Learning to Stabilize: A Policy-Based Approach to Physics Simulation Parameter Control

Yinchu Liu
yinchu.liu.1@ens.etsmtl.ca
École de technologie supérieure
Montréal, Canada

Maud Marchal
maud.marchal@irisa.fr
Univ. Rennes, INSA, IRISA
Rennes, France

Paul G. Kry
kry@cs.mcgill.ca
McGill University
Montréal, Canada

Kenny Erleben
kenny@di.ku.dk
University of Copenhagen
DARERL ApS
Copenhagen, Denmark

Sheldon Andrews
sheldon.andrews@etsmtl.ca
École de technologie supérieure
Montréal, Canada

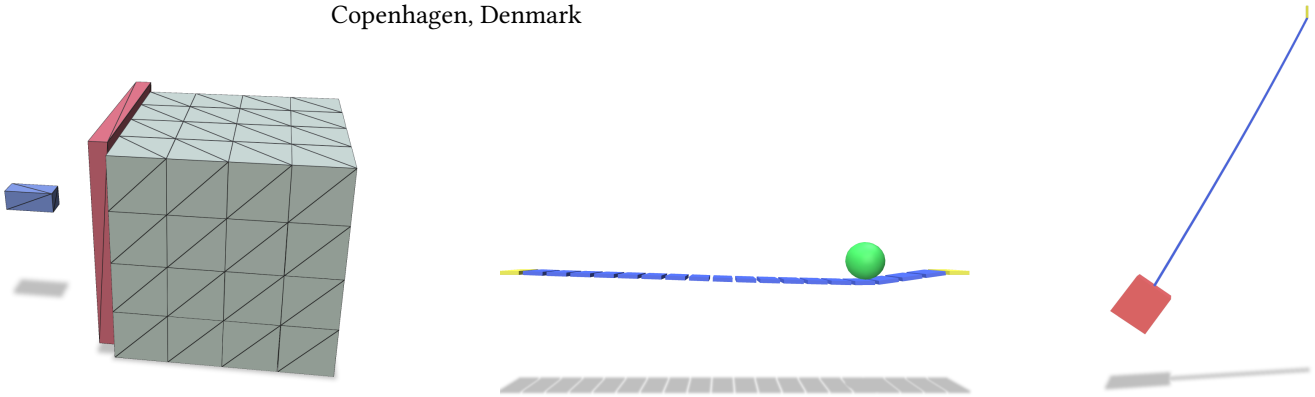


Figure 1: Challenging scenes stabilized by our parameter control policies. *Hydraulic Rig* (left): a rigid stack (sage green) is pushed by a hydraulic piston (terracotta red) against a fixed support bar (blue); *Rope Bridge* (middle): segmented deck (blue), fixed supports (yellow), impacting ball (green); *Swinging Boxes* (right): suspended payload box (red), connecting rod (blue), fixed anchor (yellow). Animations of these scenes can also be found in the supplementary video.

Abstract

Numerical stability in physics-based simulation relies on a careful balancing act between various hyperparameters. Sub-stepping and damping are primary levers for this control, yet they impose a problematic trade-off. Insufficient sub-stepping triggers divergence, while excessive sub-stepping incurs computational overhead. The interplay of damping and stiffness presents similar challenges. Under-damped systems risk numerical explosions, whereas over-compensation leads to artificial energy loss, robbing the simulation of its physical vitality and resulting in perceptually "soft" interactions. Existing approaches typically rely on fixed or heuristically tuned parameters that do not optimally balance this trade-off at each time step. We propose a reinforcement learning (RL) framework that learns adaptive hyperparameter policies, enabling an RL agent to dynamically select parameters such as the number of sub-steps and damping at each simulation time step. To improve generalization and scalability, we investigate both a global policy approach and a policy applied to local subdomains, which offers more fine-grained adaptivity. We validate the framework on several challenging stiff rigid-body simulation scenarios involving bilateral constraints and frictional contact. Our results show that the learned policies achieve stable and robust simulations with reduced computational cost and minimized artificial dissipation compared to manually tuned baselines.

CCS Concepts

• **Computing methodologies** → **Physical simulation**; *Reinforcement learning*.

Keywords

Sub-stepping, Damping, Rigid-body dynamics, Reinforcement learning, Physics-based animation

ACM Reference Format:
Yinchu Liu, Maud Marchal, Paul G. Kry, Kenny Erleben, and Sheldon Andrews. 2026. Learning to Stabilize: A Policy-Based Approach to Physics Simulation Parameter Control. In *The 19th ACM SIGGRAPH Conference on Motion, Interaction, and Games (MIG '26), December 11–13, 2026, Charleston, SC, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3828647.3857301

MIG '26, Charleston, SC, USA
© 2026 Copyright held by the owner/author(s).
This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *The 19th ACM SIGGRAPH Conference on Motion, Interaction, and Games (MIG '26), December 11–13, 2026, Charleston, SC, USA*, https://doi.org/10.1145/3828647.3857301.

