [MLP] with bounded nonlinear least squares fitting. The task at hand is parameter estimation for an ideal series RLC circuit driven with a known voltage step. A synthetic dataset was constructed contained 10,000 circuits. From this adjoint training, validation and testing splits were derived. Clean-trained and noise-augmented networks were evaluated with Gaussian noise additions. Comparison protocols for observation duration and sampling density were also established. At 10% noise, relative to clean-training channel RMS, the full-record augmented network achieves average median resistance, inductance, and capacitance errors of 2.745%, 1.971%, and 2.012%, respectively. Historical full-grid CPU timings are approximately 0.048 ms for neural inference versus 6.926 ms for fitting, measured in separate earlier experiment runs. This excludes the initial 50 ms to acquired the signal. At 101 samples per channel, a fixed five start diagnostic removed 52 clean cases where at least component had an error greater than 10%. This also reduces the 10%-noise incidence above 50% from 6.73% to 3.40%, still above the augmented network's 1.81%. These results are implementation specific and do not establish neural superiority. Hardware effects and model mismatch remain outside scope.**

***Index Terms*—RLC circuits, parameter estimation, neural networks, MLP, nonlinear least squares, noise augmentation, transient response.**

I. Introduction

The study is based on an inverse problem. The observed waveforms are available but the resistance, inductance and capacitance that generated them must be recovered. When circuit topology, excitation and initial conditions are known this can be achieved by fitting a model to the known measurements. An alternative approach is to employ supervised learning to approximate the inverse mapping.

This study also involves noise sensitivity, processing latency, and acquisition requirements.

The study does not claim superiority of the learned estimator. A physics based estimator fits the known circuit response directly to noise measurements from the inverse mapping it learns across samples. However, good typical accuracy can still mean occasional large errors.

This study evaluates these distinctions in a controlled RLC setting. The contributions can be listed: (i) a paired comparison of clean-trained and noise-augmented networks. (ii) separate observation-window and fixed-duration sampling studies. (iii) joint reporting of parameter-error distributions and processing costs. Acquisition time is treated separately in both cases.

The work does not introduce a new neural architecture or claim a first application of neural networks to circuit identification.

II. Background and Study Positioning

Forgione and Piga [1] showed that a continuous-time dynamical model can be described by a network. Their outputs described system dynamics and simulated trajectories.

Bishop [2] relates training with small input perturbations to generalized Tikhonov regularization under the paper's assumptions. This motivated this studies investigations with artificial noise. It does not guarantee better performance at finite noise levels and only infer a theoretical robustness guarantee.

Previous papers have included multiple classical system identification methods such as least-squares and prediction-error estimation of parameterized dynamical models [3] and as subspace methods for estimating linear state-space realizations [4]. For this known RLC topology, the analytical solution is well known and directly fitting a model is a simple task.

The physical comparator is a bounded nonlinear least-squares implementation. It uses a fixed initialization and numerical derivatives. It is possible that alternative initialization and multistart searches improve the analytical estimators runtime and accuracy but that has not been explored.

III. Circuit Model and Dataset

A. Forward Model and Observability

For a known series circuit driven by a 1 V step, the current i and capacitor voltage v_C follow

$$L\frac{di}{dt} + Ri + v_C = V_0, \qquad V_0 = 1\ \text{V}, \tag{1}$$

$$C\frac{dv_C}{dt} = i, \tag{2}$$

with $i(0) = v_C(0) = 0$. Both signals are measured. With ideal continuous measurements, C is related to $i/(dv_C/dt)$ wherever the derivative is nonzero, and the first equation constrains R and L. The voltage transfer function

$$\frac{V_C(s)}{V_{\text{in}}(s)} = \frac{1}{LCs^2 + RCs + 1} \tag{3}$$

contains only the products LC and RC. Therefore voltage alone cannot uniquely determine all three components without additional information, only their products. Neither estimator numerically differentiates the measured waveforms. However, the fitter does use finite differences with respect to model parameters to approximate its Jacobian.

