A Practical Pipeline for DC Motor Identification with Dead-Zone, Signed Bias, and Small Delay

Piyush Tiwari
M.Tech (Control & Automation)
piyusht24@iitk.ac.in

***Abstract*—Identification of low-cost DC motors is frequently complicated by actuator nonlinearities, particularly when using common drivers such as the L298N that exhibit significant voltage drop and thermal drift. These effects manifest as a voltage dead-zone, asymmetric responses to command polarity, and a small but significant time delay, which cannot be captured by a single linear model. This experiment report presents a practical identification pipeline that models these phenomena as a compact cascade preceding a first-order discrete plant. The cascade consists of a static dead-zone, a fractional-sample delay, and signed biases to account for the asymmetry. Parameters are identified using a repeating staircase excitation signal designed to exercise the system across its full operating range. A two-stage (coarse-to-fine) grid search efficiently determines the delay and bias parameters by minimising the median per-step mean absolute error. The proposed method, executable in under seven minutes on a cloud platform (Google Colab), yields a high-fidelity model and exports controller-ready discrete parameters (a, b). The resulting model demonstrates a substantial reduction in prediction error compared to a conventional linear fit.**

I. Context and Motivation

The performance of a model-based controller is fundamentally limited by the accuracy of the plant model. For low-cost mechatronic systems, datasheet parameters are often unreliable, necessitating empirical system identification. This work focuses on a common laboratory setup: a geared DC motor driven by an L298N H-bridge module, a component known for its non-ideal behaviour.

An initial investigation revealed a significant voltage shortfall: despite a regulated 12 V power supply, the voltage measured at the motor terminals was consistently limited to 8.8 V to 9.1 V, with noticeable drift corresponding to the driver's operating temperature. This power delivery issue creates a fragile operating environment, especially near zero speed, and gives rise to a trio of challenging nonlinear effects that must be addressed for accurate modelling:

1) **A voltage dead-zone,** where small input commands produce no physical response.
2) **Polarity asymmetry,** where the motor responds differently to forward and reverse commands of the same magnitude.
3) **A short transport delay** between the voltage command and the measured change in speed.

A conventional Linear Time-Invariant (LTI) model, such as a single transfer function, is incapable of simultaneously representing this combination of dead-zone, delay, and direction-dependent dynamics. The central premise is that these actuator-centric nonlinearities must be identified and compensated first. By modelling them explicitly, a simple, low-order linear plant can describe the system's underlying dynamics, enabling effective controller design.

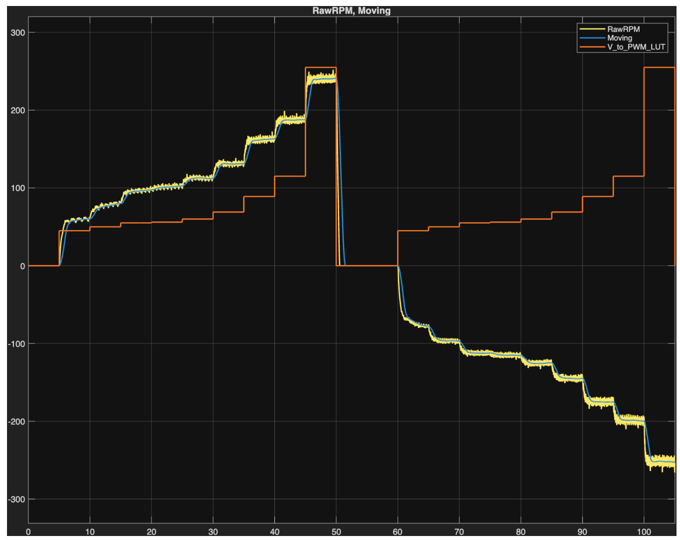

Fig. 1. Designed staircase via the PWM-to-V lookup (orange), raw RPM (yellow), and smoothed RPM (blue). The repeating sequence exercises both polarities and multiple magnitudes.

II. System Setup, Automation, and Data Path

The automation pipeline originates in Simulink. Both the simulation and hardware-in-the-loop models were maintained in a single workspace. Raw signals (PWM command, mapped voltage, and RPM) were exported through `To Workspace` blocks in *structure-with-time* format. A short MATLAB script then wrote arrays to a timestamped folder containing CSV files with `time [s]`, `voltage [V]`, and `rpm`. These records were consumed by a Python notebook for preprocessing, step detection, model search, and export of deployment-ready parameters; see Fig. 1.

