MuJoCo DC Motor Model

Google DeepMind

We review DC motors and describe MuJoCo's `dcmotor` actuator. The equations are derived for brushed motors but apply equally to brushless ones, where electronic commutation reduces to an equivalent circuit.

1 Background

We use SI units throughout, but any coherent system of units applies. We assume motion is rotational; for linear motion replace radians with meters as required.

1.1 Electromagnetic Model

The key electro-mechanical variables are

Symbol	Description	Units
v	Applied voltage	Volt
i	Current	Ampere
ω	Angular velocity	radian/second
τ	Output torque	Newton · meter

and the key constants are

Symbol	Description	Units
R	Resistance	Ohm
K_t	Torque constant	Newton · meter/Ampere
K_e	Back-EMF constant	Volt · second/radian

The quasi-static model [1, 2, 3] assumes instantaneous electrical dynamics: current and torque are direct functions of voltage and velocity. The constitutive equations are the voltage balance (1a) and the torque law (1b):

$$v = i\,R + K_e\,\omega \tag{1a}$$
$$\tau = K_t i \tag{1b}$$

Solving for current and substituting, we have

$$\tau = \frac{K_t}{R}(v - K_e\,\omega) \tag{2}$$

Output torque is proportional to the difference between applied and back-EMF voltage $v_{\text{back}} = K_e\,\omega$ (Figure 1).

Electrical constants. Fundamentally, both K_t and K_e arise from the same physical quantity: the magnetic flux Φ of the coil. Faraday's law gives $v_{\text{back}} = \Phi\,\omega$ and the Lorentz force gives $\tau = \Phi\,i$, so in SI units:

$$K_e = K_t$$

This can also be seen from energy conservation: $P_e = i\,(K_e\,\omega) = (K_t\,i)\,\omega = P_m$.

Note the dimensions match:

$$\frac{\text{Volt}}{\text{radian/second}} = \frac{\text{Joule}}{\text{Coulomb/second}} = \frac{\text{Newton}\cdot\text{meter}}{\text{Ampere}}$$

If these constants are the same, why have both? Two reasons. First, datasheets typically use mixed units (K_e in RPM/V, K_t in mN·m/A), giving different values for the same physical quantity. Second, K_t and K_e are measured differently: K_t by locking the rotor and measuring torque per Ampere; K_e by spinning the rotor and measuring open-circuit Volts per radian/second. In the first case, high currents can lead to magnetic field saturation in the core, causing the effective K_t to drop below K_e. The equality $K_e = K_t$ thus assumes Φ independent of i. We make this assumption for now and use a single motor constant $K \equiv K_t = K_e$ throughout the remainder of this document and internally in MuJoCo, but see note at end of §2.1.

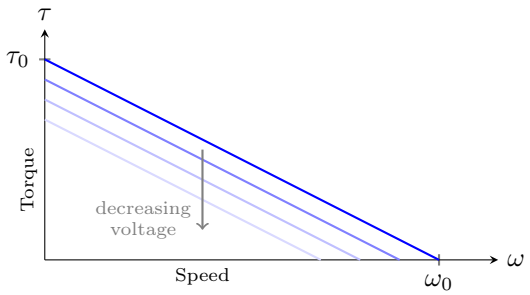


Figure 1: Torque-speed relationship (2) at fixed voltage. As voltage decreases, the maximum torque and speed decrease linearly.

Current Saturation. A maximum current rating $i_{\max}$ limits the output torque:

$$\tau = \text{clip}\left(\frac{K}{R}(v - K\,\omega),\ \pm K\,i_{\max}\right) \tag{3}$$

where the maximum torque $\tau_{\max} = K\,i_{\max}$. The feasible torque-speed envelope forms a parallelogram:

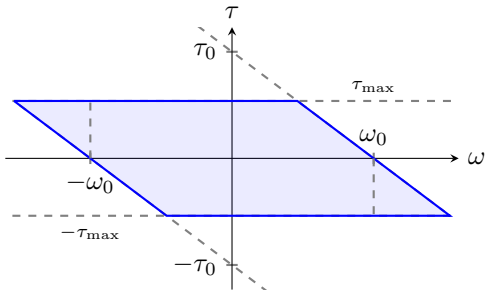


Figure 2: Torque-speed envelope with current saturation (3).

Note that datasheets typically distinguish two current limits. The *continuous* (or *nominal*) current $i_{\max}$ is the thermal limit: the maximum current the motor can sustain indefinitely without exceeding its maximum winding temperature. The *peak* current i_{peak} is a higher short-term limit, typically 5–10× the continuous value, constrained by demagnetization or commutation limits.

Symbol	Description	Condition	Formula/note
τ_0	Stall Torque	$\omega = 0$	$\tau_0 = Kv/R$
ω_0	No-Load Speed	$\tau_{\text{load}} = 0$	$\omega_0 \approx v/K$
$\partial\omega/\partial\tau$	Gradient	Slope	$-R/K^2$
$i_{\max}$	Maximum Current	Limit	Thermal limit
$\tau_{\max}$	Maximum Torque	Limit	$\tau_{\max} = K\, i_{\max}$

Table 1: Named constants derived from the motor equations.

1.1.1 Inductance

Including the effects of winding inductance L (Henry) means treating the current i as a state variable:

$$v = L\,\frac{di}{dt} + i\,R + K\,\omega \tag{4}$$

The change in current is proportional to the voltage and negatively proportional to both the instantaneous current and the rotation velocity. The time constant of this ODE is $t_e = L/R$. If $t_e \ll \Delta t$ (the simulation timestep), the current equilibrates within a single step and the quasi-static approximation (2) is adequate.

Motor drivers often impose a hard limit on di/dt to protect windings and electronics, bounding the torque ramp rate to $K \cdot (di/dt)_{\max}$.

