# Peg Insertion Simulation with PyBullet

## 1 Physics Engine & Simulation Design

The system is built around PyBullet as the core physics engine with a fixed time step $\Delta t = 1/240\ s$ and gravity $g = -9.8$ m/s$^2$. This configuration is applied once at environment construction (`env_peg_insertion`) and propagated to the constraint solver via `fixedTimeStep`, `numSolverIterations`, and `ERP`. Real-time pacing can optionally be enabled for teleoperation.

**Why this choice?**

- Fixed stepping ensures reproducible contact resolution and stabilizes peg–hole interactions.
- Bullet's LCP/PGS solver is robust when tuned for ERP and solver iterations, even for many simultaneous contact patches.

### 1.1 Task, Bodies, and Actuation

**Task.** The benchmark task is **rectangular peg insertion into a cuboid hole**. The Robotiq 2F-85 gripper is modeled as a kinematic base with mimic gear constraints that couple finger joints to a single actuated parent joint (`finger_joint`). The gear mapping is set up in `Robotiq85.__post_load__`.

**Bodies.** The scene contains:

- A static plane (URDF),
- A cuboid mesh (hole geometry),
- A rectangular peg mesh ($m \approx 0.35$ kg),
- The gripper base.

Dynamics are assigned consistently, and pad friction on contact links is raised to improve grasp stability (`env_peg_insertion`).

**Actuation.** The gripper base pose is commanded kinematically each step using `resetBasePositionAndOrientatio`
Jaw aperture is controlled via position control: a desired jaw separation (in meters) is mapped to the parent joint angle; child joints follow via gear constraints.

### 1.2 Control Modes

- **Scenario 1 (Tele-op):** Real-time stepping with a keyboard-driven teleoperation loop. A compact on-screen HUD shows key bindings (`run_scenario_1_teleop`).
- **Scenario 2 (Agent):** A scripted, phase-based controller that runs in fixed-step mode, using contact forces for decisions and an admittance-based insertion law (`run_scenario_2_agent`).

### 1.3 Contact Modeling & Forces

Bullet computes discrete contact points, with each point providing a normal force $F_n$ and a tangential friction force $F_t$ subject to the friction cone. In practice, normal forces are accessed via `p.getContactPoints(...)` [9] and summed; these are logged by the environment and used by the controller (`env_peg_insertion`, `simple_controlle`

A practical constitutive model is:

$$F_n \approx k\,\delta + c\,v_n, \tag{1}$$
$$\|F_t\| \le \mu F_n, \tag{2}$$

where $\delta$ is penetration depth, $v_n$ is the relative normal velocity, $k, c$ are Bullet's effective contact stiffness and damping parameters (often increased for crisper contacts), and $\mu$ is the lateral friction coefficient set per link or body.

### 1.4 Logging & Recording

At each step, the environment logs commanded and actual poses, jaw aperture, and summed contact forces for selected body pairs (`env_peg_insertion`).

For visualization, the `PBRecorder` can record:

- MP4 videos using Bullet's GUI logger (requires system `ffmpeg`),
- Fallback software recording using `imageio-ffmpeg`,
- GIFs from a fixed camera view.

## 2 Contacts & Unachievable Gripper Positions

### 2.1 Grasp Detection - force-based

A grasp is confirmed only when both finger pads apply sustained force:

$$F_{\text{left}} \ge F_{\min}, \quad F_{\text{right}} \ge F_{\min}$$

for $N$ consecutive frames:

$$\sum_{t=t_0}^{t_0+N} \mathbf{1}\big[F_{\text{left}}(t) > F_{\min} \land F_{\text{right}}(t) > F_{\min}\big] = N$$

This prevents false positives from transient touches. (See $\_gripper\_peg\_forces + framecounters$ in the agent.)

### 2.2 Slip Monitoring(follower consistency)

Peg motion must track gripper motion:

$$|\Delta z_g - \Delta z_p| \le \epsilon_z, \quad \|\Delta \mathbf{x}_g - \Delta \mathbf{x}_p\| \le \epsilon_{xy}$$

If violated, the jaw is micro-tightened and motion slowed.

### 2.3 Jaw Saturation

Jaw separation aperture $d$ is clamped:

$$d = \min\big(\max(d, d_{\min}), d_{\max}\big)$$

The env maps separation to the parent joint angle linearly, then gear constraints propagate to the child joints.