III. Hardware and Data at a Glance

Motor: 12 V, 300 RPM with gearbox (about 60:1).
Driver: L298N dual H-bridge (notable internal drop; effective supply 8.8 V to 9.1 V, temperature sensitive).
Sensor: Optical encoder 2400 CPR (post-quadrature).
Controller/DAQ: Arduino Mega; logging via Simulink/MATLAB to CSV.

TABLE I
DESIGNED REPEATING-SEQUENCE STAIRCASE (COMMAND VOLTAGE VS. TIME).

Seg.	Start t [s]	V [V]	Dur. [s]
1	0	0.00	5
2	5	3.56	5
3	10	4.15	5
4	15	4.62	5
5	20	4.75	5
6	25	5.20	5
7	30	5.70	5
8	35	6.50	5
9	40	7.25	5
10	45	8.81	5
11	50	0.00	5
12	55	0.00	5
13	60	–3.56	5
14	65	–4.15	5
15	70	–4.62	5
16	75	–4.75	5
17	80	–5.20	5
18	85	–5.70	5
19	90	–6.50	5
20	95	–7.25	5
21	100	–8.81	5
22	105	0.00	0

Supply and Logic: 12 V adapter; driver logic powered from Arduino 5 V with the onboard jumper off.
Sampling time: $T_s = 0.01\,\mathrm{s}$ throughout.
Electrical parameters: $R \approx 1.371\,\Omega$, $L \approx 0.0003\,\mathrm{H}$ giving $L/R \approx 0.22\,\mathrm{ms}$. An identified mechanical time constant around 0.2 s to 0.3 s supports the first-order plant assumption.

IV. Excitation Signal Design

A repeated-sequence staircase ($0 \to +V_{\max}$, back to 0, then to $-V_{\max}$) was intentionally used. Multiple step magnitudes in both directions improve robustness: a first-order model that fits a single step may not generalise across steps and polarities, whereas a staircase exposes biases, delay, and dead-zone consistently near reversals. Concentrating samples around the sign change further sharpens sensitivity to near-zero effects.

V. Voltage–PWM Mapping

A consistent notion of "one volt" was enforced between model and hardware. From approximately 150 calibration pairs (measured driver output vs. PWM), a *monotone* PCHIP map was created in Python and its inverse was applied to all commands during analysis. For deployment in Simulink, a lookup table with the same calibration data was used so that the hardware used exactly the recorded points. Although PCHIP exactly interpolates the calibration pairs, a slightly different plot-only dead-zone appears near zero; this is explained by voltmeter sensitivity and the physical ±3.5 V threshold was retained (see Fig. 2).

VI. Preprocessing and Step Detection

Three smoothers were compared on identical RPM traces: a moving average (window 50), a default Simulink low-pass,

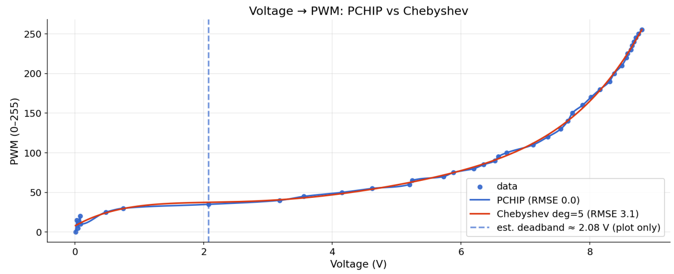

Fig. 2. Voltage-to-PWM calibration: PCHIP (exact through data) versus a degree-5 Chebyshev fit. The dashed line marks a plot-only deadband estimate that differs from the physical ±3.5 V; meter sensitivity near zero explains the difference.