1.2 Mechanical Model

Several purely mechanical phenomena affect the motor's behavior and delivered torque, warping the electromagnetic performance envelope.

Mechanical losses. These reduce the net torque available at the shaft: $\tau_{\text{net}} = \tau_{\text{elec}} - \tau_{\text{loss}}$.

- **Coulomb:** Constant torque opposing rotation (dry friction). $\tau_{\text{loss}} = \tau_c\,\text{sgn}(\omega)$. Discontinuous; already available in MuJoCo as `frictionloss`.
- **Viscous:** Drag is a smooth function of speed $\tau_{\text{loss}} = b(\omega)$. The simplest model is linear, with drag proportional to speed $\tau_{\text{loss}} = B\,\omega$, but higher order terms may be needed for higher-fidelity models e.g., $\tau_{\text{loss}} = B_1\,\omega + B_2\,\omega|\omega| + B_3\,\omega^3 + \ldots$.

Datasheets report the *no-load current* i_0: the current drawn when spinning freely at no-load speed ω_0. At steady state, the electromagnetic torque balances all mechanical losses:

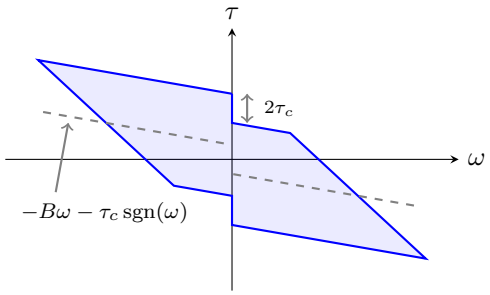
$$K\,i_0 = \tau_c + B\,\omega_0 \tag{5}$$

This provides one constraint on two unknowns (τ_c and B). Without additional data, the user must either assume one dominates or obtain friction measurements at multiple speeds. In MuJoCo terms, τ_c maps to `frictionloss` and B to `damping`.

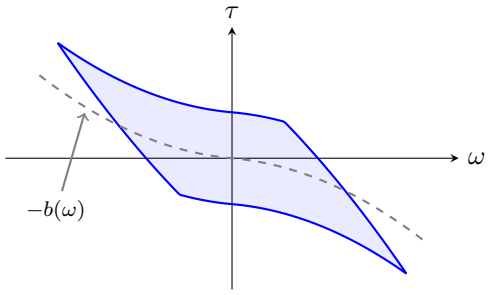
Combining current saturation with both mechanical losses, the net torque is:

$$\tau_{\text{net}} = \text{clip}\left(\frac{K}{R}(v - K\,\omega),\ \pm K\,i_{\max}\right) - B\,\omega - \tau_c\,\text{sgn}(\omega)$$

The clipping applies to the electrical torque (current limit), while both friction terms are mechanical losses applied to the output post-clipping. Viscous drag $-B\omega$ tilts the envelope; Coulomb friction $-\tau_c\,\text{sgn}(\omega)$ shifts the right half ($\omega > 0$) down and the left half ($\omega < 0$) up, creating a $2\tau_c$ discontinuity at $\omega = 0$ (Figure 3a).



(a) Linear viscous drag and Coulomb friction.



(b) Nonlinear viscous drag $b(\omega) = B_1\omega + B_2\omega|\omega|$, no friction.

Figure 3: Torque-speed envelopes with mechanical losses. The dashed gray line shows the drag function; the shaded region is the achievable torque at each speed. Note that datasheet torque-speed curves typically plot the first quadrant only.

Rotor Inertia and Gearing. Every DC motor datasheet lists the rotor inertia J_r (kg·m^2). When a gear train with ratio N is attached, the effective inertia reflected to the output shaft is $J_{\text{eff}} = J_r N^2$ [4]. Note that real gearboxes also introduce efficiency losses (typically 70–90%), which reduce the transmitted torque by a multiplicative factor η, approximated by effectively reducing the motor constant $K_{\text{eff}} = \eta K$.

Cogging Torque. Brushless DC motors exhibit *cogging torque*: a position-dependent torque ripple caused by the interaction between permanent magnets and stator slots. It can be modeled as a periodic bias:

$$\tau_{\text{cog}}(\theta) = A\sin(N_p\,\theta + \phi) \tag{6}$$

where A is the amplitude, N_p is the number of pole pairs times the number of slots per pole, and ϕ is a phase offset. Cogging torque is significant primarily at low speeds. Datasheets sometimes report peak cogging as a percentage of rated torque (typically 1–5%).

Symbol	Description	Formula / Note
τ_c	Coulomb friction	$\tau_c \operatorname{sgn}(\omega)$
B	Viscous drag (linear)	$-B\,\omega$
ω_0	No-load speed	$\omega_0 = v\,K/(K^2 + R\,B)$
J_r	Rotor inertia	units: kg·m^2
N	Gear ratio	$J_{\text{eff}} = J_r N^2$
η	Gearbox efficiency	$K' = \eta\,K$
A	Cogging amplitude	$\tau_{\text{cog}} = A\sin(N_p\theta + \phi)$
N_p	Cogging periodicity	poles × slots/pole
ϕ	Cogging phase	offset

Table 2: Named constants related to mechanical properties. Unlike in Table 1, the non-approximate expression for ω_0 takes into account the linear drag B (assuming no high-order terms).

Backlash. Gearboxes introduce backlash: a small angular deadband where the motor can turn without moving the output shaft. Datasheets report this in arcminutes. MuJoCo supports backlash modeling via a dual-joint decomposition; see here for details.