1 Introduction

Real-time physics simulation is a fundamental component of modern graphics applications, ranging from virtual reality and video

games to robotic training environments. However, maintaining numerical stability in these dynamical systems is a delicate balancing act. Simulation hyperparameters, such as the time step, and intrinsic material properties, such as stiffness and mass ratios, dictate the numerical conditioning of the underlying linear and non-linear system solvers and the stability of the numerical integrator. When these components are pushed beyond their operational limits, they suffer from numerical divergence, manifesting visually as jittering, explosive energy injection, or other non-physical artifacts.

Traditionally, avoiding unstable configurations in the numerical landscape requires simulation engineers to engage in arduous, manual hyperparameter tuning. This often means sacrificing realism or performance for numerical robustness. Common engineering workarounds include aggressive sub-stepping at the cost of increased computational overhead, artificially capping stiffness parameters, or injecting non-physical damping forces that render materials unnaturally sluggish.

Our work stems from the key insight that a wide envelope of parameters can yield stable simulations during nominal states, and aggressive parameter tuning is only strictly required when the system (temporarily) encounters an ill-conditioned configuration. If an online oracle could dynamically adjust simulation parameters at every frame, fidelity and performance could be sacrificed only when and where it is necessary to prevent divergence. To realize this strategy, we reformulate the dynamic stabilization of physics solvers as a reinforcement learning (RL) problem wherein a learned policy preemptively stabilizes the simulation based on the current state.

We demonstrate the benefits of our approach using various real-time environments of ill-conditioned rigid-body systems and show that our learned policies exhibit generalization capabilities, successfully stabilizing simulations of varying spatial dimensions and degrees of freedom without requiring retraining.

2 Related Work

Maintaining simulation stability at large time steps has historically required model-specific heuristics or manual parameter tuning. We replace these heuristics with a model-agnostic RL framework that dynamically controls numerical solver hyperparameters. In this section, we survey prior literature in simulation stabilization, adaptive time-stepping, and learning-guided solvers.

2.1 Stable Physics-based Animation

Post-step stabilization [Cline and Pai 2003] and Baumgarte stabilization [Andrews et al. 2022] are fundamental techniques to address constraint error accumulation in rigid body simulations. Other work has explored careful reordering of constraints and bodies to improve iterative solver convergence [Erleben 2007; Xiong et al. 2025]. Mass-splitting has proven to be an effective strategy for eliminating jitter in parallel iterative solvers [Tonge et al. 2012]. Primal formulations of dynamics have also been shown to be robust to numerical ill-conditioning caused by large mass ratios [Macklin et al. 2020]. Alternative strategies control the energy of a simulation through projection [Dinev et al. 2018].

More relevant to our work are adaptive time stepping schemes. Macklin et al. [2019] propose splitting a large time step into multiple sub-steps, each with a reduced number of solver iterations, and show that sub-stepping can be more effective than performing multiple iterations within a single large time step. Related are asynchronous time integration schemes, which have been shown to offer exceptional guarantees for producing interpenetration-free simulations [Harmon et al. 2009; Li et al. 2020].

Tournier et al. [2015] demonstrate that the geometric stiffness, even in its explicit form, has the ability to stabilize constrained dynamics simulations. Follow-up work by Andrews et al. [2017] finds that combining a diagonalized version of the geometric stiffness with a numerical integrator analysis can be used to adaptively apply damping elements to a simulation, resulting in a stable simulation even for very high mass ratios. More recently, Giles and Andrews [2024] extend the concept to adaptive sub-stepping, adjusting the timestep of the simulation according to stability requirements. These previous works in graphics exploit heuristics or rules derived from the simulation model or problem itself to control adaptive stabilization. It is model-specific. Our RL approach offers a more black-box view of the numerical method and the simulation model.

2.2 Learned Parameter Tuning and Stabilization

Another class of techniques leverages machine learning to improve solver robustness and simulation stability.

Notably, RL is used for model calibration or parameter estimation. Ritto et al. [2022] combine RL with approximate Bayesian computation for parameter inference, whereas Tian et al. [2022] propose a novel constrained Lyapunov-based actor–critic algorithm that applies deep reinforcement learning to train an agent that adjusts physics-based model parameters to match model response with observations. Zhang [2025] explore DRL parameter tuning for turbulence models.

Alternatives to RL approaches have instead exploited end-to-end differentiable pipelines [Arnavaz et al. 2023; Jatavallabhula et al. 2021] to tune physical simulations directly from vision input. The key difference to our work is that they tune physical parameters based on real-world observation, whereas we focus on tuning numerical solver hyperparameters to combat numerical instabilities.

However, other work has used deep learning to improve solver performance and robustness. For instance, Um et al. [2020] combine neural networks directly with physics solvers to reduce step-wise approximation error. Ichnowski et al. [2021] introduce an RL framework that accelerates quadratic programming by using a reinforcement learning policy to dynamically adjust internal solver step-size parameters between iterations. This learned in-loop control strategy achieves up to a 3× speedup over traditional mathematical heuristics. More closely related to our goals of achieving stable physics simulation, Riley et al. [2025] similarly introduce an RL framework for adaptive time-integration specifically tailored to handle non-smooth dynamics and sharp contact discontinuities. Their policy dynamically modulates the timestep based on the system state, avoiding numerical divergence. Their control strategy effectively balances computational efficiency with robust constraint enforcement in highly non-linear simulation regimes.

