## Coding Interview

This repository contains sample questions and answers for coding interviews.

### Arrays

1. \*\*Merge
   * Init
   * Con.
2. **Other**
   * For
   * Line

# PID Tuning Cheatsheet for a Servo with a Fast Current (Torque) Loop

**Goal:** A practical, engineer-friendly recipe to pick **PID** gains for a **single outer position loop** when the drive’s **current loop is fast** (20 kHz class) so **torque ≈ proportional to current command**.

This file uses GitHub math (,

). If any formulas don’t render, you can keep them as plain text.

## 1) Assumptions & Symbols

* Inner **current loop** is tightly closed by the drive → torque follows command:
  + In the outer loop we model an **effective torque gain** (K\_\tau) (use (K\_\tau = K\_t) if your controller outputs current; otherwise fold it into the PID gains).
* **Mechanical plant** about motor shaft with load reflected:
  + Inertia (J) [kg·m²], viscous damping (b) [N·m·s/rad]
* Laplace variable (s), position (\Theta(s)), speed (\Omega(s))

**Mechanical pole (no control):** (\displaystyle \omega\_m = \frac{b}{J}) (rad/s)

## 2) Plant and Loop Definitions

**Torque → motion:** [ G\_{\omega\tau}(s)=\frac{\Omega(s)}{\tau(s)}=\frac{1}{J s + b},\qquad G\_{\theta\tau}(s)=\frac{\Theta(s)}{\tau(s)}=\frac{1}{s(J s + b)}. ]

**Controller (PID):**

* Ideal form: [ C\_{\text{ideal}}(s)=K\_p+\frac{K\_i}{s}+K\_d s = \frac{K\_d s^2 + K\_p s + K\_i}{s}. ]
* Practical derivative (noise-friendly): [ C(s)=K\_p+\frac{K\_i}{s}+\frac{K\_d s}{1+\tau\_d s}. ]

**Open loop (unity feedback on position):** [ L(s)=C(s),K\_\tau,G\_{\theta\tau}(s). ]

**Closed-loop complementary sensitivity (command → position):** [ T(s)=\frac{\Theta(s)}{R(s)}=\frac{L(s)}{1+L(s)}. ]

**Sensitivity (disturbance rejection, and error to reference):** [ S(s)=\frac{1}{1+L(s)}. ]

**With ideal PID**, the denominators are explicitly: [ L\_{\text{ideal}}(s)=\frac{K\_\tau(K\_d s^2 + K\_p s + K\_i)}{J s^3 + b s^2}, ] [ T\_{\text{ideal}}(s)=\frac{K\_\tau(K\_d s^2 + K\_p s + K\_i)}{J s^3 + (b+K\_\tau K\_d)s^2 + (K\_\tau K\_p)s + (K\_\tau K\_i)}. ]

**Stability margins** (Bode of (L(j\omega))): [ \text{PM} = 180^\circ + \angle L(j\omega\_{gc}),\qquad \text{GM}*{\mathrm{dB}}=-20\log*{10}|L(j\omega\_{pc})|. ] You want **PM > 0°** (typically 45–60°) and **GM > 0 dB** (≥ 6–10 dB).

## 3) One-Shot Tuning Recipe (Position loop only)

This shapes the loop so it looks like ~(K/s) at crossover → good damping.

**Step A — Add “velocity feedback”:** place the controller zero near the mechanical pole [ T\_d ;\triangleq; \frac{J}{b},\qquad K\_d = K\_p,T\_d. ] (With the derivative filter, this behaves like velocity feedback around crossover.)

**Step B — Pick a target crossover** (\omega\_c).  
A safe starting range with the above zero is (\omega\_c \in [0.5,,2],\omega\_m).  
Go higher for bandwidth if model/derivative are trustworthy.

**Step C — Set proportional gain by the unity-gain condition** [ \boxed{K\_p = \frac{b,\omega\_c}{K\_\tau}}. ]

**Step D — Add slow integral to remove steady-state error** [ T\_i \in \Big[\frac{8}{\omega\_c},,\frac{16}{\omega\_c}\Big],\qquad \boxed{K\_i = \frac{K\_p}{T\_i}}. ] (Use the larger (T\_i) for more phase margin; reduce (T\_i) if steady-state is too slow.)

**Step E — Filter the derivative** [ \boxed{\tau\_d \approx \frac{1}{10,\omega\_c}} ] This keeps derivative “active” near (\omega\_c) but rolls off sensor noise.