1.3 Thermal Model

Winding temperature affects motor performance primarily through increased copper resistance, and can be modeled as a single lumped thermal state. The thermal constants are

Symbol	Description	Units
R_T	Thermal resistance	Kelvin/Watt
C	Thermal capacitance	Joule/Kelvin
$t_T = R_T C$	Thermal time constant	second
α	Resistance temp. coefficient	1/Kelvin
T_0	Reference temperature	degree Celsius
T_a	Ambient temperature	degree Celsius

Table 3: Thermal model constants. Units involving temperature differences use Kelvin (equivalent to Celsius for differences); absolute temperatures use degree Celsius, following datasheet convention.

Note that some manufacturers specify two thermal resistances: R_{th1} (winding-to-housing) and R_{th2} (housing-to-ambient), which sum to give the total winding-to-ambient thermal resistance $R_T = R_{\text{th1}} + R_{\text{th2}}$. The single-node model above uses R_T directly.

1.3.1 Lumped Thermal ODE

The winding temperature T evolves according to a first-order lumped model driven by the power dissipation P (Watt, detailed in §1.3.2):

$$\frac{\partial T}{\partial t} = \frac{1}{C}P - \frac{T - T_a}{t_T} \tag{7}$$

where $t_T = R_T C$ is the thermal time constant. This produces exponential rise/decay toward a steady-state temperature $T_{ss} = T_a + R_T P$ (Figure 4).

1.3.2 Losses

The dominant loss is copper (Joule) heating:

$$P = i^2 R(T)$$

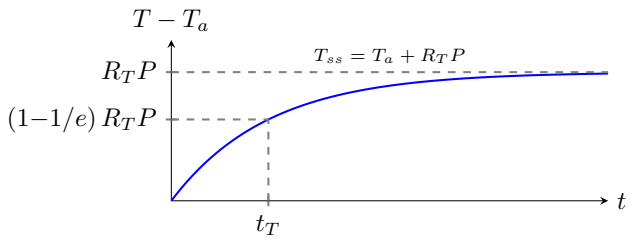


Figure 4: Temperature rise under constant power dissipation P. At $t = t_T$, it reaches $(1 - 1/e) \approx 63\%$ of its steady-state value.

Optionally, speed-dependent iron losses (eddy-current and hysteresis losses in the stator laminations) can be included [1]:

$$P = i^2 R(T) + K_{\text{fe}}\omega^2$$

The iron loss coefficient K_{fe} is not typically listed on datasheets and must be identified from efficiency curves or manufacturer simulation tools. For most hobby and robotics motors, iron losses are small compared to copper losses and can be neglected. They become significant at high speeds in large industrial motors.

1.3.3 Temperature-Dependent Resistance

Copper resistance increases approximately linearly with temperature:

$$R(T) = R_0\,(1 + \alpha(T - T_0)) \tag{8}$$

where R_0 is resistance at reference temperature T_0 and $\alpha \approx 0.0039\,\text{K}^{-1}$ for copper. This is the dominant thermal feedback: as T rises, R increases, so for a given voltage the current $i = (v - K\omega)/R(T)$ drops, reducing torque.

Note that $R(T)$ also increases heating for a given current ($P = i^2 R(T)$), creating mild positive feedback under current control.

1.3.4 Magnet Flux Derating

Permanent magnet flux weakens with temperature, reducing K:

$$K(T) = K_0\,(1 + \alpha_m(T - T_0))$$

with $\alpha_m < 0$ (motor-dependent). This effect is often small over normal operating ranges and can be ignored.

1.3.5 Thermal Derating

Real actuators limit current as winding temperature approaches a maximum:

$$i_{\max}(T) = \begin{cases} i_{\text{rated}} & T \le T_1 \\ i_{\text{safe}} + (i_{\text{rated}} - i_{\text{safe}})\,s(T) & T_1 < T < T_2 \\ i_{\text{safe}} & T \ge T_2 \end{cases}$$

where $s(T)$ is a smooth interpolant between T_1 and T_2. This reduces the maximum available torque as the motor heats up.

1.4 Micro-Friction Models

Simple macroscopic friction models (Coulomb, viscous) cannot capture complex mechanical phenomena common in real motors with gear trains, such as pre-sliding hysteresis and stick-slip limit cycles. To capture these behaviors, a richer dynamic model is required. At the microscopic level, two surfaces in contact touch at many asperities which deform elastically under tangential load. This can be modeled as an average bristle deflection z, governed by a first-order ODE driven by the relative velocity ω. Friction torque is then a function of z, $\dot{z}$, and ω.

1.4.1 Dahl Model

The simplest stateful model [5] treats friction as a rate-independent hysteresis operator derived from the stress-strain curve:

$$\dot{z} = \omega - \frac{\sigma_0}{\tau_c}|\omega|\, z$$

with output $\tau = -\sigma_0 z$. In steady state ($\dot{z} = 0$), $z_{ss} = \tau_c \operatorname{sgn}(\omega)/\sigma_0$ so $\tau_{ss} = -\tau_c \operatorname{sgn}(\omega)$: pure Coulomb friction opposing motion. The two parameters are the bristle stiffness σ_0 (torque/radian) and the Coulomb friction torque τ_c. For small displacements the model is approximately linear ($\tau \approx -\sigma_0\theta$), giving spring-like pre-sliding behavior with hysteresis during direction reversals (Figure 5). The Dahl model does not capture the Stribeck effect [6] (the drop in friction at low velocity) and thus cannot predict stick-slip motion.

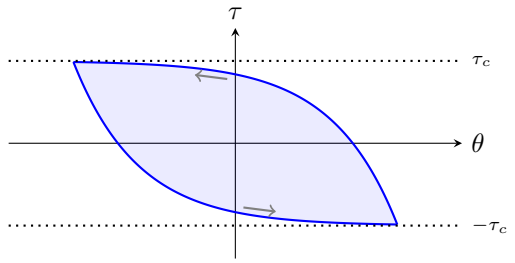


Figure 5: Hysteresis loop: friction torque τ vs. displacement θ under slow periodic loading (Dahl model). The loop area represents energy dissipated per cycle. Unlike memoryless Coulomb friction, the torque is continuous.