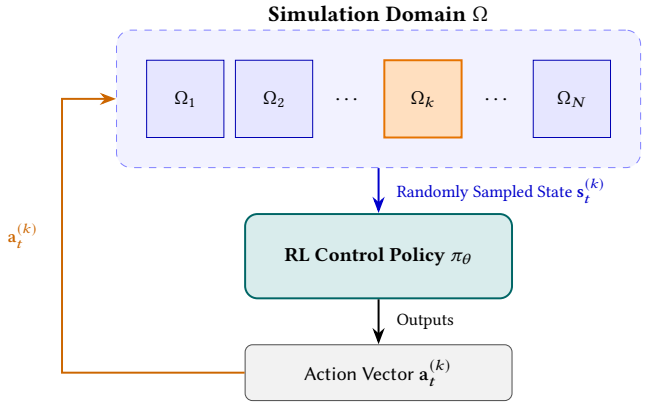


Figure 2: Overview of the RL control loop on domain Ω. At each step during training, a subdomain state $\mathbf{s}_t^{(k)}$ is randomly selected from domain subdomains ($k \in \{1, \ldots, N\}$) and passed to policy π_θ, which emits an action vector $\mathbf{a}_t^{(k)}$ representing solver hyperparameter adjustments for subdomain Ω_k. Depending on the evaluated strategy, $\mathbf{a}_t^{(k)}$ controls either Substepping or damping. Under a global control strategy ($N = 1$), there is no spatial partitioning, and the domain Ω is treated as a single unpartitioned system.

Reinforcement learning for direct simulation parameter control has also been explored for computational fluid dynamics and astrophysics. Pawar and Maulik [2021] use RL to dynamically adjust solver parameters (e.g., relaxation factors in CFD) during runtime and show that this improves convergence speed and efficiency. Saz Ulibarrena and Portegies Zwart [2025] show how to use RL for adaptive time-stepping. In their work, an RL policy controls time step size for N-body simulations and addresses limitations of fixed or heuristic step rules.

A key similarity of these approaches is the use of RL for controlling numerical parameters online. However, we target interactive computer graphics simulations, specifically rigid-body simulation with constraints.

3 Methodology

To adapt solver parameters to the evolving conditions of a physics simulation, we formalize the dynamic stabilization of numerical solvers as a Markov Decision Process (MDP). At each time step t, the simulation state is represented as a vector $\mathbf{s}_t \in \mathcal{S} \subseteq \mathbb{R}^{d_s}$, combining physical state variables with numerical stability metrics. Based on this state, a learned policy $\pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)$ outputs an action vector $\mathbf{a}_t \in \mathcal{A} \subseteq \mathbb{R}^{d_a}$ specifying adjustments to target solver hyperparameters. The simulation environment then advances the system state via forward dynamics, $\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t)$.

Figure 2 illustrates the overall closed-loop control architecture. The controller supports two distinct stabilization strategies: (i) **Substepping control**, which dynamically regulates temporal integration resolution, and (ii) **Damping control**, which modulates local energy dissipation. Both strategies can be applied either globally or locally. To enable localized stabilization, we partition the simulation domain into N subdomains, where $N = 1$ corresponds to global control (no partitioning), and $N > 1$ allows independent control of individual subdomains. In partitioned domains ($N > 1$), local actions target active subdomains directly, whereas in unpartitioned domains ($N = 1$), actions operate globally.

The policy is trained using a scalar reward function $r(\mathbf{s}_t, \mathbf{a}_t)$ that balances numerical stability against computational overhead. In the remainder of this section, we detail the underlying rigid-body formulation, dynamic spatial domain partitioning, state-action formulations, and reward definitions.

3.1 Stabilizing Constrained Rigid-body Dynamics

Our rigid body simulations follow a velocity-level, impulse-based formulation using compliant constrained dynamics. For a system of n rigid bodies subject to m-dimensional constraints, the post-step velocities and constraint impulses are computed at each time step with duration h using the first-order time-stepping scheme:

$$\begin{bmatrix} \mathbf{M} - h\mathbf{B} & -\mathbf{J}^T \\ \mathbf{J} & \frac{1}{h^2}\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{u}^+ \\ \boldsymbol{\lambda}^+ \end{bmatrix} = \begin{bmatrix} \mathbf{M}\mathbf{u} + h\mathbf{f} \\ \frac{-\boldsymbol{\phi}}{h} \end{bmatrix}, \tag{1}$$

where $\mathbf{M} \in \mathbb{R}^{6n \times 6n}$ is the block-diagonal mass matrix containing translational masses and rotational inertia, $\mathbf{B} \in \mathbb{R}^{6n \times 6n}$ is the velocity damping matrix, $\mathbf{J} \in \mathbb{R}^{m \times 6n}$ is the constraint Jacobian, and $\mathbf{C} \in \mathbb{R}^{m \times m}$ is the constraint compliance matrix. The vectors $\mathbf{u} \in \mathbb{R}^{6n}$ and $\mathbf{f} \in \mathbb{R}^{6n}$ denote the pre-step generalized velocities and external forces, respectively. The superscript '+' indicates quantities evaluated at the end of the time step. The unknown generalized constraint impulses integrated over the step are denoted by $\boldsymbol{\lambda}^+ \in \mathbb{R}^m$, and $\boldsymbol{\phi} \in \mathbb{R}^m$ are the position-level constraint errors.