**Summary (compute in this order):**

1. (T\_d = J/b)
2. choose (\omega\_c)
3. (K\_p = b,\omega\_c/K\_\tau)
4. (K\_d = K\_p,T\_d)
5. choose (T\_i) in ([8,16]/\omega\_c) and set (K\_i = K\_p/T\_i)
6. (\tau\_d \approx 1/(10,\omega\_c))

## 4) How to Verify

* **Bode of open loop (L)** → read PM/GM; aim for PM ≈ 45–65°.
* **Step of (T)** → check rise/overshoot/settling; integral ensures zero steady-state error.
* **Bode of (S)** → low-frequency (|S|) small is good for disturbance rejection.
* **Control effort** (C/(1+L)) → ensure commanded torque stays within limits (watch saturation; add anti-windup if needed).

## 5) Worked Example (numbers are illustrative)

Given (J=0.01) kg·m², (b=0.001) N·m·s/rad, (K\_\tau=K\_t=0.10) N·m/A:

* Mechanical pole: (\omega\_m=b/J=0.1) rad/s
* Choose (\omega\_c = 1.0) rad/s (≈ (10\times \omega\_m) after zero-placement we can push higher; conservative users may choose (0.5)–(1.0))
* (T\_d = J/b = 10) s
* (K\_p = b,\omega\_c/K\_\tau = 0.001\times 1.0/0.1 = 0.010)
* (K\_d = K\_p T\_d = 0.010 \times 10 = 0.10)
* (T\_i = 8/\omega\_c = 8) s → (K\_i = K\_p/T\_i = 0.010/8 = 1.25\times10^{-3})
* (\tau\_d \approx 1/(10,\omega\_c) = 0.10) s

These gains typically yield **good PM** and a **well-damped** step.

## 6) Practical Notes & Troubleshooting

* **Too slow?** Increase (\omega\_c) → raise (K\_p); keep (T\_d=J/b), reduce (T\_i) (e.g., from (16/\omega\_c) toward (8/\omega\_c)), keep (\tau\_d \approx 1/(10,\omega\_c)).
* **Ringing / low PM?** Lower (\omega\_c) or move the zero slightly earlier (increase (T\_d) a bit). If integral is aggressive, **increase (T\_i)**.
* **Noisy control / encoder noise?** Increase (\tau\_d) or lower (\omega\_c).
* **Friction (Coulomb/stiction):** treat as a disturbance; stronger integral (smaller (T\_i)) helps but watch overshoot and windup.
* **Gearing & load reflection:** if the gear ratio is (N=\omega\_m/\omega\_L), [ J = J\_m + N^2 J\_L,\qquad b = b\_m + N^2 b\_L,\qquad \theta\_L=\frac{\theta\_m}{N}. ]
* **Resonances / compliance:** a flexible shaft or two-mass load may need notch filters or an inner **velocity loop**.
* **Anti-windup:** clamp the integrator or back-calculate when torque saturates.

## 7) (Optional) Python Snippet to Plot Step & Bode

import control as ctl, matplotlib.pyplot as plt  
  
# Example parameters  
J, b, K\_tau = 0.01, 0.001, 0.10  
omega\_c = 1.0  
T\_d = J/b  
Kp = b\*omega\_c/K\_tau  
Kd = Kp\*T\_d  
Ti = 8/omega\_c  
Ki = Kp/Ti  
tau\_d = 1/(10\*omega\_c)  
  
s = ctl.TransferFunction.s  
G = 1/(s\*(J\*s+b))  
C = Kp + Ki/s + (Kd\*s)/(1+tau\_d\*s)  
  
L = C\*K\_tau\*G  
T = ctl.feedback(L,1)  
S = 1/(1+L)  
  
# Margins  
gm, pm, wg, wp = ctl.margin(L)  
print(f"GM={gm:.2g} ({20\*\_\_import\_\_('math').log10(gm):.1f} dB) @ {wg:.3g} rad/s, PM={pm:.1f}° @ {wp:.3g} rad/s")  
  
# Plots  
t, y = ctl.step\_response(T)  
plt.figure(); plt.plot(t,y); plt.grid(True); plt.title("Closed-loop Step"); plt.xlabel("Time [s]"); plt.ylabel("Position [rad]")  
plt.figure(); ctl.bode(L, dB=True); plt.suptitle("Open-loop Bode"); plt.show()