1.4.2 LuGre Model

The LuGre model [7, 8] extends Dahl by making the bristle saturation velocity-dependent and adding micro-damping and viscous terms:

$$\dot{z} = \omega - \sigma_0 \frac{|\omega|}{g(\omega)} z \tag{9a}$$

$$\tau = -(\sigma_0 z + \sigma_1 \dot{z} + \sigma_2 \omega) \tag{9b}$$

where the function $g(\omega)$ captures the Stribeck effect:

$$g(\omega) = \tau_c + (\tau_s - \tau_c)\, e^{-(\omega/\omega_s)^\gamma} \tag{10}$$

with exponent γ typically 1 or 2. In steady state, $\tau_{ss}(\omega) = -g(\omega)\operatorname{sgn}(\omega) - \sigma_2\omega$: the classic Stribeck curve (Figure 6).

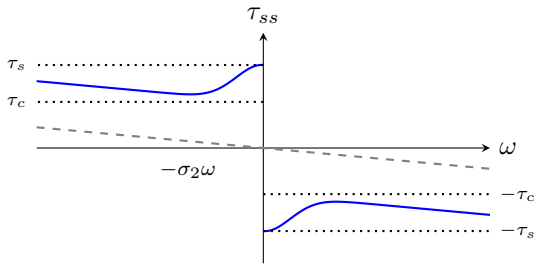


Figure 6: Steady-state friction $\tau_{ss}(\omega) = -g(\omega)\operatorname{sgn}(\omega) - \sigma_2\omega$. Stiction torque τ_s at $\omega = 0$ drops to Coulomb level τ_c over velocity scale ω_s (Stribeck effect). Friction opposes motion.

The Dahl model is recovered by setting $g(\omega) = \tau_c$ and $\sigma_1 = \sigma_2 = 0$. Linearizing around $\omega = z = 0$ gives second-order dynamics $J\ddot{\theta} - (\sigma_1 + \sigma_2)\dot{\theta} - \sigma_0\theta = \tau$ (applied torque): a spring-damper with natural frequency $\omega_n = \sqrt{\sigma_0/J}$, critically damped when $\sigma_1 = 2\sqrt{J\sigma_0}$.

The LuGre model can be shown to be input-strictly-passive (the map $\omega \mapsto -\tau$ dissipates energy) provided $\sigma_2 > \sigma_1(\tau_s - \tau_c)/\tau_c$. This passivity condition limits σ_1 and can lead to underdamped micro-dynamics, motivating the following extension.

1.4.3 Velocity-Dependent Damping

The passivity constraint on σ_1 can be relaxed by making the micro-damping decrease with velocity:

$$\sigma_1(\omega) = \bar{\sigma}_1\, e^{-(\omega/\omega_s)^\beta}$$

This allows large damping in the stiction regime (good for numerical stability and physical fidelity) while satisfying passivity at higher velocities where $\sigma_1 \to 0$.

Together with τ_c from §1.2, the LuGre model with velocity-dependent damping adds six parameters:

Symbol	Description	Units
σ_0	Bristle stiffness, pre-sliding slope	N·m/rad
$\bar{\sigma}_1$	Peak bristle damping at $\omega = 0$	N·m·s/rad
σ_2	Viscous damping coefficient	N·m·s/rad
τ_s	Stiction torque, $\tau_s \geq \tau_c$	N·m
ω_s	Stribeck velocity	rad/s
β	Damping decay exponent	dimensionless

Table 4: Parameters of the LuGre friction model (§1.4.2–1.4.3).

2 Implementation

Here we describe MuJoCo's `dcmotor` actuator. Some scalars are grouped into vectors; we use a colon to denote such scalar sub-attributes, e.g. `cogging:phase` refers to the third element of the `cogging` attribute (see Tables 5 and 8).

Attribute	Size	Description
`resistance`	1	Terminal resistance R
`motorconst`	2	Motor constants (K_t, K_e; see below)
`nominal`	3	Nominal operating point (v_n, τ_0, ω_0)
`inductance`	2	Electrical dynamics (L, t_e)
`thermal`	6	Thermal model ($R_T, C, t_T, \alpha, T_0, T_a$)
`saturation`	3	Limits ($\tau_{\max}, i_{\max}, (di/dt)_{\max}$)
`cogging`	3	Cogging torque (A, N_p, ϕ)
`lugre`	5	LuGre friction ($\sigma_0, \sigma_1, \tau_c, \tau_s, \omega_s$)
`damping`	3	Viscous damping coefficients
`armature`	1	Armature inertia
`input`	string	Input block: subset of `pos vel ff voltage`, or `none`
`controller`	6	Gains, slew, and voltage saturation ($k_p, k_i, k_d, s, I_{\max}, v_{\max}$)

Table 5: MJCF attributes for the `dcmotor` actuator, split into electrical, mechanical and controller groupings.

2.1 Stateless DC Motor

The output torque of the stateless motor follows Eq. (3), mapping physical parameters to the underlying affine model. The three core parameters are the effective motor constant K, resistance R, and maximum torque $\tau_{\max}$. They are stored in `mjModel` as follows:

Symbol	Description	`mjModel` storage
R	Resistance	`gainprm[0]`
K	Effective motor constant	`gainprm[1]`
$\tau_{\max}$	Maximum torque	`forcerange`

Table 6: Stateless DC motor core parameters.