The inclusion of a diagonal damping matrix $\mathbf{B} \in \mathbb{R}^{6n \times 6n}$ in Eq. (1) permits ambient linear and angular damping to be applied to the inertial DOFs. This helps stabilize a simulation by limiting constraint violations that can occur due to velocity updates, but can also improve numerical conditioning of the linear system.

3.1.1 Partitioned simulation. A multibody simulation can be decomposed into N smaller subdomains $\Omega_1, \Omega_2, \ldots, \Omega_N$ using a partitioning strategy indexed over the rigid bodies in the system. Since the mass and damping matrices are inherently block diagonal, the system in Eq. (1) decomposes as

$$\mathbf{M} - h\mathbf{B} = \begin{bmatrix} \mathbf{M}^{(1)} - h\mathbf{B}^{(1)} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{M}^{(N)} - h\mathbf{B}^{(N)} \end{bmatrix}, \tag{2}$$

where each block $\mathbf{M}^{(k)}$ and $\mathbf{B}^{(k)}$ corresponds strictly to the DOFs in subdomain Ω_i. We note that with $N = 1$, Eq. (2) reverts to the upper-left block of the global system in Eq. (1).

The velocity, generalized momentum, and external forces can be decomposed block-wise as

$$\mathbf{u} = \begin{bmatrix} \mathbf{u}^{(1)} \\ \vdots \\ \mathbf{u}^{(N)} \end{bmatrix}, \quad \mathbf{M}\mathbf{u} = \begin{bmatrix} \mathbf{M}^{(1)}\mathbf{u}^{(1)} \\ \vdots \\ \mathbf{M}^{(N)}\mathbf{u}^{(N)} \end{bmatrix}, \quad \text{and} \quad \mathbf{f} = \begin{bmatrix} \mathbf{f}^{(1)} \\ \vdots \\ \mathbf{f}^{(N)} \end{bmatrix}.$$

Constraint impulses are similarly partitioned across subdomains via the transpose of the Jacobian:

$$\mathbf{J}^T\boldsymbol{\lambda} = \begin{bmatrix} (\mathbf{J}^T\boldsymbol{\lambda})^{(1)} \\ \vdots \\ (\mathbf{J}^T\boldsymbol{\lambda})^{(N)} \end{bmatrix}. \tag{3}$$

3.1.2 Damping strategy. Given the angular damping coefficient used to stabilize a specific subdomain, $\boldsymbol{\beta}^{(k)}$, damping is applied directly to the rotational inertial DOFs, such that

$$\mathbf{B}^{(k)} = \begin{bmatrix} 0 & 0 & \dots \\ 0 & \boldsymbol{\beta}^{(k)}\mathbf{I}_{3\times 3} & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}. \tag{4}$$

Restricting the diagonal damping matrix $\mathbf{B}^{(k)}$ strictly to the rotational sub-blocks is motivated by the geometric stiffness analysis of Andrews et al. [2017], which notes that instabilities are predominantly due to rotational DOFs and that applying damping to translational DOFs induces non-physical spatial drag.

3.2 State Space

To handle the varying number of bodies and constraints during simulation, we construct a fixed-dimensional state representation based on aggregated physical features. Inspired by Ichnowski et al. [Ichnowski et al. 2021], the observations encode the current stability state of the system while remaining invariant to changes in the underlying DOFs.

We must also consider that constrained rigid-body simulations involve physical quantities, such as forces, impulses, velocities, and constraint errors, that naturally span many orders of magnitude. The absolute value of a physical quantity is far less informative to the RL agent than its order of magnitude, especially when using standard neural networks that operate on absolute additive differences and may overemphasize small fluctuations in large simulation quantities. An element-wise signed logarithmic transformation is thus used to map the dynamic range of physical quantities into a linear scale, allowing the network to perceive proportional physical changes uniformly across all magnitudes. For a scalar input ξ, the transformation is defined as

$$\mathcal{T}(\xi) \triangleq \operatorname{sign}(\xi)\ \operatorname{clip}\left(\log_{10}\left(1 + \frac{|\xi|}{\epsilon}\right), -L, L\right), \tag{5}$$

where $\epsilon = 10^{-10}$ and $L = 10$ are used in our experiments.

To capture extreme directional generalized forces across an arbitrary number of constraints without relying on a variable-length Jacobian, we map constraint impulses into a fixed-dimensional spatial representation. Specifically, for each active constraint c in the set of current active constraints C, let $\mathbf{J}_c$ denote its constraint Jacobian and $\boldsymbol{\lambda}_c$ denote the corresponding constraint impulses. We compute a directional impulse vector $\mathbf{f}_{\text{imp}} \in \mathbb{R}^6$ as the element-wise maximum magnitude over all constraints and each coordinate axis:

$$\mathbf{f}_{\text{imp}} = \max_{c\in C} \left|\mathbf{J}_c^\top \boldsymbol{\lambda}_c\right| \in \mathbb{R}^6. \tag{6}$$

This max-pooling operation maps an arbitrary and time-varying constraint impulse vector into a fixed 6-dimensional vector that preserves the peak force and torque applied along each canonical axis x, y, z, while also maintaining input dimension invariance.