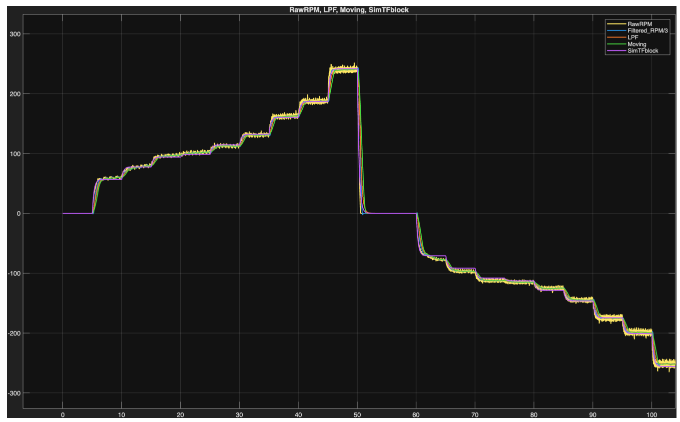

Fig. 3. Raw RPM (yellow) with multiple smoothers. The offline Savitzky–Golay shape is emulated online by a short discrete FIR, aligning real-time processing with post-analysis without prohibitive memory cost.

and Savitzky–Golay (odd window, polynomial 2–3). Savitzky–Golay preserved step edges best and minimised downstream identification error in offline analysis. To meet real-time constraints, a short discrete FIR was tuned in Simulink/Arduino so that its magnitude/phase over the relevant band emulated the Savitzky–Golay result while remaining lightweight due to Arduino memory limits (see Fig. 3). Steps in the mapped input were detected by a *median absolute deviation (MAD)*-based threshold with minimum amplitude and dwell, which segmented the staircase into positive and negative step windows used throughout scoring.

VII. Model Structure

Let $u[k]$ denote the commanded voltage after PWM-to-V mapping. A cascade of static nonlinearity and delay feeds a first-order discrete plant. **(a) Dead-zone**; $D > 0$:

$$\tilde{u}[k] = \begin{cases} 0, & |u[k]| \le D, \\ \operatorname{sgn}(u[k])\big(|u[k]| - D\big), & |u[k]| > D. \end{cases} \qquad (1)$$

(b) Small delay (fractional samples) with $n = \lfloor \theta / T_s \rfloor$ and $f = \theta / T_s - n$:

$$u_{\mathrm{d}}[k] = (1 - f)\,\tilde{u}[k-n] + f\,\tilde{u}[k-n-1]. \qquad (2)$$

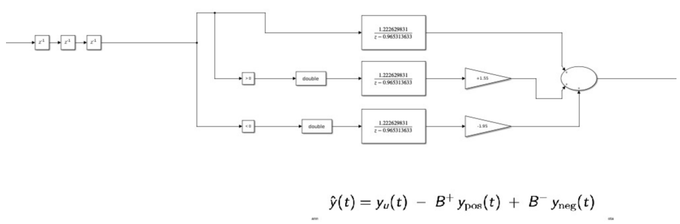

Fig. 4. Implementation: three-sample integer delay with fractional blend, sign-gated biases ($B^{+} = -1.55\,\mathrm{V}$, $B^{-} = -1.95\,\mathrm{V}$), and the discrete first-order block with $(a, b) = (0.965314, 1.222630)$.

(c) Signed bias to capture polarity asymmetry:

$$\hat{u}[k] = \begin{cases} u_{\mathrm{d}}[k] + B^{+}, & u_{\mathrm{d}}[k] > 0, \\ u_{\mathrm{d}}[k] + B^{-}, & u_{\mathrm{d}}[k] < 0, \\ 0, & u_{\mathrm{d}}[k] = 0. \end{cases} \tag{3}$$

Note (notation, paper $\leftrightarrow$ notebook). Symbols match the Python notebook names in brackets: $u[k]$ (`u`) is the mapped command in volts; $\tilde{u}[k]$ (`u_eff`) is the effective input after the dead-zone; $u_{\mathrm{d}}[k]$ (`u_del`) is the delayed version of $\tilde{u}$; $\hat{u}[k]$ (`u_hat`) is the bias-adjusted input that drives the plant; $y[k]$ (`y`/`yhat`) is the RPM output. Scalars: T_s (`Ts`) sampling time; θ (`theta`) total delay; D dead-zone threshold; B^{+} (`Bp`) and B^{-} (`Bn`) signed biases. The discrete plant uses (a, b) with $a = e^{-T_s/\tau}$ and $b = K(1 - a)$.