The simulator evaluates analytical step responses in float64. Stored waveforms use float32.

B. Parameter Sampling and Splits

The dataset contains 10,000 independent log-uniform parameter draws: $R \in [10, 100]$ Ω, $L \in [10, 50]$ mH, and $C \in [1, 10]$ μF. Each circuit has a distinct parameter combination and belongs to exactly one split: 7,000 training, 1,500 validation, or 1,500 test circuits. All seeds used to generate parameters and split shuffling were different and recorded. All additional noise retains the source circuits split. Training perturbations are not stored as an expanded noisy dataset separately.

The full record spans 0–50 ms, inclusive, at intervals of 10 μs. This results in 5,001 samples per channel. Signals retain physical amplitudes and there is no individual waveform peak normalization. The test set contains 1,494 underdamped and six overdamped circuits, with no exactly critical circuits. Only one test circuit satisfies $|\zeta - 1| < 0.01$, where $\zeta = (R/2)\sqrt{C/L}$. Consequently, the results principally characterize the underdamped portion of this distribution.

IV. Estimators and Experimental Protocol

A. Direct Neural Regression

For N samples per channel, current samples followed by voltage samples form a flattened $2N$-dimensional vector. The network is

$$2N \longrightarrow 128 \longrightarrow 64 \longrightarrow 3, \tag{4}$$

with ReLU hidden activations and linear outputs. One mean and standard deviation per input channel are fitted across clean training circuits and retained time points. The three linear outputs are standardized natural-log targets, $z_p = (\ln p - \mu_p)/s_p$, with μ_p and s_p fitted only to training labels. Training minimizes MSE averaged over circuits and the three standardized targets. The predictions can then be inverse transformed to physical units.

Training uses Adam [5] with learning rate 0.001, batch size 128, a maximum of 200 epochs, and early stopping after 25 epochs without validation improvement. Three matched clean/augmented pairs use seeds 20261014–20261016. Within a pair, initialization, shuffled circuit order, pre-processing, and optimizer settings match. Noise generation uses a separate random-number stream. Training lasted 28–200 epochs (with a median of 140.5) across 54 distinct checkpoints. 15 reached the 200-epoch cap.

Except the input dimensions, the architecture and optimizer settings were retained across conditions. This study does not perform a hyperparameter search or establish how the original widths and learning rate were selected. Checkpoint selection has been documented below however.

Clearly total parameter count is not constant. Shortening or subsampling the input also reduces the first layer. It can be inferred that this is a comparison of retrained estimators that match on inputs. There is no universal network that can accept arbitrary samples.

B. Noise and Checkpoint Selection

For a clean channel x_c, the measured channel is

$$y_c(t_k) = x_c(t_k) + \epsilon_{c,k}, \qquad \epsilon_{c,k} \sim \mathcal{N}(0, \lambda^2 \rho_c^2), \tag{5}$$

where ρ_c is the RMS over the original full-duration clean training channel, including the DC voltage. The training and original evaluation levels are $\lambda \in \{0, 0.01, 0.02, 0.05, 0.10\}$. A final fixed-checkpoint stress test additionally evaluates unseen amplitudes 0.15 and 0.20. These measurements mean that "10% noise" is not 10% of each waveform's peak. At that level, the standard deviations are approximately 124.3 μA and 0.1003 V. Noise is independent across circuits, channels, and samples, without filtering or clipping.

In each epoch, for each circuit, a noise level is independently and uniformly chosen from the five levels. That level applies to both channels and all retained time points of that circuit. On subsequent presentations, they are drawn afresh. Validation and testing use fixed seed realizations. Full grid banks are generated deterministically per batch. After each epoch, both the clean and augmented networks were evaluated for the five conditions and the checkpoint with the lowest average was retained. An older baseline used only clean validation to choose its checkpoint. Because that selection rule differs, it was not counted among the three matched seed pairs.