Let $\phi_{\max}$ denote the maximum constraint error, including contact penetration, over all the active constraints, and $m_{\min}$ and $m_{\max}$ be the minimum and maximum masses, respectively, over all bodies $\mathcal{B}$. The resulting aggregated state representation provided to the policy is expressed as

$$\mathbf{s}_t = \mathcal{T}\left(\begin{bmatrix} m_{\max} \\ m_{\min} \\ \phi_{\max} \\ \mathbf{f}_{\text{imp}} \end{bmatrix}\right) \in \mathbb{R}^9. \tag{7}$$

This minimal feature vector encapsulates the critical physical quantities required for adaptive control.

3.3 Action Space

The action space $\mathcal{A}$ defines the parameter modifications required to maintain numerical stability and constraint satisfaction at each time step. While uniform global adjustments to damping [Andrews et al. 2017] or time step size [Giles and Andrews 2024] can stabilize a system, they often introduce artificial stiffness or excessive computational overhead to regions that do not require intervention. To enable targeted stabilization while accommodating simulations of varying sizes, we adopt a spatial partitioning strategy that applies a shared policy independently to N local subdomains $\Omega_1, \dots, \Omega_N$, as illustrated in Figure 2.

At each step t, the policy evaluates the state representation $\mathbf{s}_t^{(k)}$ of an active subdomain Ω_k and outputs a corresponding control action vector $\mathbf{a}_t^{(k)}$. We consider two stabilization strategies: adaptive sub-stepping and localized damping control, applied to subdomains dynamically constructed using graph parititioning.

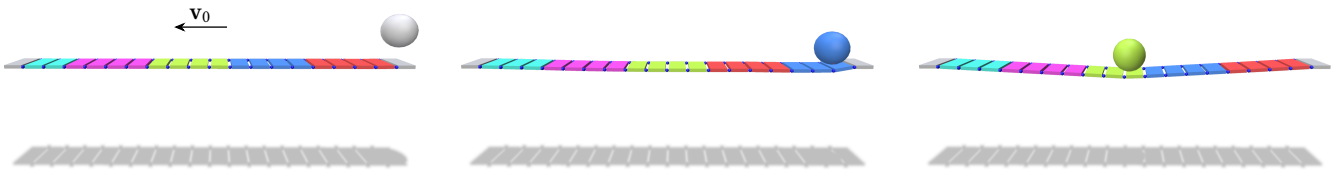
3.3.1 Adaptive Sub-stepping. Adaptive sub-stepping improves numerical stability by increasing temporal resolution when necessary. Rather than introducing asynchronous time integration across subdomains, which requires additional synchronization and coordination at subdomain boundaries, we maintain a uniform time step across the simulation.

The policy predicts a sub-step count $\bar{q}^{(k)}$ for each active subdomain Ω_k. These local requirements are aggregated to determine a global sub-step count:

$$\bar{q} = \max_{k\in\{1,\dots,N\}} \bar{q}^{(k)}. \tag{8}$$

The frame time-step duration h is then uniformly divided into $\bar{q}$ sub-steps, each of duration $\Delta t = h/\bar{q}$. This strategy allows local policy decisions to guide the temporal resolution while preserving synchronization across the entire simulation domain.

3.3.2 Damping Control. Localized damping control stabilizes the simulation by adjusting energy dissipation in individual subdomains. Unlike sub-stepping, which requires a globally synchronized time step, damping can be controlled independently within each region. For each active subdomain Ω_k, the policy predicts a three-dimensional damping vector $\boldsymbol{\beta}^{(k)} \in \mathbb{R}^3$ that controls the local damping coefficients. This enables spatially varying damping, allowing the policy to suppress localized numerical oscillations while avoiding unnecessary damping in regions that do not require intervention.



(a) Airborne ball (b) Right-span contact interaction (c) Mid-span contact interaction

Figure 3: Dynamic subdomain graph construction and updating via graph partitioning. Color-coded regions denote independent subdomain partitions. (a) The initial velocity of the ball $\mathbf{v}_0$ points to the left. When the ball is airborne and shares no active contact with the bridge deck, it exists as an unconstrained node and is excluded from the active constraint graph (neutral grey). (b–c) Upon making contact with a deck segment, a dynamic contact edge is added, and the partitioning algorithm incorporates the ball into the corresponding local subdomain partition (matching color).

In the case of a global control strategy ($N = 1$), the entire simulation domain Ω is treated as a single region, and the policy directly determines the global sub-step count $\bar{q}$ or damping vector $\boldsymbol{\beta}$ without local aggregation.

3.3.3 Subdomain Partitioning. To evaluate local policy actions, the simulation is dynamically partitioned into non-overlapping subdomains at every simulation frame. We follow the approach outlined by Liu and Andrews [2022], and specifically use a breadth-first search (BFS) algorithm. A constraint graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined, where nodes $\mathcal{V}$ represent rigid bodies (deck segments, anchors, and the ball) and edges $\mathcal{E}$ represent active bilateral joints or contact constraints. Traversal originates from the fixed scene bodies (shown in grey in Figure 3) and expands outward to cluster connected bodies into colored subdomains based on topological distance.

