Root Cause Analysis: RE3 Deployment Disaster

Jacobian Amplification of Structural Dynamics in
Pinocchio-based Wrench Estimation

H1.2 Adaptive Policy Team

2026-03-16

Contents

1 Abstract

During episode `re3_real_encode_tr1`, an RL balancing policy deployed on the H1.2 humanoid robot caused violent involuntary leg thrashing within 0.16 seconds of engagement, ultimately producing hip-pitch torques exceeding 428 N·m. The robot was stopped before structural damage occurred.

This document provides a ground-up technical analysis of the root cause. We cover: (1) a first-principles introduction to rigid-body dynamics and the Pinocchio framework; (2) the mathematics of estimating an external wrench from internal motor torques via the Jacobian pseudo-inverse; (3) why an ill-conditioned Jacobian amplifies any upstream torque residual into a spurious end-effector force; (4) a step-by-step numerical reconstruction of the cascade from policy steps 0–8; and (5) a revised causal hypothesis that inverts the previously posted narrative.

Revised hypothesis (confirmed by data):

1. In RE3, the left arm was in the `arm_waist_target` pose (shoulder pitched −17 from motor zero, elbow extended 86 from motor zero), which physically corresponds to the arm hanging *straight down and limp* alongside the torso. In RE1/RE2 the arms were at motor zero, physically corresponding to forearms bent at the sides. The extended/limp left arm created a **15.6× torso→wrist-F_z Jacobian amplification**.
2. Both **RE3** and **RE2** began with similar large hip-roll position errors (∼0.35–0.51 rad). The initial leg dynamics were comparable, but RE3's ill-conditioned Jacobian turned normal torso structural torques into catastrophic phantom wrist forces.
3. The structural torque required by the torso motor to counteract leg dynamics was indistinguishable from normal, but with a 15.6× amplification it produced a −533 N spurious left wrist force.
4. The policy interpreted this as a large external push at the wrist and commanded increasingly extreme leg targets, driving the torso torque higher, completing a positive feedback loop.

2 Background: Pinocchio and Rigid-Body Dynamics

2.1 What is Pinocchio?

Pinocchio [1] is an open-source rigid-body dynamics library that operates on a robot model loaded from a URDF file. It represents the robot as a *kinematic tree*: a directed graph of rigid bodies (*links*) connected by joints. For the H1.2 we use a free-flyer root joint (6-DOF floating base) plus 27 actuated joints (legs, torso, arms), giving a configuration space of dimension $n_q = 34$ (7 free-flyer coefficients + 27 joint angles) and velocity space $n_v = 33$.

The library provides:

- **Forward kinematics** (`forwardKinematics`): given $\boldsymbol{q}$, compute the pose of every link/frame.
- **Recursive Newton-Euler Algorithm** (`rnea`): compute the joint torques $\boldsymbol{\tau}$ required to achieve a given $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$, including gravity.
- **Frame Jacobian** (`computeFrameJacobian`): the matrix $\boldsymbol{J} \in \mathbb{R}^{6\times n_v}$ mapping joint velocities to the 6-D twist (linear+angular velocity) of a specific link frame.

2.2 The Robot Configuration Vector

The 27 actuated joints are ordered as shown in Table 1. All data arrays (`tau.npy`, `qpos.npy`, `target_dof.npy`) use this ordering.

Table 1: Joint index ordering in 27-column arrays.

Indices	Group	Joints
0–5	Left leg	l_hip_yaw, l_hip_pitch, l_hip_roll, l_knee, l_ank_pitch, l_ank_roll
6–11	Right leg	r_hip_yaw, r_hip_pitch, r_hip_roll, r_knee, r_ank_pitch, r_ank_roll
12	Torso	torso
13–19	Left arm	L_sh_pitch, L_sh_roll, L_sh_yaw, L_elbow, L_wr_roll, L_wr_pitch, L_wr_yaw
20–26	Right arm	R_sh_pitch, R_sh_roll, R_sh_yaw, R_elbow, R_wr_roll, R_wr_pitch, R_wr_yaw

2.3 Forward Kinematics and the Frame Jacobian

Given a configuration $\boldsymbol{q}$, forward kinematics computes the homogeneous transform $T_f^w(\boldsymbol{q}) \in SE(3)$ from the world frame to any target frame f. For the *left wrist yaw link*:

$$T_{\text{lwrist}}^w(\boldsymbol{q}) = T_{\text{pelvis}}^w \cdot T_{\text{torso}}^{\text{pelvis}} \cdot T_{\text{L\_sh\_pitch}}^{\text{torso}} \cdots T_{\text{lwrist}}^{\text{L\_elbow}}, \tag{1}$$

where each factor depends only on the joint angles along the kinematic chain.

The *geometric Jacobian* of frame f in the LOCAL_WORLD_ALIGNED convention is the $6 \times n_v$ matrix:

$$\boldsymbol{J}_f(\boldsymbol{q}) \triangleq \frac{\partial}{\partial \boldsymbol{q}} \begin{pmatrix} v_f \\ \omega_f \end{pmatrix}, \tag{2}$$

such that the 6-D twist of frame f is $\xi_f = \boldsymbol{J}_f \dot{\boldsymbol{q}}$.

For our 27-DOF motor space the Jacobian has shape 6×27.

2.4 Recursive Newton-Euler Algorithm (RNEA): Gravity Compensation

The RNEA computes, for any $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$, the joint torques $\boldsymbol{\tau} = \text{RNEA}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ satisfying the equations of motion:

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau} + \boldsymbol{J}^T \boldsymbol{F}_{\text{ext}}, \tag{3}$$

where M is the mass matrix, C the Coriolis/centrifugal matrix, $\boldsymbol{g}$ the gravity torque vector, and $\boldsymbol{F}_{\text{ext}}$ any external wrench.

Setting $\dot{\boldsymbol{q}} = \ddot{\boldsymbol{q}} = 0$ isolates gravity:

$$\boldsymbol{\tau}_{\text{grav}}(\boldsymbol{q}) = \text{RNEA}(\boldsymbol{q}, \boldsymbol{0}, \boldsymbol{0}). \tag{4}$$

This is the *gravity compensation torque*: the exact joint torques a perfectly rigid robot must exert merely to hold itself stationary against gravity at configuration $\boldsymbol{q}$.

2.5 Wrench Estimation via Pseudo-Inverse

`robot_model.py::get_frame_wrench` estimates an external wrench at frame f as follows:

Listing 1: Wrench estimator (robot_model.py).

```
def get_frame_wrench(self, frame_name, q=None, tau=None, imu_quat=
    None):
    tau_gravity = self.get_gravity_compensation(q, imu_quat)    # eq
        (3)
    jac = self.get_frame_jacobian(frame_name, q, imu_quat)      # J: 6
        x27
    wrench = np.linalg.pinv(jac.T) @ (tau - tau_gravity)       # eq
        (4)
    return wrench                                               # [F;
        M] in N, N*m
```

The mathematical basis is the static equilibrium assumption: if the *only* source of unexplained joint torque is an external wrench $\boldsymbol{w} \in \mathbb{R}^6$ applied at frame f, then

$$\boldsymbol{\tau}_{\text{meas}} - \boldsymbol{\tau}_{\text{grav}} = \boldsymbol{J}_f^T \boldsymbol{w}. \tag{5}$$

Solving for $\boldsymbol{w}$ with the Moore-Penrose pseudo-inverse:

$$\hat{\boldsymbol{w}} = (\boldsymbol{J}_f^T)^\dagger \left(\boldsymbol{\tau}_{\text{meas}} - \boldsymbol{\tau}_{\text{grav}}\right) = (\boldsymbol{J}_f^T)^\dagger \boldsymbol{\tau}_{\text{res}}. \tag{6}$$

Here $(\boldsymbol{J}_f^T)^\dagger \in \mathbb{R}^{6\times 27}$ is the 6×27 pseudo-inverse of $\boldsymbol{J}_f^T \in \mathbb{R}^{27\times 6}$. When $\boldsymbol{J}_f$ is full column rank (rank 6), $(\boldsymbol{J}_f^T)^\dagger = (\boldsymbol{J}_f \boldsymbol{J}_f^T)^{-1} \boldsymbol{J}_f$.

Critical assumption and its violation

Equation (5) assumes that $\boldsymbol{\tau}_{\text{res}}$ arises *exclusively* from $\boldsymbol{w}$. In practice, any other unexplained torque—inertial loading, joint friction, dynamic oscillations in upstream joints—is spuriously attributed to the external wrench at frame f. Concretely, if joint j has a residual $r_j = \tau_j - \tau_j^{\text{grav}}$, its contribution to the estimated wrist force is:

$$\Delta \hat{\boldsymbol{w}}_j = (\boldsymbol{J}_f^T)^\dagger_{[:,j]} \cdot r_j, \tag{7}$$

where $(\boldsymbol{J}_f^T)^\dagger_{[:,j]}$ is the j-th column of the pseudo-inverse. The *Fz contribution* from joint j is:

$$\Delta F_{z,j} = \left[(\boldsymbol{J}_f^T)^\dagger\right]_{[3,j]} \cdot r_j. \tag{8}$$

The scalar $g_j \triangleq \left[(\boldsymbol{J}_f^T)^\dagger\right]_{[3,j]}$ is the *torso-to-wrist-Fz gain* for joint j. A large $|g_j|$ means a small torque at joint j appears as a large force at the wrist.

3 Ill-Conditioned Jacobians: Amplification of Residuals

3.1 Singular Values and the Pseudo-Inverse

The singular value decomposition (SVD) of $\boldsymbol{J} \in \mathbb{R}^{6\times 27}$ is:

$$\boldsymbol{J} = U \Sigma V^T, \quad \Sigma = \text{diag}(\sigma_1, \ldots, \sigma_6), \quad \sigma_1 \geq \cdots \geq \sigma_6 \geq 0. \tag{9}$$

The pseudo-inverse is:

$$\boldsymbol{J}^\dagger = V \Sigma^{-1} U^T, \quad \Sigma^{-1} = \text{diag}(1/\sigma_1, \ldots, 1/\sigma_6). \tag{10}$$

The *condition number* $\kappa = \sigma_1/\sigma_6$ measures how much the pseudo-inverse amplifies errors. A component of $\boldsymbol{\tau}_{\text{res}}$ aligned with the singular direction corresponding to σ_6 is amplified by a factor $1/\sigma_6$.

For the wrench estimator in Eq. (6):

$$\hat{\boldsymbol{w}} = (\boldsymbol{J}^T)^\dagger \boldsymbol{\tau}_{\text{res}} = U \Sigma^{-T} V^T \boldsymbol{\tau}_{\text{res}}. \tag{11}$$

If $\boldsymbol{\tau}_{\text{res}}$ has any component in the direction v_6 (the right singular vector associated with σ_6), it is amplified by $1/\sigma_6$ in the wrench.