C. Physical Fitting

SciPy's `least_squares` [6] minimizes

$$\min_{\boldsymbol{\theta} \in \mathcal{B}} \sum_{c,k} \left[\frac{f_c(t_k; e^{\boldsymbol{\theta}}) - y_c(t_k)}{\rho_c} \right]^2, \tag{6}$$

where $\boldsymbol{\theta} = \log(R, L, C)$, f_c is the analytical response, and $\mathcal{B}$ is the log-transformed dataset range. Each fit runs once, beginning from the same initial guess for R,L,C, approximately with a geometric range center of (31.6228 Ω, 22.3607 mH, 3.16228 μF). The guesses do not come from a neural network or a previous circuit. No labels are provided but the allowed parameter ranges are. This prevents them from guessing outside that range. The comparison therefore does not impose identical output constraints.

The solver uses trust-region reflective optimization, a two-point finite-difference Jacobian, ordinary squared loss, `ftol`, `xtol`, and `gtol` each set to 10^{-10}, and a maximum of 200 function evaluations. The residuals are evaluated in float64.

D. Observation Ablations and Pairing

The window study retains 1, 2, 5, 10, 20, or 50 ms at the original sampling interval, giving 101, 201, 501, 1,001, 2,001, or 5,001 samples per channel. The sampling study instead retains 5,001, 1,001, 501, or 101 uniformly selected samples over the same 50 ms duration. The samples are selected without any anti-alias filtering.

Validation and test perturbations are generated on the full grid before cropping or selecting points. All estimators use identical physical measurements rounded to float32 representation, with noise hashes verifying pairing. The physical noise scale is held constant across observation conditions. The reduced-observation studies copy the previously trained full-grid checkpoints, predictions, and timings for their full-grid reference rows rather than rerunning them. Previously computed full-grid results were reused after checking circuit identities, target values, noise hashes, and the origin of the saved files. Validation and test noise are paired across observation shapes, but training noise is not.

E. Metrics and Timing

For a true component p, the absolute relative error is

$$e_p = 100\frac{|\widehat{p} - p|}{p} \quad (\%). \tag{7}$$

For each parameter, seed, and noise realization, we first compute the median absolute relative error over the same 1,500 test circuits. At a positive noise level we average the three realization medians within each seed, then average those values across the three training seeds. At zero noise there is one clean condition per seed. Fitting has no training seeds and averages only the realization statistics. These are averages of medians, not medians of pooled predictions. Where shown, $\pm$ denotes the sample SD of the three seed-level averages (denominator $3-1$), not a confidence interval or spread across circuits.

The MLP is implemented in PyTorch and the physical solver in SciPy. CPU timings use one thread. Neural single-request timings use batch size one, five untimed warmup requests, and fifty timed repetitions. Each timed request includes preprocessing, inference, and decoding to physical parameters. Fitting timings cover the actual serial solves, including conversion of measurements to float64, residual scaling, optimization, and decoding of the fitted parameters.

Disk access, noise generation, and training times are excluded. Observation duration is reported separately from compute time wherever relevant. All computation was done an a Apple M3 Pro.

V. Results and Discussion

A. Noise Augmentation and Dense Measurements

Table I reports the full noise sweep. At 10% noise, the average median $R/L/C$ errors for the full-grid clean trained MLP is 6.913/3.162/4.421%. The augmented MLP has corresponding values 2.745/1.971/2.012%. On clean inputs the corresponding errors are 0.791/0.683/0.678% and 0.958/0.677/0.815%. Augmentation therefore slightly increases clean-input resistance error (0.791% to 0.958%) and capacitance error (0.678% to 0.815%). The small inductance difference (0.683% versus 0.677%) is not evidence of improvement.

Under the matched ideal circuit model, physical fitting remains much more accurate than either neural estimator. Its full-grid 10%-noise median errors are 0.162/0.178/0.162%. All 19,500 full-grid solves reported convergence, and 39 reached at least one active bound. Maximum component errors across these full-grid conditions were approximately 1.532/2.049/2.297%.