Because physical interactions evolve dynamically over time, the constraint graph $\mathcal{G}$ is updated at each frame step. For instance, in Figure 3(a), when the ball is in free flight and has no active contact constraints with the bridge, it acts as an unconstrained "orphan node". In this state, it is excluded from the active constraint graph entirely and rendered in neutral grey. Once contact occurs, a temporary constraint edge is added to $\mathcal{E}$, allowing the BFS traversal to incorporate the ball into the local constraint graph. Consequently, the ball is assigned to the specific subdomain partition of the deck segment it impacts (Figure 3(b) and (c)), ensuring that subdomain-level policy decisions and sub-stepping actions are dynamically targeted to active contact zones.

3.4 Reward Function

Large constraint errors can lead to numerical instability and simulation blow-ups. Geometric stiffness damping and adaptive sub-stepping mitigate these errors to improve stability, but may increase computational overhead; excessive damping can also introduce visually undesirable, overly viscous motion. The reward function is therefore designed to balance numerical stability against computational cost and motion quality. We define the excess error as $\delta_\phi = \max(\phi_{\max} - \varepsilon, 0)$, where ε is a prescribed tolerance. Since both our stabilization strategies share the same goal, we define a unified reward function for both sub-stepping and damping control:

$$R(\mathbf{s}_t, \mathbf{a}_t^{(k)}) = \mathrm{clip}\left(-w_C \tanh\left(\frac{\delta_\phi}{\sigma_c}\right) - \eta \|\mathbf{a}_t^{(k)}\|_1,\ -1,\ 0\right), \qquad (9)$$

where w_C controls the constraint-error penalty, σ_c controls its transition rate, and η penalizes the applied control. Values used in our experiments are reported in Table 2.

3.4.1 Sub-stepping Control. For sub-stepping-based stabilization, the scalar action $a_t^{(k)} \in [0, 1]$ corresponds to the normalized sub-step count $\bar{q}^{(k)}$.

3.4.2 Damping Control. For policies that use localized damping to stabilize, the action $\mathbf{a}_t^{(k)} \in [0, 1]^3$ corresponds to the normalized damping vector $\bar{\boldsymbol{\beta}}^{(k)}$. We use the L_1 norm for the regularization term to induce sparsity in the action space.

3.5 Network Architecture and Training

The policy π_θ and critics Q_ϕ are represented by Multi-Layer Perceptrons (MLPs) with two hidden layers of 256 units and ReLU activations. The primary agent is trained using the Truncated Quantile Critics (TQC) algorithm [Kuznetsov et al. 2020] through Stable-Baselines3 Contrib [Raffin et al. 2021]. As an offline and sample-efficient algorithm, TQC is particularly well-suited for our environment. Similar to [Riley et al. 2025], we find that TQC prevents the policy from collapsing to the lower bound of the action space during nominal simulation states, since the regularization term in the reward function encourages minimal intervention (e.g., a single substep or zero damping). Standard online RL algorithms, such as PPO, often degenerate into outputting minimum actions to avoid the regularization penalties. TQC combats this boundary-saturation failure mode through a distributional value model, which evaluates the high-penalty risk of unmitigated instabilities to keep the policy active and responsive across both nominal and unstable dynamics.

The critic ensemble consists of three networks, each estimating 16 quantiles. The four highest quantiles are discarded during target construction to reduce overestimation bias. Training is performed for 2.0×10^6 timesteps with a batch size of 256, a learning rate of 1×10^{-4}, and a replay buffer containing 5×10^5 transitions. The complete set of training and simulation parameters is reported in Table 1.

3.6 Adaptive Control Algorithm

The execution pipeline for our RL-based stabilization is provided in Algorithms 1 and 2. We provide further algorithmic details below.

Table 1: TQC hyperparameters and simulation settings.

Hyperparameter	Value
Algorithm	TQC
Network architecture	(256, 256)
Learning rate	1×10^{-4}
Batch size	256
Buffer size	5×10^{5}
Maximum episode steps	500
Total training timesteps	2.0×10^{6}
Damping bounds	[0, 10]
Sub-step bounds	[5, 100]
Constraint target	10^{-4}

Table 2: Reward-function weights used for training. The parameters w_C, η and σ_c correspond to the constraint-error and control-cost terms in Eq. (9). The same weights are used for global and subdomain RL within each scenario.

Scenario	Control	w_C	η	σ_c
Rope Bridge	Sub-stepping	0.35	0.30	0.005
	Damping	0.50	0.05	0.005
Swinging Boxes	Sub-stepping	0.35	0.30	0.005
	Damping	0.50	0.10	0.005
Hydraulic Rig	Sub-stepping	0.35	0.50	0.05
	Damping	0.50	0.10	0.05

3.6.1 Adaptive Sub-stepping. The simulation is first partitioned into N non-overlapping subdomains $\{\Omega_k\}_{k=1}^{N}$. For each subdomain, a 9-dimensional state vector $\mathbf{s}_t^{(k)} \in \mathbb{R}^9$ is constructed from local mass bounds $(m_{\max}^{(k)}, m_{\min}^{(k)})$, maximum constraint error $\phi_{\max}^{(k)}$, and a 6D directional impulse vector $\mathbf{f}_{\text{imp}}^{(k)} \in \mathbb{R}^6$ representing the axis-wise maximum of generalized constraint impulse magnitude across all local constraints. A per-subdomain sub-step count is then computed $\bar{q}^{(k)} \in [\bar{q}_{\min}, \bar{q}_{\max}]$ via policy evaluation, before computing a global maximum $\bar{q}$ over all subdomains.