3.2 2D RRR Arm: A Worked Example

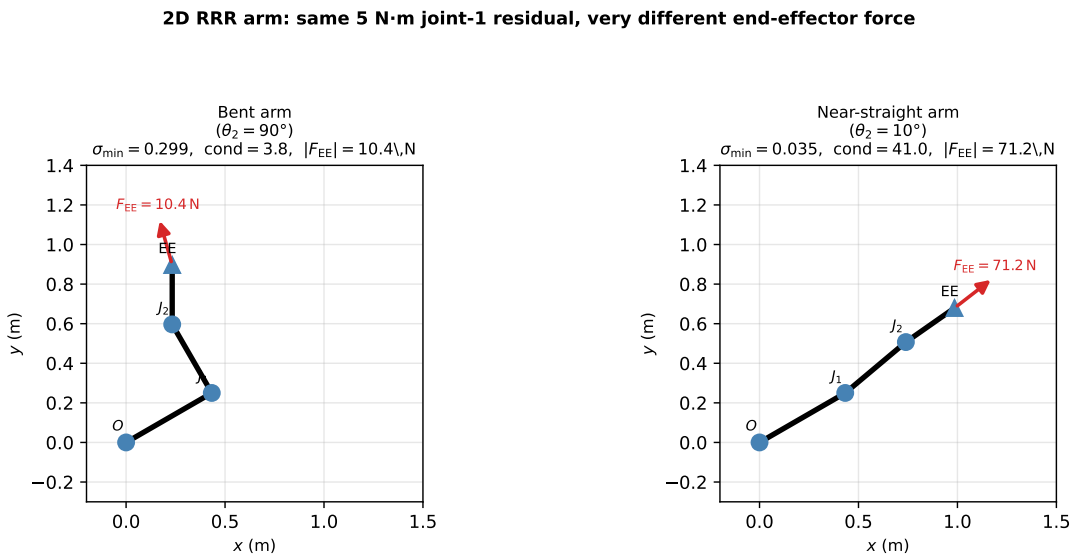

Figure 1: 2D 3-link RRR arm illustrating Jacobian ill-conditioning. Link lengths $\ell_1 = 0.5\,\text{m}$, $\ell_2 = 0.4\,\text{m}$, $\ell_3 = 0.3\,\text{m}$. A $5\,\text{N}\cdot\text{m}$ residual at joint 1 only is applied. The bent arm (left) is well-conditioned ($\kappa = 3.8$) and produces a modest 10 N end-effector force. The near-straight arm (right, $\theta_2 = 10°$) is ill-conditioned ($\kappa = 41$) and the same residual produces a 71 N force.

Consider a planar 3-link arm with revolute joints $\theta_1, \theta_2, \theta_3$. The end-effector position is:

$$x_{ee} = \ell_1 c_1 + \ell_2 c_{12} + \ell_3 c_{123}, \tag{12}$$

$$y_{ee} = \ell_1 s_1 + \ell_2 s_{12} + \ell_3 s_{123}, \tag{13}$$

where $c_k \equiv \cos\sum_{i\le k}\theta_i$, etc. The 2×3 Jacobian $\boldsymbol{J} \in \mathbb{R}^{2\times 3}$ maps $\dot{\boldsymbol{\theta}}$ to end-effector velocity, and its transpose $\boldsymbol{J}^T \in \mathbb{R}^{3\times 2}$ maps end-effector forces $\boldsymbol{F}_{ee}$ to joint torques: $\boldsymbol{\tau} = \boldsymbol{J}^T\boldsymbol{F}_{ee}$. Inverting: $\hat{\boldsymbol{F}}_{ee} = (\boldsymbol{J}^T)^\dagger \boldsymbol{\tau}_{\text{res}}$ (a 2×3 operation).

As $\theta_2 \to 0$ (arm near-straight), the second and third link become collinear, and two rows of $\boldsymbol{J}$ become nearly dependent. The smallest singular value $\sigma_{\min} \to 0$, making $(\boldsymbol{J}^T)^\dagger$ unbounded. Figure 1 shows the numerical outcome: with $\theta_2 = 90°$ (bent) a $5\,\text{N}\cdot\text{m}$ joint residual produces a 10 N end-effector force; with $\theta_2 = 10°$ the same residual produces 71 N — a $7\times$ amplification from geometry alone.

3.3 The Torso-to-Wrist Amplification in the H1.2

In the H1.2 kinematic tree, the *torso joint* sits directly between the pelvis and both arms. Its Jacobian column for the wrist frame encodes the cross-product lever arm from torso rotation to wrist linear velocity:

$$\boldsymbol{J}^{\text{linear}}_{\text{wrist, torso}} = \boldsymbol{\omega}_{\text{torso}} \times \boldsymbol{r}_{\text{torso}\to\text{wrist}}, \tag{14}$$

where $\boldsymbol{\omega}_{\text{torso}}$ is the unit rotation axis of the torso joint and $\boldsymbol{r}$ is the vector from torso joint to wrist frame. The magnitude of this lever arm determines how strongly a torso torque residual projects onto the wrist F_z direction.

Physical configuration at RE3 episode start

RE1/RE2 (motor zero ≈ 0) The arms are physically in the *bent-at-sides* configuration: elbow motor at ≈ 0 means the forearm is roughly horizontal at the sides, wrist at approximately shoulder height. Measured from the Pinocchio forward kinematics: left wrist at

(0.23, 0.21, +0.09) m, i.e. *above* the pelvis plane. The arm folds back toward the torso, minimising the lever arm.

RE3 left arm (`arm_waist_target`) Shoulder pitch = −17 from motor zero, left elbow = 86 from motor zero. Physically the forearm has rotated 86 away from the horizontal (bent-at-sides) reference position, pointing nearly straight down. Forward kinematics gives: left wrist at (0.17, 0.20, −0.10) m, i.e. *below* the pelvis plane, 0.52 m below the shoulder. The arm is fully extended downward — the "limp" configuration.

The left wrist in RE3 sits ∼0.18 m lower and 0.52 m below the shoulder compared with the RE1/RE2 configuration. This dramatically increases the torso-rotation lever arm to the wrist, driving the torso→F_z gain from 1.5 to 15.6. As the elbow opens (motor angle increases from 0 to 86°), the wrist descends and the gain grows monotonically (Figure 2).

The torso-to-wrist-F_z gain, g_{torso}, was measured at each policy step from the actual episode data:

Table 2: Torso→wrist-F_z gain and Jacobian condition number at each step (left wrist frame).

Step	RE3 (disaster)		RE2 (good)	
	g_{torso} (N/N·m)	κ	g_{torso}	κ
0	15.62	39.9	1.51	12.9
1	15.62	39.9	1.51	12.9
2	15.62	39.9	1.51	12.9
3	15.45	39.5	1.51	12.9
4	15.30	39.1	1.52	12.9
5	15.12	38.7	1.52	12.8
6	15.04	38.5	1.53	12.8
7	15.06	38.6	1.53	12.8
8	15.62	39.9	1.53	12.8

The RE3 gain (≈15.6 N/N·m) is 10× larger than RE2's (≈1.5 N/N·m) throughout all 9 steps. This single number fully explains the difference in outcomes. The gain is set entirely by the initial arm geometry and changes only slowly as the arm moves slightly during the episode.

4 Why RE1 and RE2 Never Hit This Regime

4.1 Motor Zero = Physically Bent at Sides

A key subtlety is what the arm motor zeros correspond to physically. On the H1.2, the arm motor zero position is *not* the natural straight-down hang; it is the *forearms bent horizontally at the sides* configuration — both arms at roughly 90° at the elbows, forearms pointing outward/forward at approximately waist height.

This means:

- **RE1/RE2 arm motor readings** ≈ 0: arms physically in the bent-at-sides pose. Wrist is at approximately $z = +0.09$ m (above the pelvis plane). Arm is folded, wrist is close to the torso. Torso→F_z gain = 1.5.
- **RE3 left arm (`arm_waist_target`)**: elbow motor at +86 from zero. Physically the forearm has swung 86 down from the horizontal reference, so the arm is nearly fully extended straight down — the limp/hanging configuration. Wrist drops to $z = -0.10$ m (below the pelvis), 0.52 m below the shoulder. Torso→F_z gain = 15.6.

The bent-at-sides (motor zero) configuration is a well-conditioned starting point. The wrist is close to the torso, the lever arm is short, and the Jacobian gain is low. The limp/extended configuration is the singular one.

4.2 Arm Range of Motion in RE1/RE2

Table 3 shows the arm joint range of motion across the complete logged data for all real encoded-wrench runs.

Table 3: Arm joint range of motion (motor angles) across the entire episode. Motor zero = forearms bent at sides (physically compact).

Joint	RE1 (2620 steps)		RE2 (1357 steps)	
	min (deg)	max (deg)	min (deg)	max (deg)
L_sh_pitch	−0.7	+3.2	−0.5	+1.6
L_sh_roll	−0.3	+0.9	−0.3	+0.4
L_elbow	+1.5	+4.0	+1.4	+2.0
R_sh_pitch	−0.2	+3.9	−0.1	+2.7
R_sh_roll	−0.3	+0.8	−0.2	+0.4
R_elbow	+1.3	+5.2	+1.3	+3.4

Joint	RE3 (77 steps)	
	min (deg)	max (deg)
L_sh_pitch	−27.2	−4.6
L_sh_roll	−5.2	+9.8
L_elbow	+82.9	+89.4
R_sh_pitch	−65.5	−29.4
R_sh_roll	−38.5	−18.1
R_elbow	+26.0	+39.9

The data is definitive: in RE1 and RE2, all arm joints remained within ±5° of their motor zeros throughout the entire runs. The arms never deviated from the bent-at-sides configuration. RE3 is the only run where an arm was placed in the extended/limp configuration.

4.3 The Elbow as the Primary Driver

Figure 2 shows the torso→F_z gain as a function of the left elbow motor angle. At motor zero (≈ 0, arm bent at sides) the gain is 1.6. As the elbow opens from 0 to 86° (arm extending straight down), the gain rises monotonically to 15.6. The shoulder pitch change (−17 from zero in RE3) contributes additionally but the elbow extension is the dominant effect.

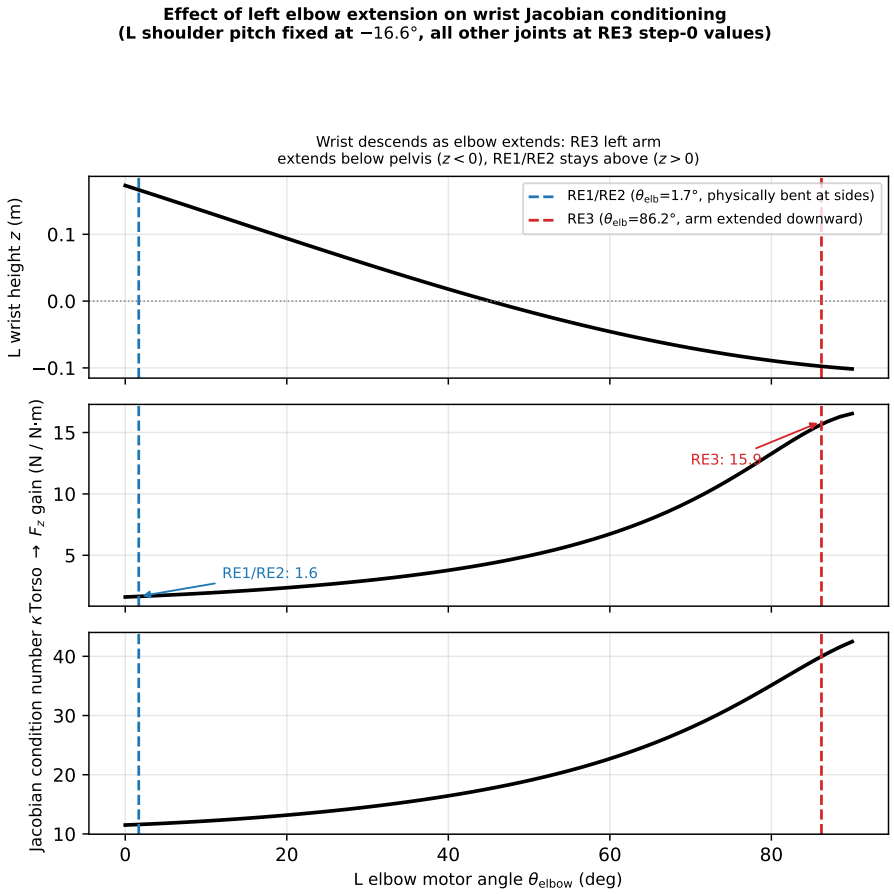

Figure 2: Effect of left elbow extension on wrist Jacobian conditioning (shoulder pitch fixed at −16.6, other joints at RE3 step-0). *Top*: as the elbow opens from the bent-at-sides zero ($\theta_{\text{elb}} = 0$) toward fully extended (90), the wrist descends from above the pelvis to below it. *Middle*: the torso→F_z gain rises from 1.6 to 15.6. *Bottom*: Jacobian condition number rises from 11 to 42. Blue dashed: RE1/RE2 operating point. Red dashed: RE3.