The study also conducted a stress test, retaining all six full-grid checkpoints and the same 1,500 testing circuits. It evaluated at 15% and 20% Gaussian noise with the original clean trained RMS (Table II). The noise-augmented network median $R/L/C$ errors rose to 3.926/2.824/2.887% and 5.133/3.715/3.720%, respectively, but remained below matched clean controls. At 15% and 20% measurement noise, the average fractions of circuits with any component error above 10% were 16.037% and 31.393% for augmented networks, versus 58.652% and 69.652% for clean-trained controls. The corresponding fractions above 50% were 0% and 0.022%, versus 4.341% and 9.059%.

B. Shorter Observation Windows

Table III shows that fitting accuracy improves as the window increases. There is limited additional median improvement beyond approximately 10–20 ms in the tested range and the augmented network already gives 2.933/0.816/0.928% median errors from a 1 ms observation.

Some shorter-window networks have lower mean median errors than the 50 ms network, but this should not be generalized. The seed to seed variation is reported in Table IV.

C. Sparse Measurements and Error Tails

Fitting retains a lower median error at every density tested (Table V). At 101 samples per channel, the single-start fitter has lower median errors, but higher mean and 95th-percentile errors for R and L. Thus, its typical performance is better, while its high-error cases are worse than the augmented network's. Table VI reports this.

At 101 samples, the augmented network has more cases with a greater than 10% error but it has fewer greater than 50% (Table VII) errors. The augmented network produces fewer extreme errors, but does not perform better uniformly. On clean sparse inputs, fitting exceeds 10% error in 52 of 1,500 circuits despite successful solver statuses. These examples demonstrate poor optimizer solutions but can still mean that the parameters can be attainted through different starting conditions.

At this density, 0.5 ms spacing corresponds to a 2 kHz sampling rate and 1 kHz Nyquist frequency, and

TABLE I. Full-grid median relative errors (%) across the original training-family noise levels. Neural entries are mean ± sample SD across three training seeds after averaging noise realizations within each seed; fitting entries average realizations only. CMLP: clean-trained; NMLP: noise-augmented; NLS: physical fitting. Clean fitting errors round to zero, not exact zero.

Noise (%)	Method	R	L	C
0	CMLP	0.791 ± 0.163	0.683 ± 0.010	0.678 ± 0.129
0	NMLP	0.958 ± 0.099	0.677 ± 0.058	0.815 ± 0.210
0	NLS	0.000	0.000	0.000
1	CMLP	1.190 ± 0.120	0.767 ± 0.031	0.922 ± 0.138
1	NMLP	1.022 ± 0.121	0.718 ± 0.058	0.845 ± 0.194
1	NLS	0.016	0.018	0.016
2	CMLP	1.792 ± 0.114	0.980 ± 0.065	1.295 ± 0.131
2	NMLP	1.137 ± 0.114	0.805 ± 0.068	0.921 ± 0.175
2	NLS	0.032	0.036	0.031
5	CMLP	3.732 ± 0.059	1.736 ± 0.106	2.458 ± 0.184
5	NMLP	1.660 ± 0.178	1.208 ± 0.111	1.280 ± 0.184
5	NLS	0.081	0.090	0.080
10	CMLP	6.913 ± 0.063	3.162 ± 0.104	4.421 ± 0.094
10	NMLP	2.745 ± 0.348	1.971 ± 0.227	2.012 ± 0.276
10	NLS	0.162	0.178	0.162

TABLE II. Unseen Gaussian-amplitude stress test: full-grid median errors (%), mean ± sample SD across three fixed training seeds after averaging three realization medians per seed. No retraining or selection.

Noise (%)	Model	R	L	C
15	CMLP	10.417 ± 0.153	4.709 ± 0.145	6.643 ± 0.224
15	NMLP	3.926 ± 0.379	2.824 ± 0.340	2.887 ± 0.374
20	CMLP	13.681 ± 0.127	6.164 ± 0.186	8.762 ± 0.303
20	NMLP	5.133 ± 0.485	3.715 ± 0.415	3.720 ± 0.460

TABLE III. Average median errors (%) at 10% noise versus window duration. Entries are $R/L/C$; CMLP: clean-trained, NMLP: noise-augmented, NLS: physical fitting.