Once $\bar{q}$ is determined, the effective timestep becomes $\Delta t = h/\bar{q}$ and the simulator executes the determined number of sub-steps, with the function Step performing collision detection, constraint resolution, and numerical integration of positions and velocities.

3.6.2 Damping Control. We alternatively employ a learned adaptive rotational damping controller as outlined in Algorithm 2. Instead of applying an isotropic coefficient, the policy outputs a three-dimensional action vector $\mathbf{a} \in [0, 1]^3$, enabling anisotropic control over rotational dissipation gains along individual canonical axes.

The simulation is again partitioned into N subdomains $\{\Omega_k\}_{k=1}^{N}$, and the per-domain state vector $\mathbf{s}^{(k)}$ is computed the same as for the adaptive sub-stepping policy. However, here the policy π produces a localized 3D action $\mathbf{a}_t^{(k)} \in [0, 1]^3$, which is mapped to a physical rotational damping $\boldsymbol{\beta}^{(k)} \in [\boldsymbol{\beta}_{\min}, \boldsymbol{\beta}_{\max}]^3$. Local damping values are assembled into a matrix form $\mathbf{B}^{(k)}$ as outlined in Section 3.1.2, and then into a global matrix damping $\mathbf{B}$ using a block diagonal assembly process. The simulation is then stepped using timestep h and modified mass matrix $\mathbf{M} - h\mathbf{B}$.

Algorithm 1: Learned adaptive sub-stepping.

Data: Bodies $\mathcal{B}$, positions $\mathbf{x}$, velocities $\mathbf{u}$ and constraint impulses $\boldsymbol{\lambda}_c$ from previous frame, time step h, action bounds $[\bar{q}_{\min}, \bar{q}_{\max}]$, policy π, number partitions N

$\{\Omega_k\}_{k=1}^{N} \leftarrow$ CreateSubDomains$(\mathcal{B}, N)$ // Section 3.3.3

foreach Ω_k **do**

  $\mathbf{f}_{\text{imp}}^{(k)} \leftarrow \max_{c \in C_k} |\mathbf{J}_c^{\top} \boldsymbol{\lambda}_c| \in \mathbb{R}^6$ // Equation (6)

  $\mathbf{s}_t^{(k)} \leftarrow \mathcal{T}\left(\left[m_{\max}^{(k)}, m_{\min}^{(k)}, \phi_{\max}^{(k)}, \mathbf{f}_{\text{imp}}^{(k)\top}\right]^{\top}\right)$ // Equation (7)

  $a^{(k)} \leftarrow \pi(\mathbf{s}_t^{(k)}) \in [0, 1]$ // num. sub-steps

  // map action to $[\bar{q}_{\min}, \bar{q}_{\max}]$

  $\bar{q}^{(k)} \leftarrow \left\lceil \bar{q}_{\min} + a^{(k)}(\bar{q}_{\max} - \bar{q}_{\min}) \right\rceil$

// Conservative local-to-global temporal synchronization

$\bar{q} \leftarrow \max_{k \in \{1,\ldots,N\}} \bar{q}^{(k)}$

// Numerical integration over $\bar{q}$ sub-steps

$\Delta t \leftarrow \frac{h}{\bar{q}}, i \leftarrow 0$

while $i < \bar{q}$ **do**

  $\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda} \leftarrow$ Step$(\mathbf{x}, \mathbf{u}, \Delta t)$

  $i \leftarrow i + 1$

Algorithm 2: Learned adaptive damping.

Data: Bodies $\mathcal{B}$, positions $\mathbf{x}$, velocities $\mathbf{u}$ and constraint impulses $\boldsymbol{\lambda}_c$ from previous frame, time step h, action bounds $[\beta_{\min}, \beta_{\max}]$, policy π, number partitions N

$\{\Omega_k\}_{k=1}^{N} \leftarrow$ CreateSubDomains$(\mathcal{B}, N)$ // Section 3.3.3

foreach Ω_k **do**

  $\mathbf{f}_{\text{imp}}^{(k)} \leftarrow \max_{c \in C_k} |\mathbf{J}_c^{\top} \boldsymbol{\lambda}_c| \in \mathbb{R}^6$ // Equation (6)

  $\mathbf{s}_t^{(k)} \leftarrow \mathcal{T}\left(\left[m_{\max}^{(k)}, m_{\min}^{(k)}, \phi_{\max}^{(k)}, \mathbf{f}_{\text{imp}}^{(k)\top}\right]^{\top}\right)$ // Equation (7)

  $\mathbf{a}_t^{(k)} \leftarrow \pi(\mathbf{s}_t^{(k)}) \in [0, 1]^3$ // per-axis rot. damping

  // map action to $[\beta_{\min}, \beta_{\max}]$

  $\boldsymbol{\beta}^{(k)} \leftarrow \boldsymbol{\beta}_{\min} + \mathbf{a}_t^{(k)}(\boldsymbol{\beta}_{\max} - \boldsymbol{\beta}_{\min})$

  // assemble local rotational damping matrix $\mathbf{B}^{(k)}$ using $\boldsymbol{\beta}$