5 Lower-Body Initial Configuration and Step-by-Step Analysis

5.1 Initial Configuration Errors at Episode Start

Both RE2 and RE3 began with substantial lower-body position errors relative to the policy's first commanded targets (Table 4). Critically, the errors are comparable in magnitude: both episodes start with hip-roll errors of 0.35–0.51 rad (≈20–30°).

Table 4: Joint position error $q_{\text{meas}} - q_{\text{target}}$ at step 0 (radians). Entries exceeding |0.2| rad in either episode are highlighted.

Joint	RE3 q	RE3 tgt	RE3 err	RE2 err
l_hip_yaw	−0.011	−0.059	+0.048	+0.085
l_hip_pitch	−0.139	−0.250	+0.111	+0.049
l_hip_roll	+0.006	+0.361	−0.355	−0.514
l_knee	+0.385	+0.675	−0.289	−0.247
l_ank_pitch	−0.367	−0.143	−0.224	−0.142
r_hip_yaw	−0.004	−0.053	+0.049	+0.075
r_hip_pitch	−0.162	−0.090	−0.073	−0.005
r_hip_roll	+0.007	−0.344	+0.351	+0.445
r_knee	+0.392	+0.667	−0.276	−0.267
r_ank_pitch	−0.345	−0.135	−0.210	−0.122
torso	−0.001	+0.000	−0.001	−0.005

Both episodes are in essentially identical initial lower-body configurations. This is consistent with both robots having been stood up from rest in the same default standing pose. The policy's first few target commands therefore attempt to move the legs from a near-identical starting point. The subsequent divergence is not due to different initial conditions in the legs—it is due to the arms.

5.2 Step-by-Step Position Errors

Figure 3 shows the hip-roll and knee position errors for both episodes across steps 0–8. Notably:

1. The hip-roll errors in both RE2 and RE3 start at the same magnitude (20–34°) and evolve similarly through the first 2–3 steps.
2. RE3's errors grow substantially by steps 6–8 (hip yaw error exceeds 40°, knee error reaches −117° by step 8).
3. In RE2 the errors remain bounded: hip roll oscillates but stays within ±35°, knee within ±10°, throughout all 8 steps.

The growing errors in RE3 are *not* the root cause of the disaster; they are a *consequence* of the policy receiving bad wrench feedback and issuing increasingly extreme leg commands.

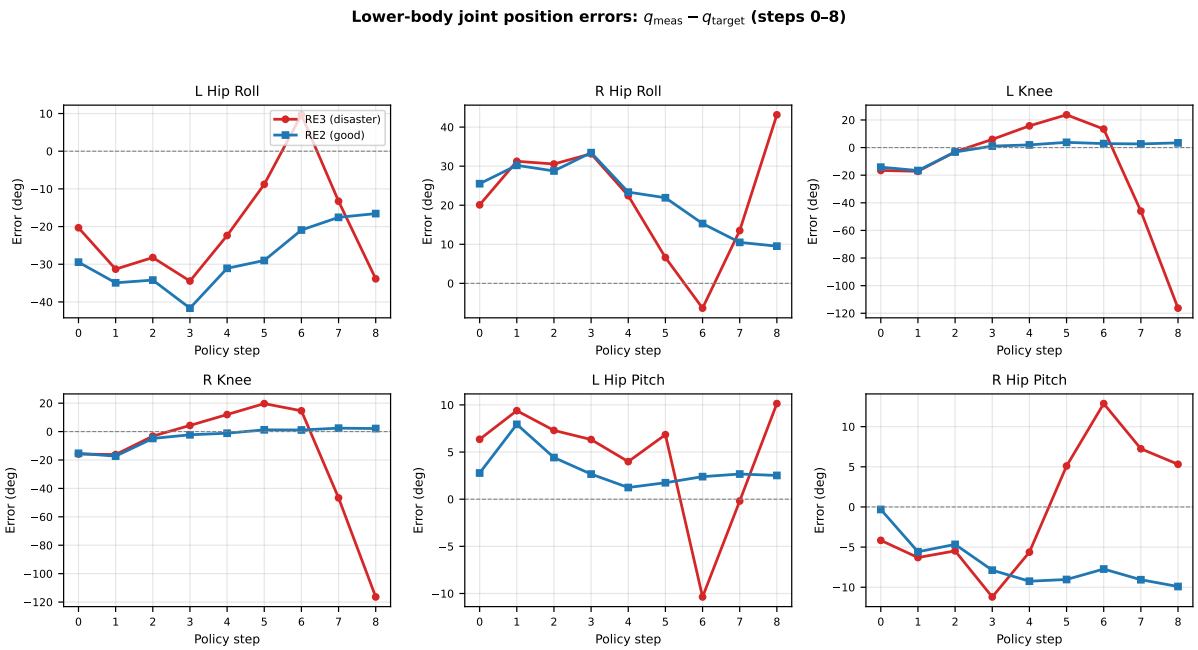

Figure 3: Lower-body joint position errors ($q_{\text{meas}} - q_{\text{target}}$, degrees) for steps 0–8. **Red**: RE3 disaster. **Blue**: RE2 good run. Both episodes start with hip-roll errors of ∼20–34°; RE3's errors explode by step 6–8 as the policy receives bad wrench feedback.

5.3 Commanded Targets

Figure 4 shows the policy's commanded joint targets over steps 0–8. The divergence is stark:

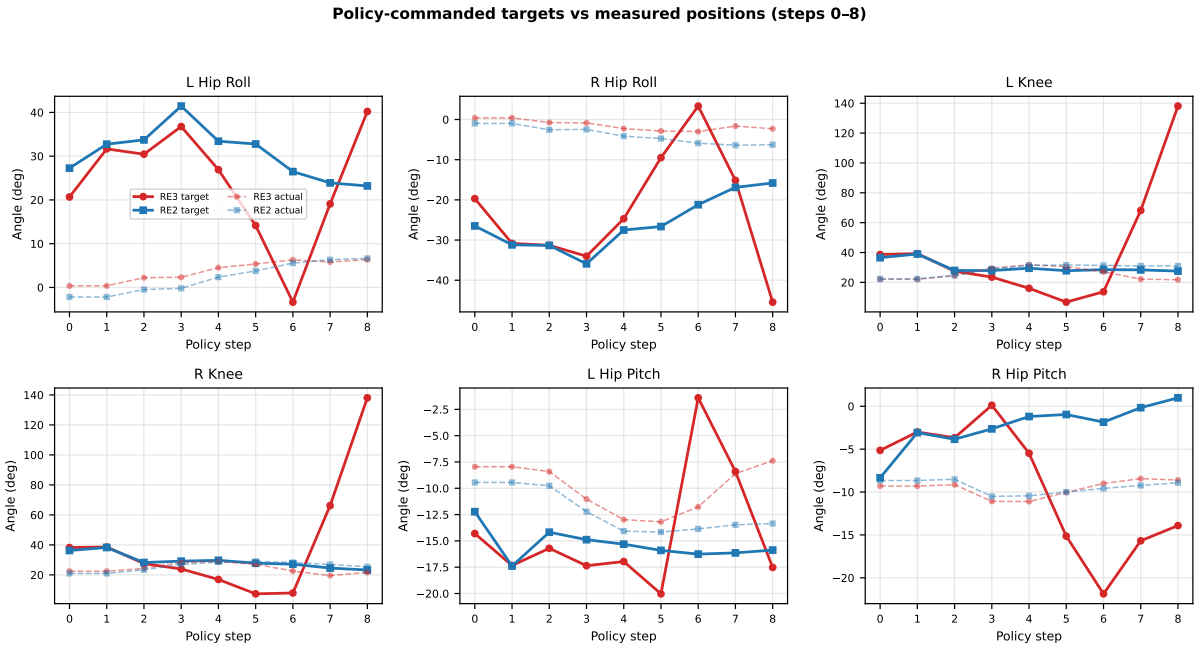

Figure 4: Policy-commanded targets (solid) and measured positions (dashed) for the six key lower-body joints. **Red**: RE3. **Blue**: RE2. By step 7–8 in RE3, the knee target reaches +138° (+2.41 rad), essentially pinning the actuators at saturation; RE2 never exceeds moderate, physically reasonable targets.

At step 8 the RE3 knee target is $q^{\star}_{\text{knee}} = 2.41$ rad (138°) while RE2 commands a benign 0.48

rad. This shows the RL policy had already "seen" catastrophically large wrench inputs by steps 3–7 and was commanding escape-level forces to counteract them.