## 2.4 Admittance Insertion

For insertion (along $z$):

$$m\dot{v}_z + bv_z + k(z - z_{\text{ref}}) = F_d - F_{\text{meas}}$$

where $F_d$ is target force, $F_{\text{meas}}$ measured force, and $(m, b, k)$ virtual dynamics parameters.

Discretized (Euler forward):

$$v_z^{k+1} = v_z^k + \frac{\Delta t}{m}\Big((F_d - F_{\text{meas}}) - bv_z^k - k(z^k - z_{\text{ref}})\Big)$$

$$z^{k+1} = z^k + v_z^{k+1}\Delta t$$

Lateral correction using contact normal $\mathbf{n} = (n_x, n_y, n_z)$:

$$x_{\text{corr}} = x - \beta\frac{n_x}{\|\mathbf{n}_{xy}\|}\frac{F}{F_{\text{cap}}}, \quad y_{\text{corr}} = y - \beta\frac{n_y}{\|\mathbf{n}_{xy}\|}\frac{F}{F_{\text{cap}}}$$

## 2.5 Unachievable Poses

In the current implementation, the gripper base is directly **teleported** to the commanded pose in Cartesian space, bypassing kinematic constraints. Collisions at the target pose are then resolved by the solver: because strong contact stiffness/damping and high pad friction are applied, "impossible" configurations (e.g., gripper penetrating the cuboid) are pushed out by reaction forces rather than causing numerical blow-ups. The use of small integration steps and high solver iteration counts ensures these corrections remain stable.

However, this implicitly assumes that the gripper can always reach the commanded grasp pose, which may not hold in a realistic robot. Handling truly unachievable poses requires additional mechanisms:

- **Reachability constraints:** Instead of directly teleporting, inverse kinematics (IK) or Jacobian-based solvers can enforce joint limits and workspace boundaries:

  $$\min_q \ \|f(q) - x_d\|^2 \quad \text{s.t. } q_{\min} \le q \le q_{\max},$$

  where $f(q)$ maps joint configuration $q$ to end-effector pose $x_d$.
- **Projection to feasible space:** If no exact IK solution exists, the desired pose can be projected to the closest feasible pose under kinematic and collision constraints, reducing penetration before execution.
- **Motion planning fallback:** Sampling-based planners (RRT, PRM) or optimization-based planners can generate trajectories that avoid infeasible or colliding states.
- **Virtual compliance:** For cases where an approximate but infeasible command is given (e.g., a grasp slightly inside the cuboid wall), an admittance or impedance controller can soften the correction, letting the gripper slide to a nearby valid pose rather than resisting rigidly.

In practice, these methods ensure that control inputs respect the physical robot's capabilities and avoid reliance on the physics engine alone to reject invalid states. For a real hardware system, such reachability checks are essential to prevent commanding unreachable or unsafe poses.

# 3 Challenges

## 3.1 Contact Jitter

Direct mesh–mesh collisions between STL geometries often introduced high-frequency jitter. This arises from instability in solving the contact complementarity conditions

$$\lambda \ge 0, \quad v_n \ge 0, \quad \lambda v_n = 0,$$

where $\lambda$ is the normal contact force and $v_n$ is the normal velocity. In practice, when the local mesh curvature is high or contact normals fluctuate between adjacent triangles, the solver oscillates between penetration and separation states. To mitigate this, insertion motions were executed at very low approach velocities and with increased contact damping, thereby allowing the penalty formulation

$$F_n = kd + cv_n$$

to dissipate oscillations more effectively. In addition, grasp detection employed hysteresis (requiring forces to remain above threshold for multiple frames) to filter out transient contact spikes. During the insertion phase, controller gains were deliberately lowered and compliance (admittance control) was introduced to absorb noise rather than amplifying it.