The gain $G = K/R$ and back-EMF bias $-GK\omega$ are computed at runtime. Storing R separately allows temperature-dependent resistance $R(T)$, Eq. (8), to be applied. These three core parameters can be specified with a combination of eight sub-attributes

Attribute	Symbol	Units
`resistance`	R	Ohm
`motorconst:Kt`	K_t	N·m/A
`motorconst:Ke`	K_e	V·s/rad
`nominal:voltage`	v_n	Volt
`nominal:stall_torque`	$\tau_0 = K v_n / R$	N·m
`nominal:no_load_speed`	$\omega_0 \approx v_n / K$	rad/s
`saturation:torque`	$\tau_{\max}$	N·m
`saturation:current`	$i_{\max}$	Ampere

Table 7: Stateless DC motor basic attributes.

The attribute `motorconst` has two sub-attributes, `Kt` and `Ke`. If both are positive, $K = \sqrt{K_t K_e}$ (preserving power balance $K^2 = K_t K_e$). If only one is positive, K equals that value.

The following attribute combinations are supported:

1. Effective motor constant K, one of:
   - `motorconst:Kt` *and/or* `motorconst:Ke`
   - `nominal:no_load_speed` *and* `nominal:voltage`
2. Resistance R, one of:
   - `resistance`
   - `nominal:stall_torque` *and* `nominal:voltage`
3. Maximum torque $\tau_{\max}$, one of:
   - `saturation:torque`
   - `saturation:current`

`saturation:current` corresponds to the continuous (thermal) current limit. Peak current behavior can be approximated with the thermal model (§2.3).

Rotor Inertia and Gearing. To model rotor inertia with a gear train, set the actuator's `armature` $= J_r$ and `gear` $= N$. Actuator-level `armature` automatically scales the inertia by N^2 to reflect J_{eff} to the output shaft.

Mechanical Drag. The full torque-speed envelope applies viscous drag *outside* the current clamp:

$$\tau_{\text{net}} = \text{clip}\left(\frac{K}{R}(v - K\,\omega),\ \pm\tau_{\max}\right) - b(\omega)$$

Actuator-level `damping` reproduces this post-clamp behavior. It accepts an array of polynomial drag coefficients (`damping[0]` $= B_1$, `damping[1]` $= B_2$, ...). As with `armature`, actuator-level `damping` is scaled by N^2.

Cogging Torque. The magnetic torque ripple of Eq. (6) is modeled as a periodic bias added to the actuator force, where θ is the `actuator_length`, i.e. the transmission-transformed joint angle.

Attribute	Symbol	Units
`cogging:amplitude`	A	N·m
`cogging:poles`	N_p	dimensionless
`cogging:phase`	ϕ	radian

Table 8: Cogging torque attributes.

Mapping to Isaac Lab. Isaac Lab [9] implements the stateless DC motor model. Table 9 maps its attributes to the constants defined in this document and to the `dcmotor` attributes.

Symbol	MuJoCo	Isaac Lab
τ_0	`nominal:stall_torque`	`saturation_effort`
ω_0	`nominal:no_load_speed`	`velocity_limit`
$\tau_{\max}$	`saturation:torque`	`effort_limit`

Table 9: Mapping of attributes to Isaac Lab.

Isaac Lab does not expose electrical parameters. To reproduce the same torque-speed envelope, set `nominal:voltage` to any positive value (e.g., `1`) and `ctrlrange` to $\pm$ that value (e.g., `"-1 1"`).

Gearbox Efficiency. Gearbox efficiency η is not a separate attribute. To account for transmission losses, reduce the motor constant: $K \to \eta K$. This correctly reduces forward torque transmission.

Computed parameters. Several derived quantities that appear on datasheets can be computed and used to cross-check the parameterization. The torque-speed gradient $\partial\omega/\partial\tau = -R/K^2$ gives the slope of the torque-speed line (Table 1). The mechanical time constant $t_m = RJ/K^2$ is the time for the motor to reach 63% of its no-load speed under a voltage step, where J is the rotor inertia (`armature`). The nominal (continuous) torque is $\tau_n = K \cdot i_{\max}$. The no-load current i_0 can be computed from Eq. (5) given known friction parameters. See Table 17.

Not modeled: Nonlinear torque constant $K_t(i)$. Separate K_t and K_e values are accepted via `motorconst:Kt` and `motorconst:Ke` but collapsed to a single effective $K = \sqrt{K_t K_e}$.

2.2 Stateful Current

A winding current state variable governed by Eq. (4) is added if the electrical time constant $t_e > 0$ (derived from inductance $L > 0$ or specified directly). When enabled, the state is integrated by `mjDYN_DCMOTOR`, and the gain switches from K/R (stateless) to K (stateful). The time constant t_e can be determined by either:

- `inductance:timeconst`
- `inductance:L` *and* `resistance` (via $t_e = L/R$)

Current rate limiting. When the sub-attribute `saturation:current_rate` is set ($(di/dt)_{\max} > 0$) and the current state is enabled ($t_e > 0$), the rate of change of current is clamped:

$$\frac{di}{dt} \leftarrow \text{clip}\left(\frac{di}{dt},\ \pm(di/dt)_{\max}\right)$$

This limits the torque ramp rate to $K \cdot (di/dt)_{\max}$ (N·m/s) without requiring any additional state variables, since the current i is already an activation variable and we are simply clamping its rate of change. This attribute has no effect when $t_e = 0$ (stateless current).

Attribute	Symbol	Units
`inductance:L`	L	Henry
`inductance:timeconst`	$t_e = L/R$	second
`saturation:current_rate`	$(di/dt)_{\max}$	Ampere/second

Table 10: Stateful current attributes.

2.3 Temperature

A winding temperature state governed by the lumped ODE (7) is added if any of the thermal attributes (R_T, C, t_T) are specified. The state T is the temperature rise above ambient ($T = T_{\text{winding}} - T_a$), so the absolute temperature is $T + T_a$. Temperature modifies the winding resistance via Eq. (8), which feeds back into the motor equation: higher temperature increases resistance, leading to reduced current for a given voltage.