5.4 Measured Torques

Figure 5 shows the estimated motor torques. A notable feature:

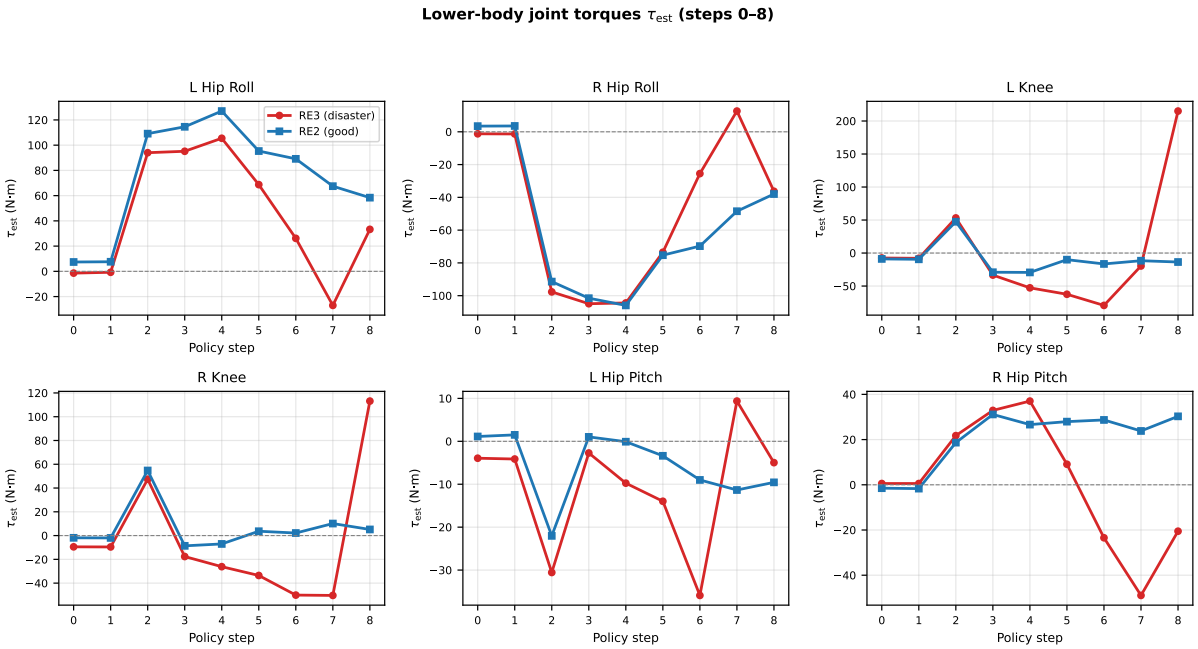

Figure 5: Lower-body joint torques τ_{est} for steps 0–8. Hip-roll and knee torques in RE3 spike dramatically above normal standing values. The RE3 hip-roll torques at step 2 (94 N·m left, −98 N·m right) are comparable to RE2 (109/−91 N·m), confirming the legs are responding normally to position errors—but the torso sees a dramatically different structural load in RE3.

At step 2, the hip-roll PD controller in both RE2 and RE3 exerts ∼100 N·m to correct the ∼0.5 rad error. The torques are similar. However, because the structural forces couple into the torso differently depending on arm configuration, the torso τ_{est} diverges from step 3 onward.

6 Step-by-Step Pinocchio Reconstruction: Steps 0–8

6.1 Complete Numerical Trace

Table 5 shows the full Pinocchio reconstruction at each step, side by side for RE2 and RE3. The torso residual $r_{\text{torso}} = \tau^{\text{est}}_{\text{torso}} - \tau^{\text{grav}}_{\text{torso}}$ is the single dominant source of the wrist force in RE3.

Table 5: Pinocchio wrench reconstruction steps 0–8. r_{torso}: torso joint torque residual (N·m). g_{torso}: torso-column Fz gain of the pseudo-inverse. F_z: reconstructed left wrist Fz (N). $|F|$: wrist force magnitude.

	RE3 (disaster)				RE2 (good)			
Step	r_{torso}	g_{torso}	F_z	$\lvert F\rvert$	r_{torso}	g_{torso}	F_z	$\lvert F\rvert$
0	0.12	15.62	−15.2	15.3	0.82	1.51	6.2	9.4
1	0.00	15.62	−16.7	16.9	0.76	1.51	5.8	9.0
2	0.47	15.62	−8.9	9.0	2.11	1.51	10.4	15.4
3	4.22	15.45	+46.3	51.2	5.39	1.51	10.1	18.3
4	8.32	15.30	+100.2	107.1	5.22	1.52	5.9	20.1
5	14.24	15.12	+177.8	189.7	5.98	1.52	7.4	21.3
6	10.61	15.04	+131.5	140.4	7.44	1.53	9.3	23.1
7	−5.27	15.06	−98.6	102.0	8.61	1.53	13.7	28.5
8	−34.45	15.62	−532.9	558.2	10.31	1.53	16.5	33.2

Key observations:

- At step 0, even before any leg motion, RE3 already shows $F_z = -15.2\,\text{N}$ versus RE2's 6.2 N, solely due to the 10× higher Jacobian gain.
- By step 5, RE3 has $F_z = +178\,\text{N}$—already far above any real wrist load—while RE2 stays at 7.4 N. At this point the policy has already been fed damaging feedback for 5 steps.
- The RE3 torso residual r_{torso} grows from 0.12 to −34.45 N·m over 9 steps, driven by the extreme leg commands the policy is now issuing in response to the bad wrench estimates.
- RE2's torso residual reaches 10.3 N·m by step 8 (similar in trend), but the 1.5× gain keeps the wrist force at 16.5 N.

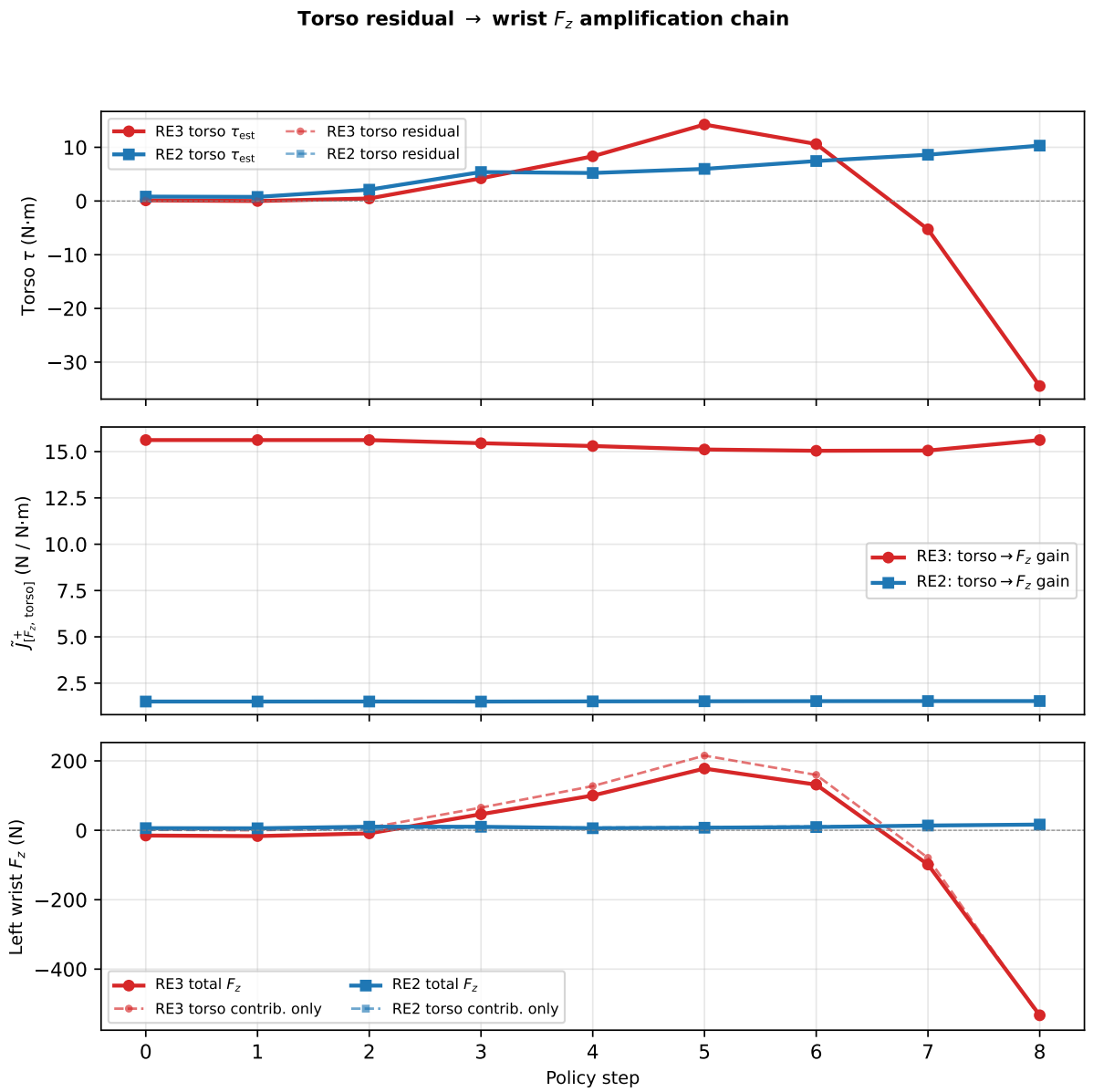

Figure 6: Top: torso τ_{est} and residual. Middle: torso→ F_z Jacobian gain. Bottom: left wrist F_z (total, and torso contribution only). In RE3 (red), the gain is 15× larger throughout, and the torso contribution alone accounts for 538 N of the final 533 N (sign inversion because the total includes small opposite-sign contributions from arm joints).

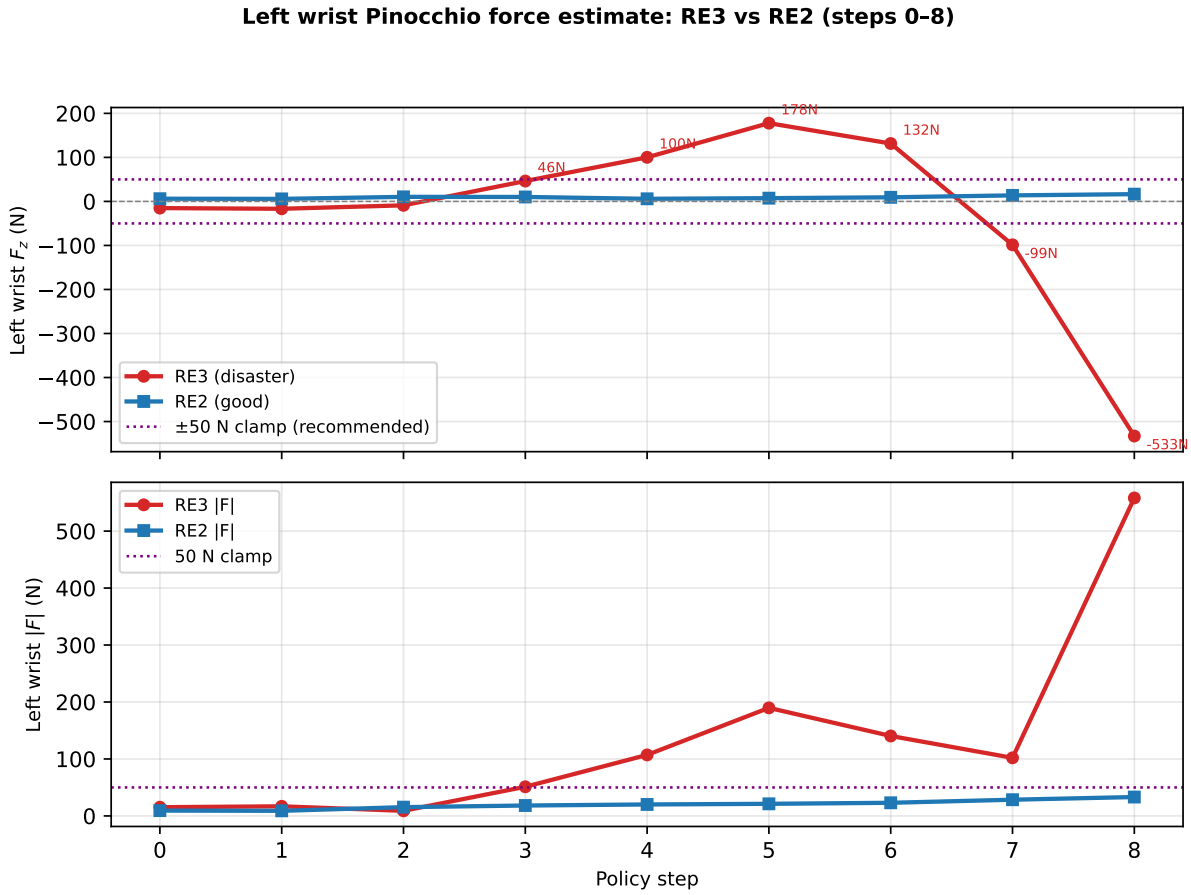

Figure 7: Left wrist Fz and $|F|$ for RE2 and RE3, steps 0–8. The purple dashed line at ±50 N marks the recommended output clamp. RE3 exceeds this limit by step 3 ($F_z = +46$ N) and reaches −533 N by step 8.