Window (ms)	CMLP	NMLP	NLS
1	3.659 / 1.104 / 1.227	2.933 / 0.816 / 0.928	0.830 / 0.370 / 0.382
2	2.966 / 1.225 / 1.146	2.310 / 0.983 / 0.869	0.305 / 0.242 / 0.207
5	2.522 / 1.512 / 1.405	1.614 / 0.967 / 1.017	0.192 / 0.190 / 0.171
10	2.710 / 1.655 / 1.794	1.825 / 1.314 / 1.303	0.168 / 0.180 / 0.165
20	3.835 / 2.143 / 2.736	2.077 / 1.417 / 1.407	0.163 / 0.178 / 0.162
50	6.913 / 3.162 / 4.421	2.745 / 1.971 / 2.012	0.162 / 0.178 / 0.162

some allowed circuits oscillate above that frequency. Aliasing and optimizer difficulty are plausible contributors, but their causal contributions were not isolated. Fast underdamped transients provide another possible source of difficulty: at $\tau = 0.2$ ms, the envelope of the response has fallen to $e^{-2.5} \approx 0.082$ by the second sample, 0.5 ms after the step. Thus few samples may resolve the informative transient even apart from oscillatory aliasing.

A separate test checked whether the fitter's large errors were caused by its initial guess. Using the same 101 samples per channel and 1,500 test circuits, each fit was run from five fixed starting points: the original geometric center and four log-uniformly sampled points generated with seed 20261019. The same five points were used for every circuit, with unchanged solver settings and a maximum of 200 function evaluations per start. The solution with the lowest residual cost was retained, without using the true component values. This removed all 52 clean-input cases with any component error above 10%; the largest remaining R/L/C errors were 4.582/4.577/4.806%. Across the three original 10%-noise realizations, the fraction above 10% error fell from 17.44% to 14.60%, and the fraction above 50% fell from 6.73% to 3.40%. These are 4,500 repeated observations of the same circuits, not 4,500 independent circuits. Mean R/L/C errors fell from 22.891/15.412/6.128% to 4.559/6.207/5.355%. However, median processing time increased from 1.652 ms for one start to 9.012 ms for five starts, measured in the same run. Poor initialization therefore explains much of the original extreme-error behavior, although the five-start rate above 50% remained higher than the augmented network's 1.81%. Five starts do not guarantee the best possible fit, and solver convergence does not guarantee accurate parameters.

We also examined how errors varied with transient decay time and damped oscillation frequency, using saved predictions without retraining or refitting. Circuits were grouped using fixed physical ranges; this analysis covered the 1,494 underdamped test circuits and excluded the six overdamped cases. At 10% noise, the relationship between decay time and augmented-network error changed with observation duration: errors tended to increase with decay time for the 1 ms network but decrease with decay time for the 50 ms network. Sparse fitting was less reliable for higher-frequency and faster-decaying circuits. Among the 160 circuits oscillating at 1–2 kHz, the fraction with any component error above 10% fell from 65.00% with one start to 38.13% with five starts; below 0.5 kHz, it was approximately 3.41% for either method. With five starts, this fraction was 53.41% for decay times of 0.2–0.5 ms and zero for 5–10 ms. These patterns suggest that rapidly changing transients are harder to estimate from sparse samples, but

TABLE IV. Neural median errors (%) at 10% noise, reported as mean ± sample SD across three training seeds after averaging three realization medians within each seed. Window rows retain the original spacing; sample-count rows span 50 ms. The 50 ms and 5,001-sample entries reuse the same checkpoints and are not independent replications.