Attribute	Symbol	Units
`thermal:resistance`	R_T	K/W
`thermal:capacitance`	C	J/K
`thermal:timeconst`	$t_T = R_T C$	s
`thermal:tempcoef`	α	1/K
`thermal:reftemp`	T_0	°C
`thermal:ambient`	T_a	°C

Table 11: Thermal model attributes.

Any two of `thermal:resistance`, `thermal:capacitance` and `thermal:timeconst` determine the third. If all three are given, the capacitance is taken as $C = t_T/R_T$.

Not modeled: Iron losses (§1.3.2), magnet flux derating (§1.3.4), and thermal current derating (§1.3.5). Only copper losses (i^2R) drive the thermal model; K is treated as temperature-independent.

2.4 Stateful Friction

A bristle deflection state governed by the LuGre model (§1.4.2) is added if the bristle stiffness $\sigma_0 > 0$.

The Stribeck function $g(\omega)$, Eq. (10), determines velocity-dependent friction, and the friction force is given by Eq. (9b). The bristle state is integrated using the exact ZOH scheme (11). The viscous term $\sigma_2\omega$ is specified by via the standard `damping` attribute to leverage MuJoCo's implicit integration.

Integration. The bristle stiffness σ_0 is typically very large (10^5–10^6 N·m/rad), creating a stiff ODE. At constant velocity, the state equation (9a) has the form $\dot{z} = az + b\omega$ where $a = -\sigma_0|\omega|/g(\omega)$ and $b = 1$. Euler integration is unstable unless $|1 + a\Delta t| < 1$, requiring impractically small timesteps ($\Delta t < 2g(\omega)/(\sigma_0|\omega|)$, on the order of microseconds). Under a zero-order hold assumption (ω constant over the timestep), the linear ODE $\dot{z} = az + b\omega$ can be solved exactly:

$$z_{k+1} = e^{a\Delta t} z_k + \frac{b(e^{a\Delta t} - 1)}{a}\,\omega \tag{11}$$

reducing to $z_{k+1} = z_k + b\omega\Delta t$ in the limit $a \to 0$. This integration is unconditionally stable for any Δt.

Attribute	Symbol	Units
`lugre:stiffness`	σ_0	N·m/rad
`lugre:damping`	σ_1	N·m·s/rad
`lugre:coulomb`	τ_c	N·m
`lugre:static`	τ_s	N·m
`lugre:stribeck`	ω_s	rad/s

Table 12: LuGre friction attributes.

Not modeled: Velocity-dependent bristle damping $\sigma_1(\omega)$ (§1.4.3), a constant σ_1 is used. The Stribeck exponent is not exposed and fixed at $\gamma = 2$.

2.5 PID Controller

Many actuators embed an on-board controller computing drive voltage from position or velocity commands. To model such actuators, `dcmotor` supports an optional controller layer upstream of the motor physics. Two attributes control this behavior:

Attribute	Type	Description
`input`	string	subset of `pos vel ff voltage`, in canonical order; `none` (passive)
`controller`	vector	Gains, slew, and voltage saturation $(k_p, k_i, k_d, s, I_{\max}, v_{\max})$

Table 13: Controller attributes. Default `input` is `voltage`.

Unlike the motor parameters in Table 17, controller gains are user-specified firmware settings. MuJoCo uses *torque-space* units (e.g., k_p in N·m/rad), matching the `pid` actuator. Voltage-space datasheet gains (V/rad) convert by multiplying by K/R, though empirical calibration is often necessary due to unknown internal units on real hardware.

The controller computes a target voltage v from the `ctrl` command. All motor physics — cogging, saturation, friction, etc. — apply identically downstream of v. The `input` attribute selects the controller:

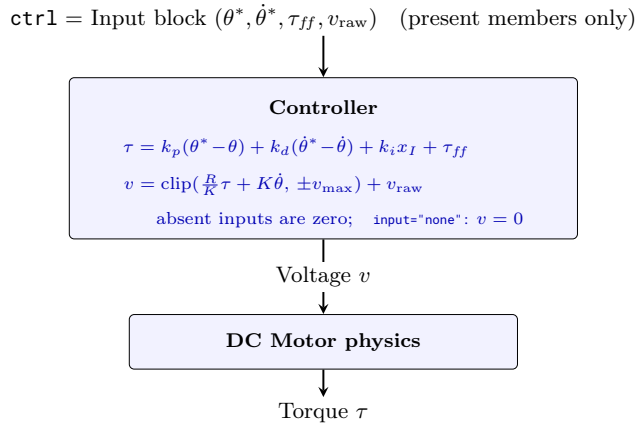


Figure 7: Controller pipeline. The `input` attribute selects which members of the control block are present; motor physics is identical downstream.

2.5.1 Setpoint Mode

When `input="pos vel ff"` (or subset thereof), the actuator owns a block of controls $u = (\theta^*, \dot\theta^*, \tau_{ff})$: a position setpoint, a velocity setpoint and a torque feedforward. Any subset may be selected, and the `dcmotor`'s `actuator_ctrlnum` is variable. An absent input corresponds to a setpoint frozen at zero. The controller commands a torque and converts it to drive voltage:

$$\tau = k_p\,(\theta^* - \theta) + k_d\,(\dot\theta^* - \dot\theta) + k_i\,x_I \;+\; \tau_{ff} \tag{12}$$

$$v = \frac{R}{K}\,\tau + K\dot\theta \tag{13}$$

where the second term of (13) compensates back-EMF, as the current loop of a real torque-mode driver does: commanded torque is delivered *exactly* until a limit binds, and the torque-speed envelope emerges from voltage saturation (§2.5.5). The map uses the nameplate R; thermal resistance growth (§1.3.3) is not compensated. The integrator state accumulates the position error, $\dot x_I = \theta^* - \theta$, subject to anti-windup (§2.5.4). When $k_i = 0$, no integral state is added and the controller reduces to PD; when $k_p = k_i = 0$, it is a pure velocity servo. The back-EMF compensation also cancels the motor's $-K^2/R$ damping bias, so the net physical damping delivered by an unclipped torque-mode motor is $-k_d$, matching `pid`. The separate `voltage` input is the raw terminal voltage, added downstream of the controller and its saturation.