Note (fractional-sample delay, weight blending). When θ is not an integer multiple of T_s, $n = \lfloor \theta/T_s \rfloor$ and $f = \theta/T_s - n \in [0, 1)$. Define weights $w_0 = 1 - f$, $w_1 = f$ (so $w_0, w_1 \geq 0$ and $w_0 + w_1 = 1$), then

$$\begin{aligned} u_{\mathrm{d}}[k] &= w_0\, \tilde{u}[k-n] + w_1\, \tilde{u}[k-n-1], \\ (w_0, w_1) &= (1 - f,\ f). \end{aligned} \tag{4}$$

This is the linear interpolation used in the notebook: `u_del = w0*u_eff[k-n] + w1*u_eff[k-n-1]`. *Example (final parameters):* $T_s = 0.01\,\mathrm{s}$, $\theta = 31.25\,\mathrm{ms} \Rightarrow n = 3,\ f = 0.125$, so $u_{\mathrm{d}}[k] = 0.875\, \tilde{u}[k-3] + 0.125\, \tilde{u}[k-4]$.

This arrangement allows a single first-order dynamic element to remain valid in both directions once small biases and delay are absorbed up front; the Simulink realisation is shown in Fig. 4. The discrete (a, b) form is used for deployment because:

1) exact ZOH discretisation ensures the simulation matches the microcontroller implementation sample-for-sample;
2) numerical conditioning improves by avoiding repeated evaluation of $e^{-T_s/\tau}$ on embedded hardware; and
3) Simulink blocks and generated code interface cleanly with (a, b), reducing parameter drift between analysis and deployment.

VIII. First-Order Step Fits

From per-step first-order fits before applying signed biases, median parameters were obtained as

$$K \approx 35.18\,\mathrm{RPM/V}, \quad \tau \approx 0.300\,\mathrm{s},$$

with median $R^2 \approx 0.974$ (median step MAE $\approx 0.53\,\mathrm{RPM}$). These values justify a single first-order plant once signed bias handles asymmetry.

IX. Search Algorithm for Delay and Signed Biases

A two-stage grid search was performed around a fixed dead-zone D and initial (K, τ) from step fits. Candidates were scored by the *median per-step MAE* across the entire staircase run.

A. Algorithm Outline

1) **Data preparation:** apply the inverse voltage map to PWM; smooth RPM (Savitzky–Golay offline / FIR online); detect step windows via MAD with amplitude/dwell guards.
2) **Plant prefit:** obtain (K, τ) by first-order fits over many steps; fix D from a voltage sweep around ±3.5 V.
3) **Coarse search:** sweep $\theta \in [0,\ 0.30]\,\mathrm{s}$ and $(B^{+}, B^{-}) \in [-5,\ 5]\,\mathrm{V}$ on a coarse grid; simulate the cascade and compute MAE per step for each triple; retain the top-K by median MAE.
4) **Fine search:** centre a denser grid around the retained triples; rescore and select the winner by median MAE (report interquartile range as dispersion).
5) **Export:** compute (a, b) at $T_s = 0.01\,\mathrm{s}$ and fractional-delay taps $(w_0, w_1) = (1 - f, f)$ with $f = \theta/T_s - n$.

B. Reproducible Pseudocode (vectorised)

```
INPUTS: t, pwm_cmd, rpm; Ts = 0.01
CALIB: invert_map = PCHIP(about 150 pairs);
    fixed dead-zone D approx 3.5 V
OUTPUTS: a, b, Bp, Bn, theta, n, f, (w0, w1)

# Map, smooth, and segment
u = invert_map(pwm_cmd) # volts
y = smooth(rpm) # SG offline / FIR online
S = detect_steps(u, method="MAD") # [ (i0,i1),
     ... ]

# Baseline FO plant
K, tau = fit_many_steps(u, y, S)

# Two-stage search (vectorised grids)
best = []
for (theta, Bp, Bn) in GRID(stage="coarse"):
   yhat = simulate(u, D, theta, Bp, Bn, K, tau,
        Ts)
   s = mae_per_step(y, yhat, S) # vector
   best = keep_topK(best, (theta,Bp,Bn),
       median(s), IQR(s))

for (theta, Bp, Bn) in GRID(center=best, stage
    ="fine"):
   yhat = simulate(u, D, theta, Bp, Bn, K, tau,
        Ts)
   s = mae_per_step(y, yhat, S)
   best = keep_topK(best, (theta,Bp,Bn),
       median(s), IQR(s))

# Export discrete params
n = floor(theta/Ts); f = theta/Ts - n
```

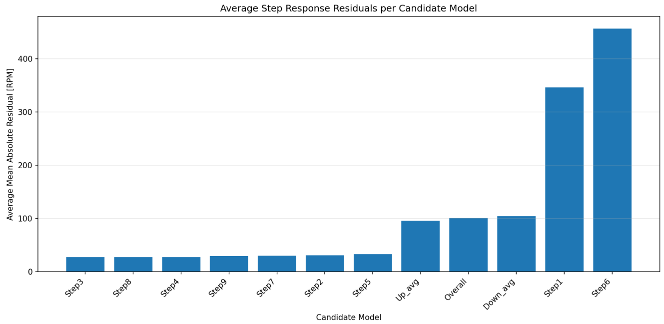

Fig. 5. Average per-step absolute residual per candidate model (lower is better). Consistency across steps is revealed by this view.