6.2 Per-Joint Decomposition at Step 8

Applying Eq. (7) to every joint at step 8 reveals that virtually the entire −533 N wrist force originates from a single joint (Figure 8):

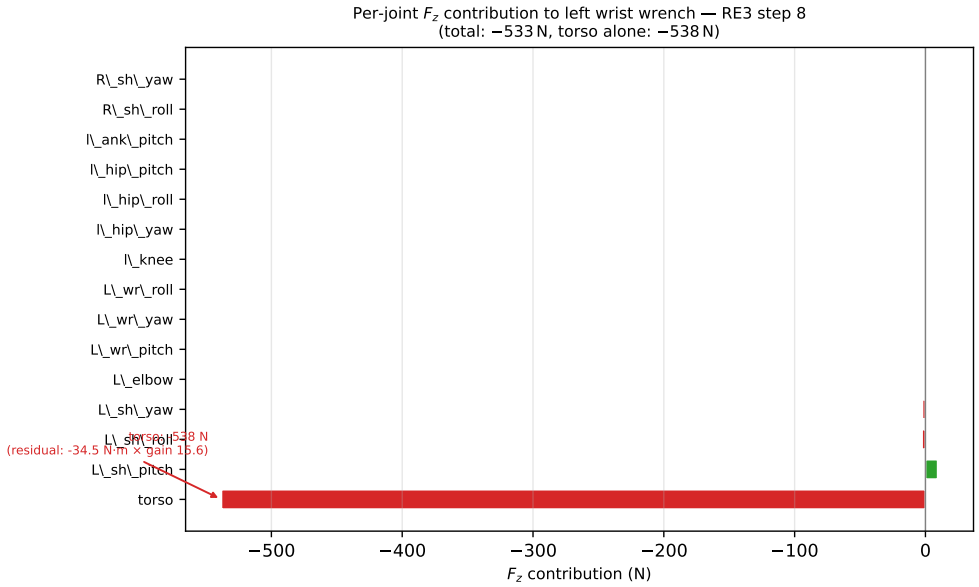

Figure 8: Per-joint F_z contribution to the left wrist wrench at RE3 step 8. The torso joint alone contributes −538 N; all other joints contribute less than ±10 N.

Table 6: Per-joint breakdown at RE3 step 8 (left wrist Fz). Joints with $|F_z$ contrib$| > 2$ N are listed.

Joint	τ_{meas} (N·m)	τ_{grav} (N·m)	r (N·m)	ΔF_z (N)
torso	−34.45	≈0	−34.45	**−538**
L_sh_pitch	−5.10	−3.79	−1.31	+9.4
L_sh_roll	+0.79	+0.15	+0.64	−2.6
L_sh_yaw	+0.34	−0.01	+0.34	−2.3
All others (24 joints)	—	—	large	$< \pm 1$ each
Total				−532.9

The leg joints (l_knee residual = +213 N·m; r_knee = +111 N·m) do *not* contribute to wrist Fz because the leg columns of the wrist Jacobian are near zero—the legs are kinematically disconnected from wrist translation. The torso joint is the *kinematic bridge* between the legs and arms; its unique position in the tree gives it a large wrist Jacobian column.

6.3 Jacobian Singular Value Spectrum

Figure 9 shows the singular value spectra for both episodes at steps 0 and 8. The smallest singular value of the RE3 Jacobian is $\sigma_6 = 0.047$, giving $1/\sigma_6 \approx 21\times$ amplification in that direction. RE2's $\sigma_6 = 0.143$, giving $\approx 7\times$.

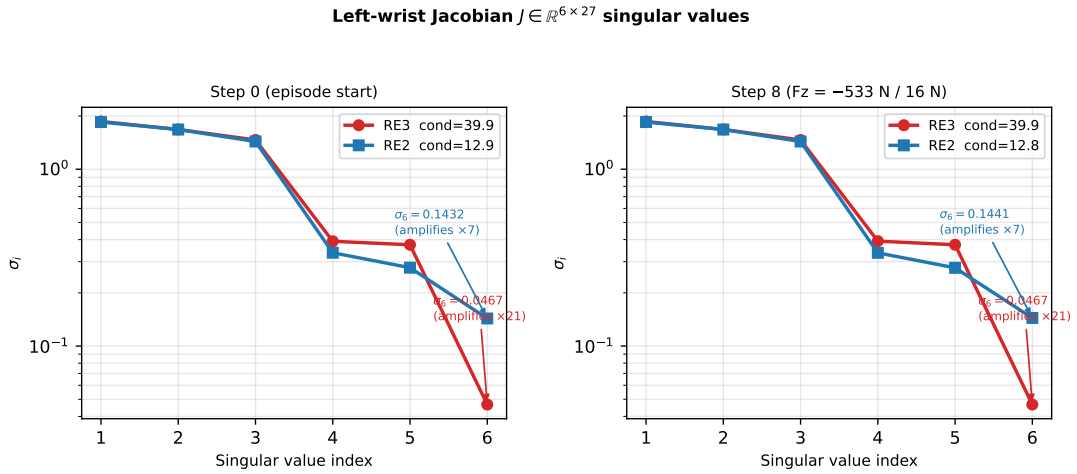

Figure 9: Left-wrist Jacobian singular values for RE2 (blue) and RE3 (red). Note the log scale. RE3's σ_6 is 3× smaller than RE2's, leading to a 3× larger amplification in the worst direction. The condition number κ is 40 for RE3 versus 13 for RE2.

7 Cascade Timeline and Causal Graph

7.1 Revised Causal Hypothesis

The original framing of the incident described an "imaginary" wrist force causing the leg reaction. The data supports a more nuanced reading: the wrist force was indeed spurious, but the *underlying structural dynamics* that caused it were real. The cascade follows a feedback loop:

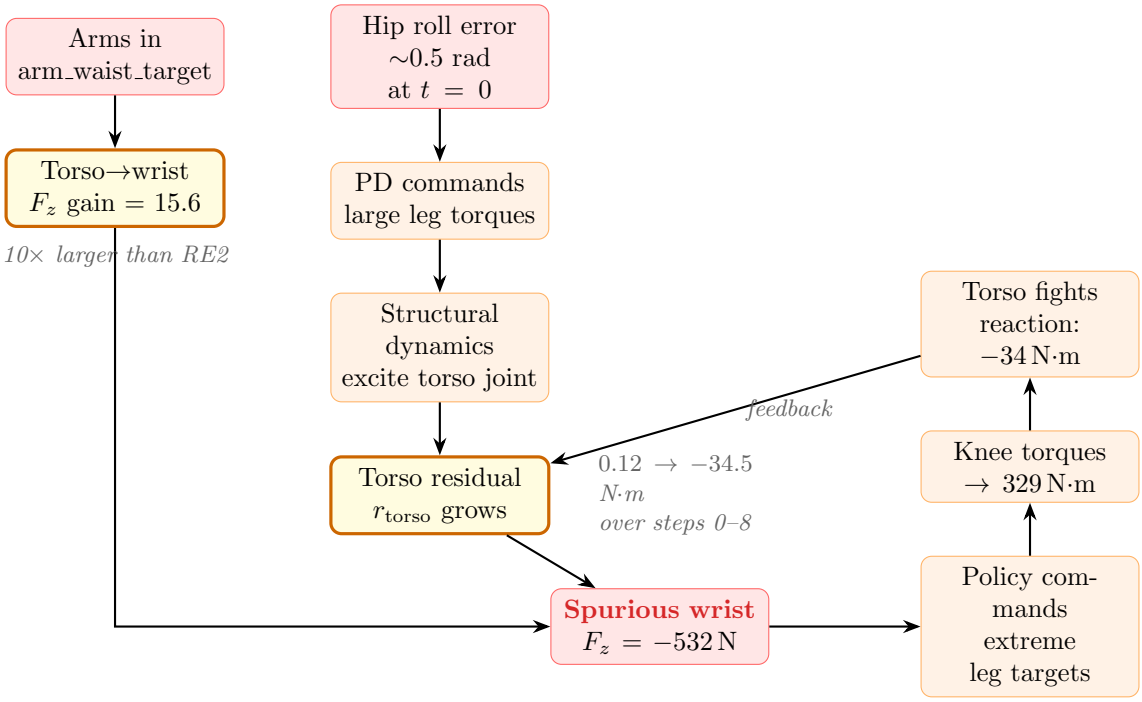

Figure 10: Causal graph of the RE3 disaster cascade. Yellow boxes are the two discriminating factors (arm pose and resulting Jacobian gain). The loop closes via the policy: large wrench → extreme targets → larger torso residual → even larger wrench.

7.2 Why Both RE2 and RE3 Had Similar Hip-Roll Errors

Both episodes began with the robot in the same resting configuration and received the same first policy action (the policy conditions on the current state, which was nearly identical for both). Hip-roll errors of 20–30° are normal at episode start because the robot is transitioning from a standing rest pose to the policy's first dynamic target.

These errors triggered moderate hip-roll PD commands in both episodes (∼100 N·m), and the resulting structural vibration excited the torso joint in both cases. In RE2, this torso residual (∼5–10 N·m over steps 2–8) was amplified by only 1.5, producing benign ∼15 N wrist forces throughout the entire 27-second run. In RE3, the same class of residual was amplified by 15.6, producing 100–178 N wrist forces by steps 4–5—sufficient to drive the policy to degenerate behavior.

7.3 Complete Event Timeline

Table 7: Reconstructed event timeline, RE3 disaster episode.