2.5.2 Passive Operation

`input="none"` selects the *empty* input block: the actuator owns no controls at all and acts as a purely passive device. The terminal voltage is identically zero, so the motor operates with shorted terminals: back-EMF drives current through R and brakes the joint (dynamic braking), while cogging (6) and LuGre friction (§1.4.2) act as passive joint forces. Setting `motorconst` to zero decouples the electrical branch, leaving pure friction and cogging (R must still be positive). This exists because MuJoCo associates auxiliary dynamic states (such as the LuGre bristle) with actuators, not joints: a passive `dcmotor` is the idiomatic way to attach these passive phenomena to a joint. Controller gains, slew rate limiting and integral gain require a controller input and are rejected.

2.5.3 Setpoint Slew Rate

The user command u = `ctrl` can change discontinuously between timesteps. When `controller:slewmax` is set ($s > 0$), the effective setpoint is rate-limited:

$$u \leftarrow \text{clip}(u,\; u_{\text{prev}} \pm s \cdot \Delta t)$$

where u_{prev} is the previous effective setpoint and Δt is the timestep. This smoothly ramps the reference trajectory instead of allowing instantaneous jumps.

State variable. When $s > 0$, one activation state u_{prev} is allocated, and updated each step to u (post-clamping).

Units. The slew rate s limits the first controller input, with corresponding units: position setpoint (rad/s), velocity setpoint (rad/s^2), or torque feedforward (N·m/s). The raw `voltage` input is never rate-limited; `slewmax` requires a controller input.

2.5.4 Anti-windup

When $k_i > 0$, the integrator state x_I provides steady-state error correction. However, sustained saturation or large setpoint changes can cause x_I to grow excessively, leading to overshoot (integral windup). To prevent this,

when `controller:Imax` is set ($I_{\max} > 0$), the state is bounded each step:

$$x_I \leftarrow \text{clip}(x_I, \pm I_{\max})$$

This prevents controller windup even when drive signals are saturated.

2.5.5 Voltage Saturation

With setpoint inputs, the computed voltage v can be arbitrarily large (proportional to the error). Real motor drivers are limited by their supply voltage. When `controller:Vmax` is set ($v_{\max} > 0$), a voltage clamp is applied before the motor equations:

$$v \leftarrow \text{clip}(v, \pm v_{\max})$$

This differs from `ctrlrange` (clamping user commands u) and `forcerange` (clamping output torque). The clamp applies to the controller output only: the raw `voltage` input is added downstream, unclamped (use `ctrlrange` to bound it). Because the torque-to-voltage map (13) compensates back-EMF, the achievable torque under `controller:Vmax` shrinks with speed: the torque-speed envelope of §1.1.

Attribute	Symbol	Units
`controller:kp`	k_p	N·m/rad
`controller:ki`	k_i	N·m/(rad·s)
`controller:kd`	k_d	N·m·s/rad
`controller:slewmax`	s	ctrl-units/s
`controller:Imax`	$I_{\max}$	rad·s
`controller:Vmax`	$v_{\max}$	Volt

Table 14: Controller attributes.

2.6 Low-Level Semantics

The `dcmotor` actuator uses the enum value types `mjGAIN_DCMOTOR`, `mjDYN_DCMOTOR`, `mjBIAS_DCMOTOR`, and populates the rows of several `mjModel` arrays (all `actuator_*`), as follows:

Array	Index	Symbol	Description
`gainprm`	0	R	Resistance (Ω)
	1	K	Motor constant (N·m/A)
	2	α	Resistance coeff. (K^{-1})
	3	T_0	Reference temperature (°C)
	4	k_p	Controller proportional gain
	5	k_i	Controller integral gain
	6	k_d	Controller derivative gain
	7	$v_{\max}$	Voltage saturation (V)
	8	—	Reserved
`dynprm`	0	t_e	Electrical time constant (s)
	1	$(di/dt)_{\max}$	Current rate limit (A/s)
	2	R_T	Thermal resistance (K/W)
	3	C	Thermal capacitance (J/K)
	4	T_a	Ambient temperature (°C)
	5	σ_0	LuGre bristle stiffness
	6	σ_1	LuGre bristle damping
	7	s	Controller slew rate
	8	$I_{\max}$	Integral limit (anti-windup)
`biasprm`	0	A	Cogging amplitude (N·m)
	1	N_p	Cogging periodicity
	2	ϕ	Cogging phase (rad)
	3	τ_c	LuGre Coulomb fric. (N·m)
	4	τ_s	LuGre static fric. (N·m)
	5	ω_s	Stribeck velocity (rad/s)
`forcerange`	0	$-\tau_{\max}$	Minimum torque (N·m)
	1	$\tau_{\max}$	Maximum torque (N·m)
`damping`	0	$B_1(+\sigma_2)$	Linear (+ LuGre viscous)
	1	B_2	Quadratic
	2	B_3	Cubic
`armature`	0	J_r	Actuator armature
`gear`	0	N	Gear ratio
`ctrlspec`	0	—	Input signature (`mjtCtrlInput` bitmask)

Table 15: `mjModel` array semantics for the `dcmotor` actuator.