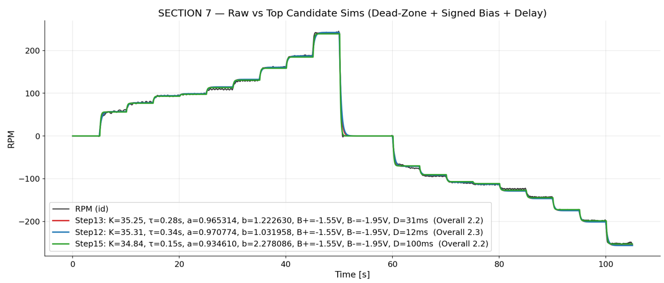

Fig. 6. Full-run overlays of raw data and the top candidates. Reversals expose delay differences most clearly; selection is by median per-step MAE (see also Fig. 5).

```
w0, w1 = 1-f, f
a = exp(-Ts/tau); b = K*(1-a)
```

X. Final Run: Selected Candidate and Leaderboard

Selected candidate (Step13):

- $K = 35.248$, $\tau = 0.283\,\mathrm{s}$, $a = 0.965\,314$, $b = 1.222\,630$ (ZOH, $T_s = 0.01\,\mathrm{s}$)
- $B^+ = -1.55\,\mathrm{V}$, $B^- = -1.95\,\mathrm{V}$, $\theta = 31.25\,\mathrm{ms} \Rightarrow n = 3, f = 0.125, (w_0, w_1) = (0.875, 0.125)$
- Dead-zone: $D^+ = +3.50\,\mathrm{V}$, $D^- = -3.50\,\mathrm{V}$
- Overall MAE: **2.209 RPM**

XI. Validation Visuals

Most residuals lie within a ± 2 MAD band (1.6 RPM); spikes at reversals and one larger negative excursion around 50 s to 52 s are consistent with brief unmodelled effects (current saturation together with encoder-edge artefacts). See Fig. 9.

XII. Contributions

- A generalisable search-based identification procedure (coarse-to-fine) that locates delay and signed biases quickly (6 min to 7 min on Colab) using median per-step MAE over a staircase run.
- A compact cascade (dead-zone, fractional delay, signed bias, first-order plant) that captures low-speed behaviour on low-cost hardware without invoking named block-models.

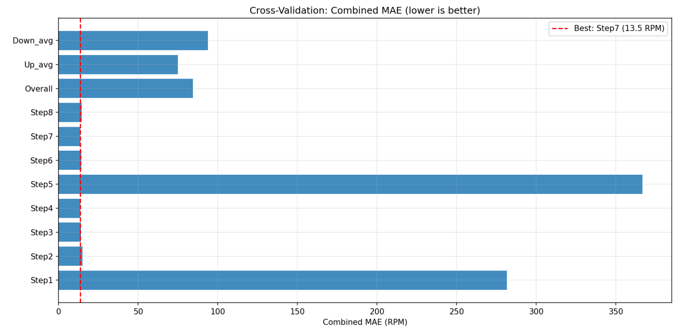

Fig. 7. Cross-validation combined MAE across candidates (lower is better). The dashed marker indicates the winner.

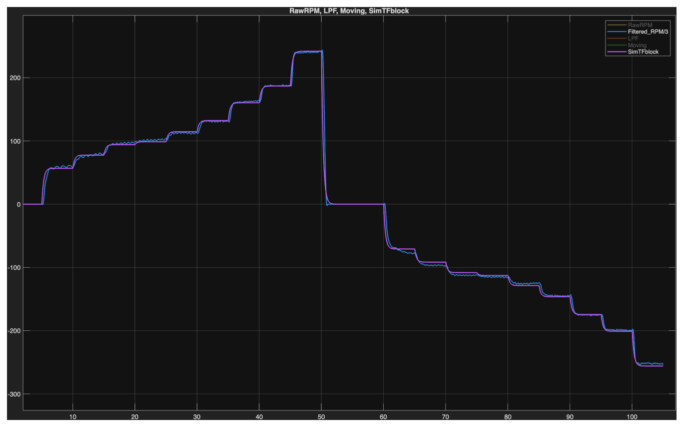

Fig. 8. Final Simulink run with the selected parameters. Close alignment is obtained throughout with small localised deviations at sign changes.