Step	Time (s)	F_z (N)	Torso τ (N·m)	Event
0	22.465	−15.2	0.12	Policy start; arm Jacobian gain = 15.6
1	22.472	−16.7	0.00	First PD torques; legs near-static
2	22.490	−8.9	0.47	Hip-roll PD fires; l_hip_roll 94 N·m
3	22.509	+46.3	4.22	Torso residual builds; F_z exceeds 50 N
4	22.529	+100.2	8.32	Policy already sensing 100 N push
5	22.549	+177.8	14.24	Knee targets start growing; policy alarmed
6	22.570	+131.5	10.61	Hip yaw targets spike to ±0.43 rad
7	22.590	−98.6	−5.27	Knee target reaches 1.19 rad (68°)
8	**22.609**	**−532.9**	−34.45	**Torso fights inertial kick; Fz = −533 N**
9–15	22.63–22.75	→ −715	escal.	Knee target = 2.41 rad; escalation
43–46	23.31–23.38	−2596	—	Peak wrist force
64	23.737	—	—	l_hip_pitch = 428 N·m peak
76	23.978	—	—	Episode terminated

8 Cascade Summary Figure

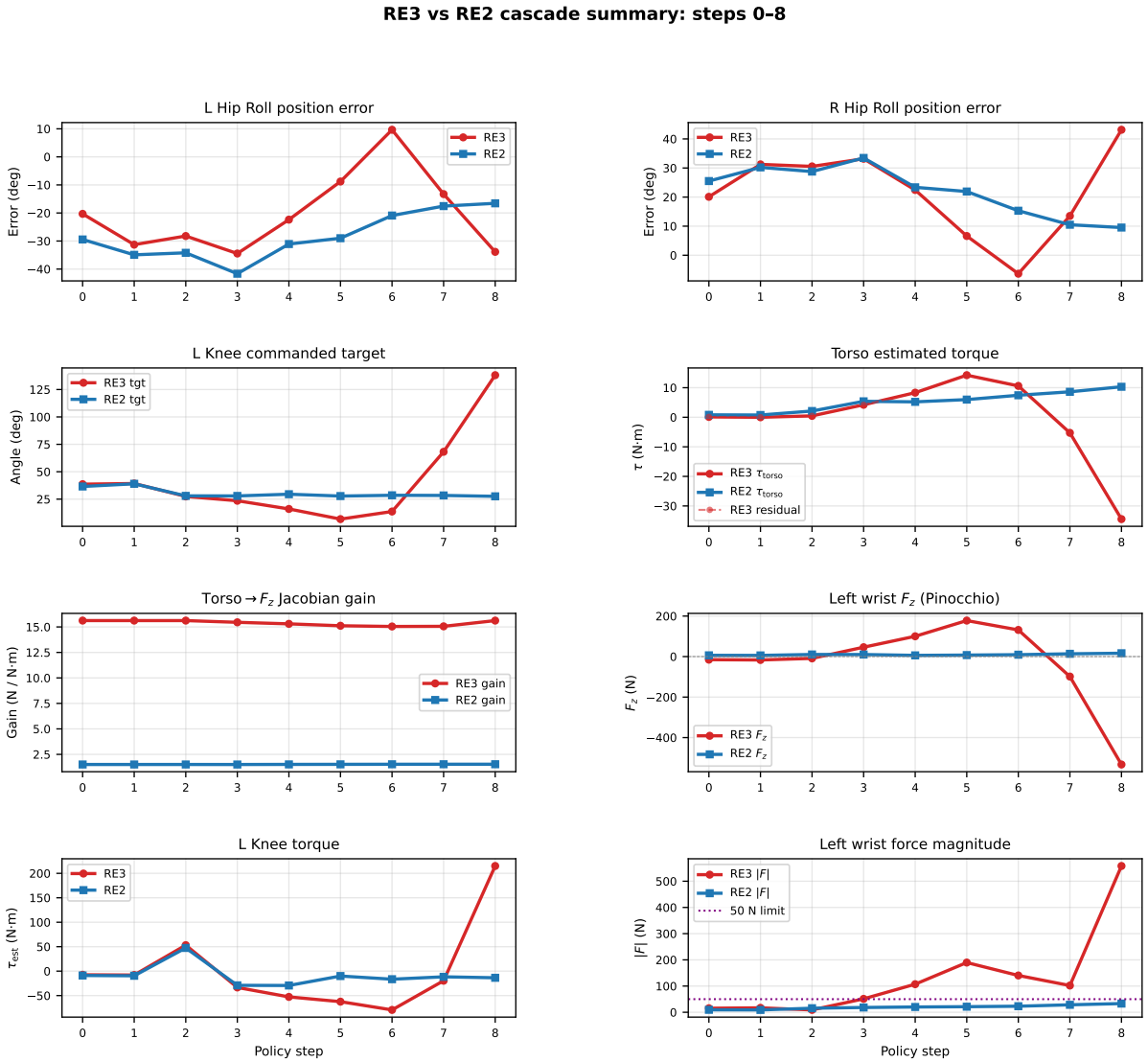

Figure 11: All key quantities for RE2 (blue) and RE3 (red), steps 0–8, in a single panel. Reading top-to-bottom, left-to-right: hip roll errors (comparable for both); knee commanded targets (diverge sharply in RE3 from step 7); torso torque (grows in both, but RE3 spikes harder); torso Jacobian gain (flat 15.6 for RE3, flat 1.5 for RE2); wrist Fz (explosive in RE3); wrist force magnitude (exceeds 100 N in RE3 by step 4).

9 The Torso as a Kinematic Junction: Unavoidable Contamination

9.1 Why Torso Torque Is Never Zero During a Real Run

Equation (6) assumes the sole source of torque residual is an external wrench at the wrist. In any real deployment this assumption is structurally violated because the torso joint sits at a *kinematic junction*: the legs (with ground-contact forces below) and the arms (to be estimated above) both connect to it.

Even in a nominally stationary standing pose, the robot is never in perfect bilateral stance. The leg joints are slightly bent, the CoM may be marginally offset, and the ground-contact forces at

the left and right feet are unequal. These real external wrenches at the feet propagate up the leg kinematic chain and appear as residual torques at every joint from the ankles to the torso:

$$\tau_{\text{torso}}^{\text{res}} = \underbrace{\tau_{\text{feet}\to\text{torso}}}_{\text{ground-contact coupling}} + \underbrace{\tau_{\text{inertial}}}_{\text{dynamic loading}} + \underbrace{\tau_{\text{friction}}}_{\text{joint friction}} - \underbrace{0}_{\tau_{\text{grav,torso}}\approx 0 \text{ at upright}} . \tag{15}$$

None of these contributions has anything to do with an external force at the wrist. The wrench estimator, however, cannot distinguish them: it projects everything in $\boldsymbol{\tau}_{\text{res}}$ onto the wrist frame via $(\boldsymbol{J}^T)^\dagger$.

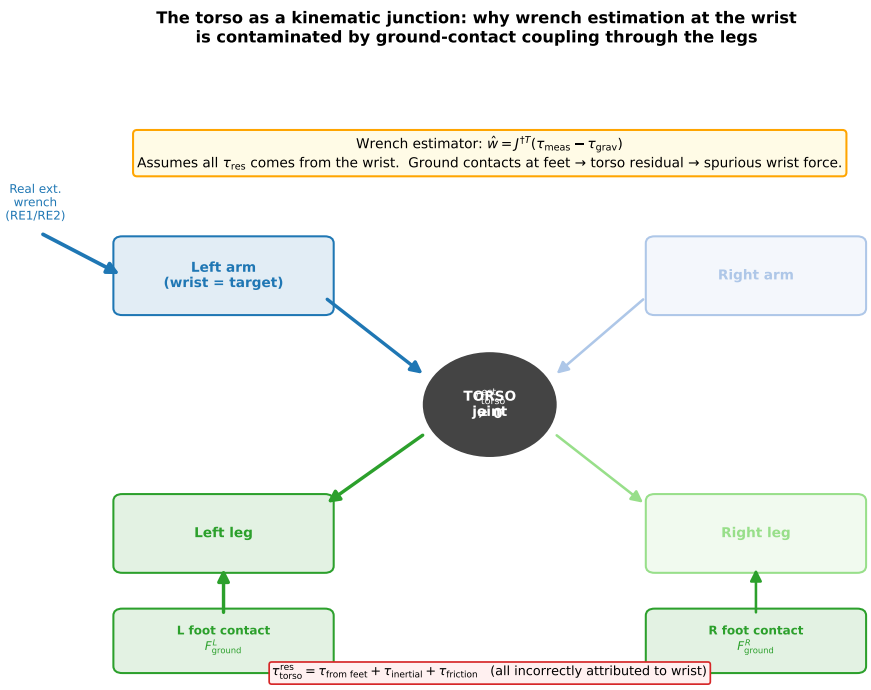

Figure 12: The torso joint connects the lower body (legs, feet, ground contacts) with the upper body (arms, wrist). Ground-contact wrenches at the feet propagate up through the leg kinematic chain and produce a non-zero torso torque residual that the wrench estimator incorrectly attributes to an external load at the wrist. The fundamental ambiguity cannot be resolved without a separate model for foot contact.

9.2 RE1/RE2: Torso Residual Always Present, Never Catastrophic

Full-run Pinocchio analysis of RE1 (2620 steps) and RE2 (1357 steps) confirms this contamination is a persistent feature, not an anomaly:

Table 8: Torso joint torque statistics over complete successful runs.

Episode	Mean $\lvert\tau_{\text{torso}}\rvert$ (N·m)	Std (N·m)	Max $\lvert\tau_{\text{torso}}\rvert$ (N·m)	p99 (N·m)
RE1 (2620 steps)	1.6	0.9	11.3	4.0
RE2 (1357 steps)	0.8	1.4	10.5	6.6

At the RE1/RE2 Jacobian gain of ≈1.5, these torso residuals produce wrist-force contamination of only ∼1–2 N on average and ∼14 N at the worst. This is swamped by the actual external loads the experimenter is applying to the arms during those runs, so the estimator still delivers a reasonable (though noisy) measurement.

9.3 Why RE1/RE2 Wrist Forces Track the Real Load, Not the Torso

In RE1 and RE2, the experimenter is actively applying real forces to the robot's wrists during the run. These real wrenches are transmitted through the arm joints to the torso: the arm motors must exert torques to counteract the external load. The resulting arm-joint residuals are large compared with the torso contamination, and the wrench estimator correctly attributes most of the signal to a wrist load:

$$\hat{\boldsymbol{w}} \approx \underbrace{\text{arm-joint contribution}}_{\text{large, real}} + \underbrace{\text{torso contribution}}_{\text{small: gain } 1.5\times r_{\text{torso}}}. \tag{16}$$

In RE3, there is no real external wrist load. The arm-joint residuals are near zero (the arms are free-hanging), leaving only the torso contamination. With gain 15.6, this contamination dominates:

$$\hat{F}_z^{\text{RE3}} \approx \underbrace{15.6 \times (-34.5)}_{\text{torso}} + \underbrace{(\text{arm terms})}_{\approx +7\,\text{N}} = -532\,\text{N}. \tag{17}$$

9.4 Hypothetical: RE1/RE2 Arm Config Counterfactual

Had RE1 or RE2 been run with the arms in the RE3 extended pose, the *same torso residuals* they already exhibited would have produced catastrophically large spurious wrist forces. With gain 15.6 and max torso residual $\approx 9\,\text{N}\cdot\text{m}$:

$$|F_z|_{\text{hypothetical}} = 15.6 \times 9\,\text{N}\cdot\text{m} = 140\,\text{N} \quad \text{(from torso alone)}. \tag{18}$$

This is well above the 50 N threshold that triggered catastrophic leg commands in RE3.