Observation	Method	R	L	C
1 ms	CMLP	3.659 ± 0.057	1.104 ± 0.072	1.227 ± 0.081
1 ms	NMLP	2.933 ± 0.251	0.816 ± 0.082	0.928 ± 0.099
2 ms	CMLP	2.966 ± 0.081	1.225 ± 0.037	1.146 ± 0.079
2 ms	NMLP	2.310 ± 0.225	0.983 ± 0.064	0.869 ± 0.058
5 ms	CMLP	2.522 ± 0.080	1.512 ± 0.081	1.405 ± 0.032
5 ms	NMLP	1.614 ± 0.217	0.967 ± 0.128	1.017 ± 0.101
10 ms	CMLP	2.710 ± 0.239	1.655 ± 0.080	1.794 ± 0.149
10 ms	NMLP	1.825 ± 0.356	1.314 ± 0.236	1.303 ± 0.128
20 ms	CMLP	3.835 ± 0.129	2.143 ± 0.074	2.736 ± 0.021
20 ms	NMLP	2.077 ± 0.141	1.417 ± 0.055	1.407 ± 0.069
50 ms	CMLP	6.913 ± 0.063	3.162 ± 0.104	4.421 ± 0.094
50 ms	NMLP	2.745 ± 0.348	1.971 ± 0.227	2.012 ± 0.276
5001 samples	CMLP	6.913 ± 0.063	3.162 ± 0.104	4.421 ± 0.094
5001 samples	NMLP	2.745 ± 0.348	1.971 ± 0.227	2.012 ± 0.276
1001 samples	CMLP	14.362 ± 1.126	6.566 ± 0.508	9.360 ± 0.194
1001 samples	NMLP	2.856 ± 0.088	2.280 ± 0.052	2.222 ± 0.099
501 samples	CMLP	21.053 ± 2.265	8.976 ± 0.280	11.516 ± 0.530
501 samples	NMLP	2.611 ± 0.034	2.349 ± 0.050	2.213 ± 0.103
101 samples	CMLP	19.088 ± 0.522	18.959 ± 0.776	12.903 ± 0.533
101 samples	NMLP	3.709 ± 0.097	4.221 ± 0.308	3.510 ± 0.198

TABLE V. Average median errors (%) at 10% noise for a fixed 50 ms duration. Entries are $R/L/C$; estimator labels follow Table III.

Samples	CMLP	NMLP	NLS
5001	6.913 / 3.162 / 4.421	2.745 / 1.971 / 2.012	0.162 / 0.178 / 0.162
1001	14.362 / 6.566 / 9.360	2.856 / 2.280 / 2.222	0.360 / 0.398 / 0.366
501	21.053 / 8.976 / 11.516	2.611 / 2.349 / 2.213	0.493 / 0.552 / 0.502
101	19.088 / 18.959 / 12.903	3.709 / 4.221 / 3.510	1.327 / 1.850 / 1.569

TABLE VI. Mean and 95th-percentile errors (%) at 101 samples/channel and 10% noise. Each statistic is computed over 1,500 circuits separately for each realization, then averaged over three realizations and, for NMLP, three training seeds. The p95 entries are averages of condition-specific percentiles, not pooled percentiles.

Statistic	R	L	C
NMLP mean	6.093	7.541	6.089
NLS mean	22.891	15.412	6.128
NMLP p95	19.981	27.074	22.947
NLS p95	47.734	55.398	33.015

TABLE VII. Percentage of observations with any component error above the indicated threshold at 10% noise. N: augmented MLP; F: fitting. Thresholds are post-run descriptive diagnostics.

Samples	N > 10%	F > 10%	N > 50%	F > 50%
5001	4.36	0.00	0.00	0.00
1001	7.67	0.00	0.00	0.00
501	6.61	0.00	0.00	0.00
101	27.97	17.44	1.81	6.73

they do not establish whether decay, aliasing, or optimization caused the errors. They also do not identify a universally best observation duration.

D. Processing Cost Versus Time to First Estimate

The full-grid measurements give approximately 0.048 ms of CPU processing for the network and 6.926 ms for fitting, roughly 145 times. These measurements came from separate earlier experiment runs, not a contemporaneous benchmark. Both require 50 ms of observations, specifically illustrative acquisition-plus-processing times are about 50.05 and 56.93 ms, excluding sensor and communication overhead.