- Plug-and-play export for Simulink/Arduino, providing $(a, b, D^+, D^-, B^+, B^-)$ and θ, and delay taps (w_0, w_1), plus a lookup-based PWM map.
- Evidence via overlays, residuals, and sensitivity that the approach remains accurate and practical under measurement noise and temperature drift of the driver.

XIII. Limitations and Next Steps

The dead-zone was treated as fixed, although it may drift with temperature/load and with ageing. A single (K, τ) was used for both directions; if residuals show polarity-dependent dynamics, splitting the plant into (K^+, τ^+) and (K^-, τ^-) may be explored. The Savitzky–Golay surrogate FIR may slightly bias very fast transients; results with raw RPM can be reported or the tach filter can be modelled explicitly.

Next steps: (i) adaptively re-estimate the dead-zone at each run by a short voltage sweep around ±3.5 V; (ii) re-calibrate the PWM-to-V lookup whenever the measured V_{out} at PWM$= 255$ differs from historical values; (iii) expose a one-click routine in the Python notebook to regenerate (a, b, θ, B^+, B^-); and (iv) design and validate a PI controller around the FO+delay plant using the exported parameters.

TABLE II
CROSS-VALIDATION LEADERBOARD (SORTED BY OVERALL MAE).

Candidate	K	τ [s]	B^{+} [V]	B^{-} [V]	θ [s]	MAE [RPM]	Mean Step MAE [RPM]
Step13	35.248	0.283	-1.550	-1.950	0.031	2.209	16.202
Step15	34.838	0.148	-1.550	-1.950	0.100	2.225	16.363
Step7	35.178	0.149	-1.550	-1.950	0.100	2.284	16.306
Step12	35.309	0.337	-1.550	-1.950	0.012	2.287	16.290
Step6	33.252	0.300	-1.725	-2.175	0.042	2.622	17.098
Step2	32.814	0.576	-1.800	-2.275	0.000	3.725	18.229
Step5	38.242	0.300	-1.250	-1.575	0.012	4.289	16.242
Step10	31.072	0.366	-1.950	-2.425	0.021	4.460	18.659
Step4	40.316	0.368	-1.025	-1.400	0.000	6.396	16.512
Step14	27.510	0.285	-2.350	-2.900	0.071	7.838	21.023
Step8	47.680	0.276	-0.625	-0.925	0.019	13.941	18.495
Step11	18.073	0.279	-4.375	-5.150	0.113	17.509	29.145

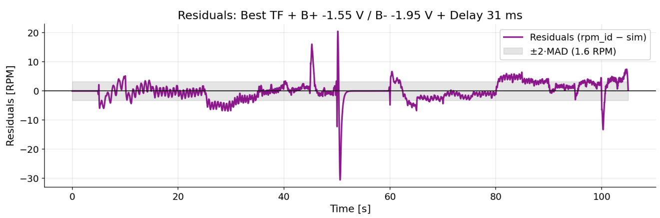

Fig. 9. Residuals for the selected model with a ± 2 MAD band. Outliers are concentrated near reversal instants; elsewhere the error remains confined to a narrow band.

XIV. REPRODUCIBILITY AND ARTEFACTS

Repository (notebooks, scripts, outputs): https://github.com/Piyushiitk24/dc-motor-signed-bias-deadzone-id-control.

Inputs: CSV logs `time [s]`, `voltage [V]` (mapped), `rpm`.

Rapid re-identification (field workflow):

1) Measure V_{out} at PWM$=255$ (full duty) at the driver terminals; update the lookup if it has changed.
2) Export `time`, `voltage`, and `rpm` via `To Workspace` blocks; the MATLAB script writes a timestamped folder.
3) Open the notebook in the repository and *Run All*. Parameters $(a, b, \theta, B^{+}, B^{-})$ and delay taps are regenerated in 6 min to 7 min with T_s fixed at 0.01 s.