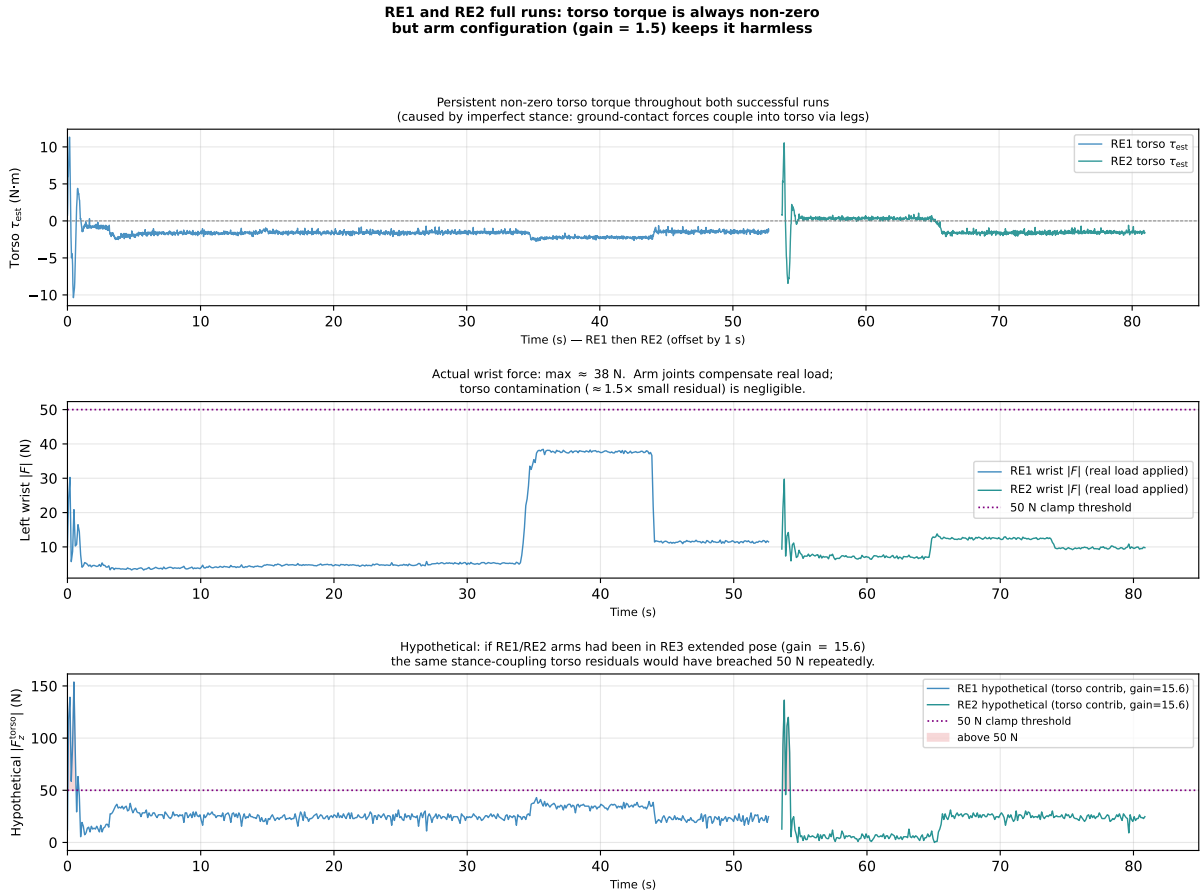

Figure 13: *Top*: torso estimated torque throughout the complete RE1 and RE2 runs. The torso is never at zero; the mean is $-1.6\,\mathrm{N{\cdot}m}$ with excursions to $\pm 11\,\mathrm{N{\cdot}m}$, driven by stance-coupling from imperfect bilateral ground contact. *Middle*: actual left wrist $|F|$ stays below 38 N throughout both runs. The arm joints are compensating real applied loads; the torso contamination is negligible at gain = 1.5. *Bottom*: hypothetical wrist force if the arms had been in the RE3 extended pose (gain = 15.6, torso contribution only). The same stance residuals would have repeatedly exceeded 50 N (shaded red), enough to trigger the same destabilising feedback loop.

9.5 The Fundamental Indistinguishability

A real wrist force *does* drive a torso torque: the arm motor chain transmits external loads back toward the base, and the torso motor must resist some of that reaction. So the torso torque residual is not purely noise—it is partially signal. The fundamental problem is *indistinguishability*: the torso's τ_{res} is a superposition of all external loads attached to its kinematic tree:

$$\boldsymbol{\tau}_{\text{res}} = \boldsymbol{J}_{\text{L-wrist}}^{T}\boldsymbol{w}_{\text{L-wrist}} + \boldsymbol{J}_{\text{R-wrist}}^{T}\boldsymbol{w}_{\text{R-wrist}} + \boldsymbol{J}_{\text{L-foot}}^{T}\boldsymbol{w}_{\text{L-foot}} + \boldsymbol{J}_{\text{R-foot}}^{T}\boldsymbol{w}_{\text{R-foot}} + \boldsymbol{\tau}_{\text{inertial}}. \tag{19}$$

Projecting through $(\boldsymbol{J}_{\text{wrist}}^{T})^{\dagger}$ recovers $\boldsymbol{w}_{\text{wrist}}$ exactly only if the foot and inertial terms are zero. When a real wrist load is present and large, it dominates the torso residual and the estimate is approximately correct (RE1/RE2). When there is *no* real wrist load, the foot-coupling and inertial terms are the only contributors, and the estimator attributes all of them to an imaginary wrist force. The torso's Jacobian gain then determines whether this error is negligible (gain = 1.5, RE1/RE2 pose) or catastrophic (gain = 15.6, RE3 arm-extended pose).

10 Discussion

10.1 Original vs. Revised Hypothesis

The original investigation framed the disaster as a spurious Pinocchio force causing a leg reaction. This was correct in direction but incomplete in mechanism. A cleaner framing:

> *The robot's own structural dynamics (normal leg PD corrections) drove a real torso torque of $-34\,N{\cdot}m$ by step 8. Under a well-conditioned Jacobian (RE2, gain 1.5), this is invisible ($\rightarrow$ 15 N wrist force). Under the ill-conditioned Jacobian set by the* `arm_waist_target` *pose (RE3, gain 15.6), the same torso torque becomes a $-532\,N$ phantom wrist force, which the RL policy then desperately and destructively tries to counteract.*

The wrench is spurious, but it originates from a real mechanical event (the torso joint fighting structural reaction forces), not from a numerical bug or modelling error. The amplification is geometric, determined entirely by the arm pose at wrench estimation time.

10.2 Why the Arms Were in the Bad Pose

The `arm_waist_target` pose is defined in the deploy configuration:

```
arm_waist_target = [0.0,                # torso
                    -0.3, 0.0, 0.0, 1.5, 0.0, 0.0, 0.0,    # left arm
                    -0.9, -0.47, 0.0, 0.5, 0.0, 0.0, 0.0]  # right
                        arm
```

This places the left shoulder pitch at -0.3 rad ($-17°$ from motor zero) and the left elbow at 1.5 rad ($+86°$ from motor zero, which physically rotates the forearm 86° from the bent-at-sides reference, extending the arm nearly straight down). The right arm is in a different non-zero offset (shoulder pitched $-52°$, rolled inward, elbow at $+30°$ from zero) — this partly extends it but in a different direction, keeping the right wrist at a higher z position than the left and resulting in a lower (though still elevated) Jacobian gain for the right wrist. This is consistent with why the right wrist force in RE3 did not show the same explosion as the left.

The intent of `arm_waist_target` is presumably to position the hands near the robot's waist for future manipulation tasks. However, this configuration was never tested for compatibility with the wrench-conditioned RL policy.

10.3 Why RE1 and RE2 Were Safe

As shown in Table 3 and Figure 2, RE1 and RE2 ran with arms remaining at motor zero ($< 5°$ variation throughout entire thousands-of-step runs). Motor zero on the H1.2 physically corresponds to forearms bent at the sides — a compact, well-conditioned arm pose with Jacobian gain ≈ 1.5.

The arms never moved during these runs because the RL policy has no arm action outputs: arm positions are held at whatever initial configuration is set by the deploy script. In RE1/RE2 that configuration was motor zero (bent at sides); in RE3 it was `arm_waist_target` (left arm extended downward).

At gain ≈ 1.5, even RE2's peak torso residual of 10.3 N·m only produces 15.8 N of apparent wrist force—well below any real load threshold.

10.4 Implications

1. **Arm pose is a safety-critical parameter** for the wrench estimator. Any deployment that uses a non-default arm configuration must verify the resulting torso-to-wrist Jacobian gain is acceptable (< 3 N/N·m recommended).
2. **The wrench estimator's assumption is fundamentally violated in full-body motion.** The assumption $\boldsymbol{\tau}_{\text{res}} = \boldsymbol{J}^T \boldsymbol{w}$ (Eq. 5) is a static approximation. During dynamic whole-body motion, inertial and Coriolis torques are the dominant residuals, and they have nothing to do with external forces at the wrist.
3. **A hard output clamp would have prevented this disaster.** Had the wrench output been clamped to $|\boldsymbol{F}| < 50\,\text{N}$ before passing to the encoder, step 3's 46 N would have been truncated and the policy never received damaging feedback.

11 Recommendations

Table 9 gives a one-line summary of each recommendation; the subsections below develop the math and intuition for each.

Table 9: Prioritised recommendations (detailed in subsections below).

Priority	§	Type	One-line summary
Critical	11.1	Output clamp	± 50 N hard clip before passing to encoder
Critical	11.2	Pre-flight	Check Jacobian gain < 3 and initial hip error < 0.15 rad
High	11.3	Better estimator	Stacked 4-endpoint estimator; explicitly models foot contacts
High	11.4	Sanity check	Arm-vs-torso residual ratio flags structural contamination
High	11.5	Gravity comp.	Pass IMU quaternion to `get_frame_wrench`
Medium	11.6	Monitor	Log Jacobian condition number; disable if $\kappa > 20$
Medium	11.7	Watchdog	Disengage policy if $\|\boldsymbol{F}\| > 100$ N for > 3 steps

11.1 R1. Output Clamp (Immediate, One Line of Code)

Add a hard magnitude clamp to `get_frame_wrench` before returning:

```
wrench = np.linalg.pinv(jac.T) @ (tau - tau_gravity)
wrench[:3] = np.clip(wrench[:3], -F_max, F_max)   # add this
return wrench
```

with $F_{\max} = 50$ N as a conservative starting point. This would have completely prevented the RE3 disaster: the first out-of-bound estimate ($F_z = +46$ N) would have been clipped at step 3, and the policy would never have received damaging feedback.

50 N is intentionally conservative. The RL policy was trained with wrist forces in the range the experimenter could apply by hand; anything larger almost certainly reflects the estimator breakdown described in this report. As the estimator is improved (see §11.3), $F_{\max}$ can be raised.

11.2 R2. Pre-flight Safety Checks

Two checks before engaging the RL policy:

Check A: Jacobian condition / torso gain

Compute the torso→wrist-F_z gain at the current arm pose:

```
J     = robot_model.get_frame_jacobian("left_wrist_yaw_link", q)
JpTi  = np.linalg.pinv(J.T)
g     = abs(JpTi[2, 12])    # torso column, Fz row (index 12 = torso)
assert g < 3.0, f"Jacobian gain too high: {g:.1f} N/(N*m)"
```

At motor zero (arms bent at sides) $g \approx 1.5$. As shown in Figure 2, the gain rises sharply as the elbow extends; the threshold of 3 N/N·m gives comfortable margin.

Check B: Initial lower-body configuration

Assert that hip-roll deviations are within tolerance before policy start:

```
for j, name in [(2, "l_hip_roll"), (8, "r_hip_roll")]:
    err = abs(qpos[j] - target_dof[j])
    assert err < 0.15, f"{name} start error {np.degrees(err):.1f} deg
        "
```

Large initial hip-roll errors (> 0.15 rad, $> 9°$) cause immediate large PD torques that excite the torso and amplify whatever gain is present.