TABLE VIII. Measured CPU processing time versus observation duration. Neural entries are the minimum–maximum of three augmented-seed medians, each from 50 timed single requests. Fitting has no training seeds: each entry is the pooled median of 19,500 solve times (1,500 circuits in 13 conditions: clean once and four positive noise levels with three realizations each), not a median of condition medians. The 50 ms row reuses timings from the earlier noise (neural) and fitting experiments; only the 1–20 ms rows were measured in the window experiment.

Window (ms)	Neural (ms)	Fitting (ms)
1	0.022–0.022	1.504
2	0.023–0.023	1.667
5	0.025–0.026	2.306
10	0.028–0.028	2.962
20	0.033–0.034	3.979
50	0.048–0.048	6.926

Table VIII shows that shorter observations reduce time to first estimate (acquisition plus processing) more directly. For example, a 5 ms record gives approximate acquisition-plus-processing times of 5.026 ms for the network and 7.306 ms for fitting. Reducing density to 101 points over 50 ms lowers processing to approximately 0.023–0.024 ms and 1.623 ms, respectively, but does not shorten acquisition. Historical full-grid timings were not remeasured alongside these reduced-observation runs.

VI. Limitations and Reproducibility

The training simulator and fitting model describe the same ideal circuit family. There are no real measurements, the excitation is known, initial conditions are fixed, noise is artificial, and there is no anti-alias filtering. Float32 measurement representation does introduce ordinary floating-point rounding. The results

do not establish hardware robustness or performance under model mismatch.

The sampled distribution strongly favors underdamped circuits and gives little training coverage of the overdamped regime. Six overdamped test cases cannot support regime-generalization claims.Three training seeds and repeated noise realizations provide descriptive replication but do not provide population confidence intervals. The same test circuits are reused across experiment stages and the best observed window or grid must not be presented as a prospectively selected optimum.

A. Reproduction and Availability

Parameter generation and split shuffling used NumPy PCG64 seeds 20261010 and 20261011, respectively. The training seeds are 20261014–20261016; validation and test noise use separate roots 20261017 and 20261018. Recorded versions are Python 3.14.8, NumPy 2.5.3, SciPy 1.18.1, and PyTorch 2.14.1 on macOS 27.0 arm64. Training used the PyTorch MPS backend; seeds do not guarantee bit-identical MPS training.

VII. Conclusion

Direct neural regression offers low measured processing cost for series-RLC parameter estimation, and noise augmentation improves its in-distribution Gaussian-noise tolerance over matched clean-trained controls. Under dense measurements and exactly known ideal physics, bounded nonlinear fitting is much more accurate. With sparse measurements, five predetermined starts remove the single-start baseline's 52 clean cases above 10% error and reduce its noisy above-50% incidence from 6.73% to 3.40%, at a higher processing cost. The augmented network's corresponding 1.81% incidence remains lower, but its above-10% incidence is higher (27.97% versus 14.60%). Therefore we can explain a substantial part of the he historical tail through initialization. Five starts still do not establish the best achievable physical estimator. Observation duration, sampling density, error tails, and processing time therefore require separate consideration. The present evidence describes implementation-specific accuracy, speed, and outlier behavior, not a fundamental trade-off between neural and physical estimation. Hardware evaluation and stronger physical baselines are future work and are not demonstrated capabilities.

VIII. Video

https://youtu.be/QVXtu-RmGPA

References

[1] M. Forgione and D. Piga, "Continuous-time system identification with neural networks: Model structures and fitting criteria," *European Journal of Control*, vol. 59, pp. 69–81, 2021.

[2] C. M. Bishop, "Training with noise is equivalent to Tikhonov regularization," *Neural Computation*, vol. 7, no. 1, pp. 108–116, 1995.

[3] L. Ljung, *System Identification: Theory for the User*, 2nd ed. Prentice Hall, 1999.

[4] P. V. Overschee and B. D. Moor, *Subspace Identification for Linear Systems: Theory—Implementation—Applications*. Kluwer Academic Publishers, 1996.

[5] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015. [Online]. Available: https://arxiv.org/abs/1412.6980

[6] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.