### 3.2 Friction Realism

The choice of the friction coefficient $\mu$ is central to producing realistic object interaction. If $\mu$ is set too low, the peg readily slips from the gripper, leading to failed lifts. If it is set too high, contacts exhibit "sticking" behavior, causing solver chatter and preventing realistic micro-slips. The solution was to selectively tune lateral and spinning friction only on the gripper finger links, while leaving global defaults moderate. This local tuning better represents rubberized pads on rigid jaws:

$$F_t \leq \mu F_n,$$

with $\mu_{\text{pad}} > \mu_{\text{global}}$,

ensuring reliable grasps while maintaining natural sliding interactions for other bodies. This compromise allowed the peg to be lifted consistently without inducing nonphysical sticking in the environment.

### 3.3 Stability

Maintaining stability of the simulation during delicate contact phases required explicitly slowing down gripper motions. By reducing commanded velocities, the solver had sufficient iterations per contact to resolve penetration depths and forces without divergence. This is especially critical during insertion, where small lateral misalignments produce significant contact forces. A quasi-static regime was therefore enforced: gripper motions were damped, and insertion progressed incrementally, which allowed accurate force capture while avoiding solver blow-up. In practice, this meant the agent transitioned to slower gains near contact, effectively emulating a human operator's guarded move strategy.

## 4 Potential Improvements

### 4.1 Contact & Geometry

In the current setup, peg and cuboid are modeled directly from STL meshes. While this preserves visual fidelity, it makes the contact solver handle large numbers of tiny, irregular triangles, which amplifies jitter and false penetrations. A practical improvement is to preprocess these meshes into **convex decompositions** (e.g., using VHACD). Convex proxies reduce the number of contact points, ensure cleaner normals, and make the contact complementarity problem more stable, without sacrificing much realism. Another geometric refinement is to **add compliant finger pads** with softer contact stiffness and damping than the rigid gripper body. Real-world grippers typically have elastomeric pads, and modeling this compliance smooths force transitions, reduces chatter, and produces more human-like grasp forces. Together, convex decomposition and soft pads would turn noisy, discontinuous force spikes into smoother force ramps, improving both control stability and physical realism. Introducing **rolling friction** is another realistic addition, since the peg and cuboid edges experience both sliding and rolling interactions; without it, objects may spin unnaturally or fail to dissipate energy as expected. These refinements collectively help the engine approximate physical contacts more closely while maintaining numerical stability.

### 4.2 Solver & Dynamics

Although the simulation uses a stable step of $\Delta$t=1/240 s and tuned solver iterations, the insertion task still highlights trade-offs between speed and stability. Increasing `numSolverIterations` and tuning `erp/contactERP` can reduce interpenetration and help the solver converge more robustly. However, a finer step size (e.g.,

$\Delta$t=1/300 - 1/480 s) during high-contact phases like insertion would yield more accurate force trajectories and smoother dynamics, albeit with higher computational cost.

## 4.3 Control

The current admittance-based insertion strategy is aligned with the world zzz-axis, which works in controlled setups but ignores the fact that the **true insertion axis is defined by the cuboid's local frame**. Re-aligning the admittance law to this axis ensures that compliance responds to forces along the actual hole axis, not global coordinates, making insertion more robust under tilted placements. Additionally, adding a **spiral or "wiggle" micro-motion search** would replicate human-like strategies for overcoming small misalignments: instead of forcing a straight-line insertion, the peg traces a small oscillatory path that increases the chance of finding clearance. Finally, for tele-operation, direct keyboard inputs are mapped without filtering, which can produce jerky motions. A simple **low-pass filter** on commanded velocities or positions would smooth these inputs, improving controllability and reducing the risk of destabilizing the contact solver.

## 4.4 Performance

Performance bottlenecks arise primarily during logging and GUI rendering. By default, the PyBullet GUI overlays (contact points, debug visuals) impose additional overhead, which can distort benchmarking results. Disabling these overlays during performance runs frees computation for the physics solver itself. Similarly, when the agent operates in open-loop scripted mode, simulation does not need to synchronize with wall-clock time. In such cases, batching multiple `stepSimulation` calls per control tick allows faster-than-real-time execution while still maintaining phase-based logic. This makes large-scale data collection (e.g., for learning-based controllers) feasible without compromising the fidelity of logged trajectories.

# Conclusion

The implemented pipeline emphasizes **stability and safety** over raw speed, with friction tuning, force-based grasp checks, slow guarded motion, and admittance control. Future upgrades—convex collision proxies, soft pads, axis-aligned admittance, and solver tuning—will enhance realism and efficiency.