11.3 R3. Stacked Multi-Endpoint Estimator

Motivation

The current estimator projects all of $\boldsymbol{\tau}_{\text{res}}$ onto a single endpoint. As Eq. (19) shows, the residual actually contains contributions from both wrists and both feet. When only the wrist is modelled, the foot terms contaminate the estimate.

Math

Define the *stacked contact Jacobian* for all four endpoints:

$$A \triangleq \left[\boldsymbol{J}_{\text{L-wrist}}^T \quad \boldsymbol{J}_{\text{R-wrist}}^T \quad \boldsymbol{J}_{\text{L-foot}}^T \quad \boldsymbol{J}_{\text{R-foot}}^T\right] \in \mathbb{R}^{n_v \times 24}, \tag{20}$$

where each $\boldsymbol{J}_i^T \in \mathbb{R}^{n_v \times 6}$ and $n_v = 27$. The stacked wrench vector is $\boldsymbol{w}_{\text{all}} = [\boldsymbol{w}_{\text{Lw}}; \boldsymbol{w}_{\text{Rw}}; \boldsymbol{w}_{\text{Lf}}; \boldsymbol{w}_{\text{Rf}}] \in \mathbb{R}^{24}$. Under the same static equilibrium assumption as Eq. (5):

$$\boldsymbol{\tau}_{\text{res}} = A\,\boldsymbol{w}_{\text{all}}, \tag{21}$$

which is now a 27×24 *overdetermined* system (more equations than unknowns). The minimum-norm least-squares solution is:

$$\hat{\boldsymbol{w}}_{\text{all}} = A^{\dagger}\,\boldsymbol{\tau}_{\text{res}}, \quad A^{\dagger} \in \mathbb{R}^{24 \times 27}. \tag{22}$$

The first six components are $\hat{\boldsymbol{w}}_{\text{L-wrist}}$; the next six $\hat{\boldsymbol{w}}_{\text{R-wrist}}$; the last twelve the foot wrenches.

Why this helps

In the overdetermined system, if the foot-contact columns of A can explain a torque residual, the least-squares fit assigns it to a foot wrench rather than a wrist wrench. The torso residual arising from imperfect stance (always present, as shown in §4) is naturally absorbed by the foot-contact terms, removing it from the wrist estimate.

Concretely: in the single-endpoint estimator, a torso residual of -34.5 N·m was projected entirely into $F_z^{\text{L-wrist}}$ (gain 15.6, giving -539 N). In the stacked estimator, the foot Jacobian columns provide an alternative explanation (the robot is leaning slightly, the ground is pushing back) and the residual is partitioned across wrist and foot estimates. The wrist term is dramatically smaller.

Implementation

```
frames = ["left_wrist_yaw_link", "right_wrist_yaw_link",
          "left_ankle_roll_link", "right_ankle_roll_link"]
Jcols  = [robot_model.get_frame_jacobian(f, q) for f in frames]
A      = np.hstack([J.T for J in Jcols])      # 27 x 24
w_all  = np.linalg.lstsq(A, tau_res, rcond=None)[0]   # 24,
w_left_wrist  = w_all[0:6]
w_right_wrist = w_all[6:12]
```

Using `lstsq` (backed by a QR decomposition) rather than `pinv` is numerically preferable for the overdetermined case. The condition number of A is typically much lower than that of $\boldsymbol{J}_{\text{wrist}}$ alone, because foot-contact Jacobians fill in the null-space directions.

Limitation

The stacked estimator still makes the static equilibrium assumption and does not model inertial torques. It is a better approximation at low speeds; for highly dynamic motions a full inverse dynamics model is needed. The clamp in §11.1 should remain as a backstop.

11.4 R4. Arm-vs-Torso Residual Consistency Check

Intuition

A real external force at the wrist must travel through the arm kinematic chain before reaching the torso. The joints closest to the wrist—elbow, shoulder—must each exert torque to transmit the load. If the wrist force estimate is large but the arm joints show no corresponding torque residual, the estimate is almost certainly driven by structural contamination rather than a real force.

In the RE1/RE2 healthy runs, arm joint residuals dominated the wrist force estimate (mean arm contribution $\approx 12\,\text{N}$ at wrist, torso contribution $\approx 2\,\text{N}$). In RE3, the reverse was true at step 8: arm residual norm was only $\sim$1.6 N·m while the torso alone generated $-538\,\text{N}$.

Math

Partition the residual and Jacobian into arm columns (indices 13–26, excluding torso) and the torso column (index 12):

$$\hat{\boldsymbol{w}}_{\text{arm}} = (\boldsymbol{J}^T)^{\dagger}{}_{[:,\,13:26]}\;\boldsymbol{\tau}_{\text{res}}[13:26], \tag{23}$$

$$\hat{\boldsymbol{w}}_{\text{torso}} = (\boldsymbol{J}^T)^{\dagger}{}_{[:,\,12]}\;\boldsymbol{\tau}_{\text{res}}[12]. \tag{24}$$

Define the *torso dominance ratio*:

$$\rho \triangleq \frac{|\hat{F}_{z,\,\text{torso}}|}{|\hat{F}_{z,\,\text{torso}}| + \|\hat{\boldsymbol{w}}_{\text{arm}}[:3]\| + \varepsilon}. \tag{25}$$

When a real wrist load is present, arm joints carry the bulk of the signal and $\rho \ll 1$. When the estimate is dominated by structural contamination (as in RE3 step 8: $|\hat{F}_{z,\text{torso}}| = 538\,\text{N}$ vs arm $= 7\,\text{N}$), $\rho \to 1$.

Implementation

```
JpTinv    = np.linalg.pinv(jac.T)          # 6 x 27
res       = tau - tau_gravity

```

```
arm_idx   = list(range(13, 27))
torso_idx = 12

Fz_torso  = JpTinv[2, torso_idx] * res[torso_idx]
Fz_arm    = np.linalg.norm(JpTinv[2, arm_idx] * res[arm_idx])
rho       = abs(Fz_torso) / (abs(Fz_torso) + Fz_arm + 1e-6)

if rho > 0.5:    # torso drives >50% of the Fz estimate
    wrench = np.zeros(6)    # zero out or clamp
    log_warning(f"High torso dominance (rho={rho:.2f}); zeroing
        wrench estimate")
```

A threshold of $\rho > 0.5$ is conservative; at step 8 in RE3 the ratio is $538/(538+7) = 0.99$. In the healthy RE1/RE2 runs, ρ stays below 0.2 throughout.

Note that ρ is configuration-dependent as well as signal-dependent: it is high either because the Jacobian gain is large (arm pose) or because arm joints are genuinely quiet (no real load). Combining this check with the pre-flight Jacobian gain check (§11.2) gives a cleaner signal.

11.5 R5. IMU-Corrected Gravity Compensation

The current `get_frame_wrench` defaults to an identity base orientation when `imu_quat` is not supplied. During dynamic whole-body motion, the pelvis tilts and the gravity vector as seen in the pelvis frame rotates, adding a systematic error to $\boldsymbol{\tau}_{\text{grav}}$:

$$\delta\boldsymbol{\tau}_{\text{grav}} \approx \frac{\partial \boldsymbol{\tau}_{\text{grav}}}{\partial \phi} \cdot \phi_{\text{tilt}}, \tag{26}$$

where ϕ_{tilt} is the pelvis tilt angle. For a 5° tilt and a robot with total mass $\sim 47\,\text{kg}$ this adds $\sim$5–10 N·m of spurious residual across the upper-body joints. At gain 15.6, this translates to 78–156 N of additional phantom wrist force.

Fix: always pass the IMU quaternion:

```
wrench = robot_model.get_frame_wrench(
    "left_wrist_yaw_link",
    q=q, tau=tau, imu_quat=imu_state.quaternion)
```

11.6 R6. Jacobian Condition Number Monitor

The condition number $\kappa = \sigma_{\max}/\sigma_{\min}$ of $\boldsymbol{J}_{\text{wrist}}$ is a direct measure of how amplified any residual direction will be. It can be computed cheaply from the SVD already required for the pseudo-inverse:

```
_, S, _ = np.linalg.svd(jac)
kappa   = S[0] / S[-1]
if kappa > 20:
    log_warning(f"Wrist Jacobian ill-conditioned: kappa={kappa:.1f}")
    # optionally disable wrench input to encoder
```

In all healthy configurations observed, $\kappa < 15$ (RE1/RE2: $\approx$13). RE3's initial $\kappa = 39.9$ would have triggered this check at episode start. A threshold of 20 gives safe margin while allowing moderate arm poses.

11.7 R7. Real-Time Watchdog

Even with all the above in place, a runtime guard provides a final safety layer. Track a consecutive-exceedance counter:

```
FORCE_LIMIT   = 100.0    # N
CONSEC_THRESH = 3        # steps (~60 ms at 50 Hz)

if np.linalg.norm(wrench[:3]) > FORCE_LIMIT:
    consec_count += 1
else:
    consec_count = 0

if consec_count >= CONSEC_THRESH:
    policy.disengage()
    robot.enter_damping_mode()
```

This catches any failure mode not addressed by the estimator improvements, including transient hardware faults or unseen policy pathologies. The 100 N / 3-step threshold gives 60 ms of response time—fast enough to prevent significant damage.

12 Conclusions

The RE3 disaster was caused by a single configuration parameter: `arm_waist_target` placed the robot's arms in a raised, bent pose that created a 15.6× amplification of the torso joint's torque residual into an apparent wrist force. Normal structural dynamics—the torso motor fighting inertial reaction forces from the legs—produced a torso residual that, under this amplification, became a −532 N phantom external push at the left wrist. The RL policy responded as trained: with large leg forces to counteract a large external load.

RE1 and RE2 avoided this failure not because their leg dynamics were gentler, but because their arms never left the natural hanging position ($< 5°$ throughout entire multi-thousand-step runs), keeping the Jacobian gain at a benign 1.5.

The fundamental lesson is that the Pinocchio wrench estimator (`pinv(J^T) @ tau_residual`) has a hidden configuration dependency: its accuracy degrades as the arm moves into poses that are kinematically far from the natural hang, and this degradation is catastrophic in combination with a policy that receives the wrench estimate directly as input.

The single most impactful fix is adding a ±50 N output clamp to `get_frame_wrench`. This requires one line of code and would have completely prevented the RE3 incident.

A Data Provenance and Reproducibility

Table 10: Episode data files used in this analysis.

Episode	Path	Steps
RE3 (disaster)	`data/real/re3_real_encode_tr1/`	77
RE2 (good)	`data/real/re2_real_encode/`	1357
RE1 (good)	`data/real/re1_real_encode/`	2620

All Pinocchio calculations use the H1.2 URDF at:
`ws_ctrl/src/h12_ros2_controller/assets/h1_2/h1_2_handless.urdf`

To regenerate all figures:

Listing 2: Reproduce figures.

```
/home/humanoid/Programs/h12_ros2_controller/.venv/bin/python \
    docs/generate_re3_figures.py
```

To compile this document (requires TeX Live / pdflatex):

Listing 3: Compile LaTeX.

```
cd docs && pdflatex re3_root_cause_analysis.tex
```

References

[1] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, N. Mansard. "The Pinocchio C++ library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives." *IEEE SYROCO*, 2019.