Runtime mutability. Most `mjModel` parameters listed above may be freely modified at runtime for system identification or gain tuning. However, five parameters control the *number* of activation states, which is determined at compile time and cannot change during simulation. Toggling any of the following parameters between zero and positive after compilation is an error:

Parameter	Storage	State	Semantics
s	`dynprm[7]`	u_{prev}	previous control
k_i	`gainprm[5]`	x_I	controller integral
R_T, C	`dynprm[2,3]`	T	temperature rise
σ_0	`dynprm[5]`	z	bristle deflection
t_e	`dynprm[0]`	i	winding current

Table 16: Compile-time state switches, allocated in `act` in the order shown. Do not toggle between zero and positive at runtime.

3 Datasheet Mapping

Table 17 maps commercial motor datasheet specifications to attributes. The left column shows the datasheet entry as typically labeled by motor manufacturers; the right column shows the corresponding `dcmotor` MJCF attribute, when one exists. Derived quantities that are not direct attributes (gradient, mechanical time constant) are included for completeness.

Specification	Symbol	Formula / Note	Datasheet Symbol	Attribute
Resistance	R	Terminal resistance	R, Ra	`resistance`
Torque Constant	K_t	$\tau = K_t i$	kt, km	`motorconst:Kt`
Back-EMF Constant	K_e	$v_{\text{back}} = K_e \omega$	ke	`motorconst:Ke`
Speed Constant	K_v	$K_v = 1/K_e$	kn, kv	1/`motorconst:Ke`
Nominal Voltage	v_n	Rated voltage (e.g., 24V)	Un, VDC	`nominal:voltage`
No-load Speed	ω_0	$\omega_0 \approx v_n/K$	n0	`nominal:no_load_speed`
Stall Torque	τ_0	$\tau_0 = K v_n/R$	MH, Ts	`nominal:stall_torque`
Stall Current	i_s	Max. possible: $i_s = v_n/R$	IA	
Nominal Current	$i_{\max}$	Thermal limit (continuous)	Ic, IN	`saturation:current`
Peak Current	i_{peak}	Short-term limit	Ipk	
Coulomb Friction	τ_c	Dry friction opposing motion	T_f	`frictionloss` (joint)
Viscous Friction	B	Drag $\propto \omega$	C_v	`damping`
Rotor Inertia	J_r	Reflected: $J_{\text{eff}} = J_r N^2$	J, Jm	`armature`
Gear Ratio	N	Reduction ratio	N, i	`gear`
Gearbox Efficiency	η	Fold into K: use ηK	η	
Cogging Amplitude	A	Peak cogging torque	—	`cogging:amplitude`
Cogging Periodicity	N_p	Poles $\times$ slots/pole	—	`cogging:poles`
Nominal Torque	τ_n	$\tau_n = K \cdot i_{\max}$	M_N, T_c	
No-load Current	i_0	Friction: Eq. (5)	I_0	
Gradient	$\partial\omega/\partial\tau$	$-R/K^2$	$\Delta n/\Delta M$	
Mech. Time Const.	t_m	$t_m = R\,J/K^2$	τ_m	
Inductance	L	Terminal inductance	L	`inductance:L`
Elec. Time Const.	t_e	$t_e = L/R$	τ_e	`inductance:timeconst`
Thermal Resistance	R_T	Winding-to-ambient	Rth	`thermal:resistance`
Thermal Capacitance	C	$C = t_T/R_T$	C_{th}	`thermal:capacitance`
Thermal Time Const.	t_T	$t_T = R_T C$	τ_{th}	`thermal:timeconst`
Ref. Temperature	T_0	Temperature at which R is specified	T_{ref}	`thermal:reftemp`
Ambient Temperature	T_a	Operating environment	—	`thermal:ambient`
Max. Winding Temp.	$T_{\max}$	Absolute limit	$T_{\max}$	
Res. Temp. Coeff.	α	$\approx 0.0039\,\text{K}^{-1}$ (copper)	α_{Cu}	`thermal:tempcoef`

Table 17: Datasheet parameters and their relation to model constants. Groups: electrical, mechanical, derived, inductance, thermal.

References

[1] A. Hughes and B. Drury, *Electric Motors and Drives: Fundamentals, Types and Applications*. Newnes, 5th ed., 2019.

[2] Maxon Motor AG, "Key Information on Maxon DC Motors and Maxon EC Motors." https://www.maxongroup.com, 2024. Maxon Academy Technical Notes.

[3] MathWorks, "DC Motor — Simscape Electrical Block Reference." https://www.mathworks.com/help/sps/ref/dcmotor.html, 2024.

[4] R. Tedrake, *Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation*. MIT, 2024. Course notes for MIT 6.832, https://underactuated.mit.edu.

[5] P. Dahl, "A Solid Friction Model," Tech. Rep. TOR-0158(3107-18)-1, The Aerospace Corporation, El Segundo, CA, 1968.

[6] R. Stribeck, "Die wesentlichen Eigenschaften der Gleit- und Rollenlager," *Zeitschrift des Vereines Deutscher Ingenieure*, vol. 46, pp. 1341–1348, 1432–1438, 1463–1470, 1902.

[7] C. Canudas de Wit, H. Olsson, K. J. Åström, and P. Lischinsky, "A New Model for Control of Systems with Friction," *IEEE Transactions on Automatic Control*, vol. 40, pp. 419–425, Mar. 1995.

[8] K. J. Åström and C. Canudas de Wit, "Revisiting the LuGre Friction Model," *IEEE Control Systems Magazine*, vol. 28, pp. 101–114, Dec. 2008.

[9] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Isaac Lab: A Unified and Modular Framework for Robot Learning," *arXiv preprint arXiv:2502.11048*, 2025.