  $\mathbf{B}^{(k)} \leftarrow$ DampingMatrix$(\boldsymbol{\beta})^{(k)}$

// Step physics with rotational damping $\mathbf{B}^{(1)}, \ldots, \mathbf{B}^{(N)}$

$\mathbf{B} \leftarrow$ BlockDiagonal$(\mathbf{B}^{(1)}, \ldots, \mathbf{B}^{(N)})$

assemble modified mass matrix $\mathbf{M} - h\mathbf{B}$ // Equation (2)

$\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}_c \leftarrow$ Step$(\mathbf{x}, \mathbf{u}, \Delta t)$

4 Results

We evaluate the proposed learned adaptive sub-stepping and damping control frameworks across the three benchmark environments shown in the teaser (see Figure 1), which exhibit stiff, non-smooth contact dynamics and ill-conditioned multibody systems. All experiments are conducted on an Intel Core i9-13900K CPU with an NVIDIA RTX 4090 GPU and a simulation time step $h = 1/100$ s.

Table 3: Randomization ranges for each simulation scenario.

Scenario	Parameter	Range
Hydraulic Rig	mass	$[5, 100]$ kg
	piston linear velocity	$[0, 0.01]$ m/s
	source pressure	$[1 \times 10^6, 1 \times 10^7]$ Pa
Rope Bridge	ball mass	$[500, 5000]$ kg
	ball angular velocity	$[-10, 0]$ rad/s
	ball linear velocity	$[0.5, 10]$ m/s
	deck mass	$[1, 5]$ kg
Swinging Boxes	v_x	$[-10, 10]$ m/s
	v_y	$[-1, 1]$ m/s
	v_z	$[-10, 10]$ m/s
	angular velocity	$[-2, 2]$ rad/s

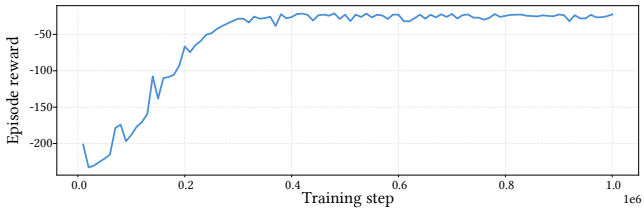


Figure 4: Policy Learning Performance on the *Hydraulic Rig* Scenario. Mean evaluation return over 10^6 training steps under global policy control, converging to a stable plateau.

To improve generalization across varying system dynamics, the physical parameters and initial conditions are randomized at the beginning of each training episode. The parameter ranges are scenario-dependent and are summarized in Table 3.

4.1 Policy Convergence

We first evaluate the learning efficiency and optimization stability of the policy on the complex, highly constrained *Hydraulic Rig* scenario. As illustrated in Figure 4, the agent initially explores the action space before the episodic reward steadily increases and eventually converges after approximately 250k training steps.

4.2 Structural Cohesion of Hydraulic Rig

To evaluate physical stability and geometric fidelity, Figure 5 illustrates the structural behavior of the *Hydraulic Rig* block assembly at frame 200 under different damping schemes. Standard Projected Gauss-Seidel (PGS) solver damping (Figure 5(a)) fails to mitigate non-smooth rotational chatter at contact manifolds, causing accumulative inter-block shearing and geometric misalignments across adjacent block interfaces. PGS is adopted for its computational efficiency, as direct solvers are generally too expensive for real-time simulation of densely stacked, contact-rich scenarios and may also exhibit numerical instabilities that lead to explosive behavior.

Conversely, both the Global RL policy (Figure 5(b)) and Subdomain RL policy (Figure 5(c)) dynamically adapt anisotropic rotational damping to neutralize high-frequency contact energy spikes. Consequently, both learned strategies preserve a tighter stack alignment and structural lattice cohesion throughout the simulation. Animations for this example can also be found in the supplementary video. Notably, the Subdomain RL policy achieves identical spatial coherence while operating independently per partition, demonstrating that localized anisotropic damping is sufficient to prevent interface shearing without over-damping non-interacting DOFs.

4.3 Adaptive Sub-Stepping Control

We evaluate learned sub-stepping against a conventional geometric adaptive heuristic across 250 frames of the *Rope Bridge* benchmark. Experiments are conducted under fixed initial conditions, with a ball mass of $m = 5000$ kg, an initial velocity of $v_x = 8$ m/s, and an angular velocity of $\omega_z = 5$ rad/s.

As illustrated in Figure 6 for the *Rope Bridge*, all three strategies exhibit sharp, localized increases in sub-step resolution in response to transient contact events across the 250 frame simulation. As shown in Figure 6(a), the Heuristic Sub-stepping method exhibits pronounced sub-stepping peaks during contact events, whereas Global RL and Subdomain RL handle these transients without requiring similarly high peaks. The steady increase in kinetic energy toward the end of the trajectory in Figure 6(b) results from the ball clearing the bridge and entering unconstrained gravitational freefall, during which gravitational potential energy is converted into kinetic energy. The near-identical kinetic energy profiles and joint error magnitudes across all three methods in Figures 6(b) and 6(c), respectively, demonstrate that both RL policies preserve physical fidelity comparable to the heuristic baseline while avoiding its large sub-stepping peaks.

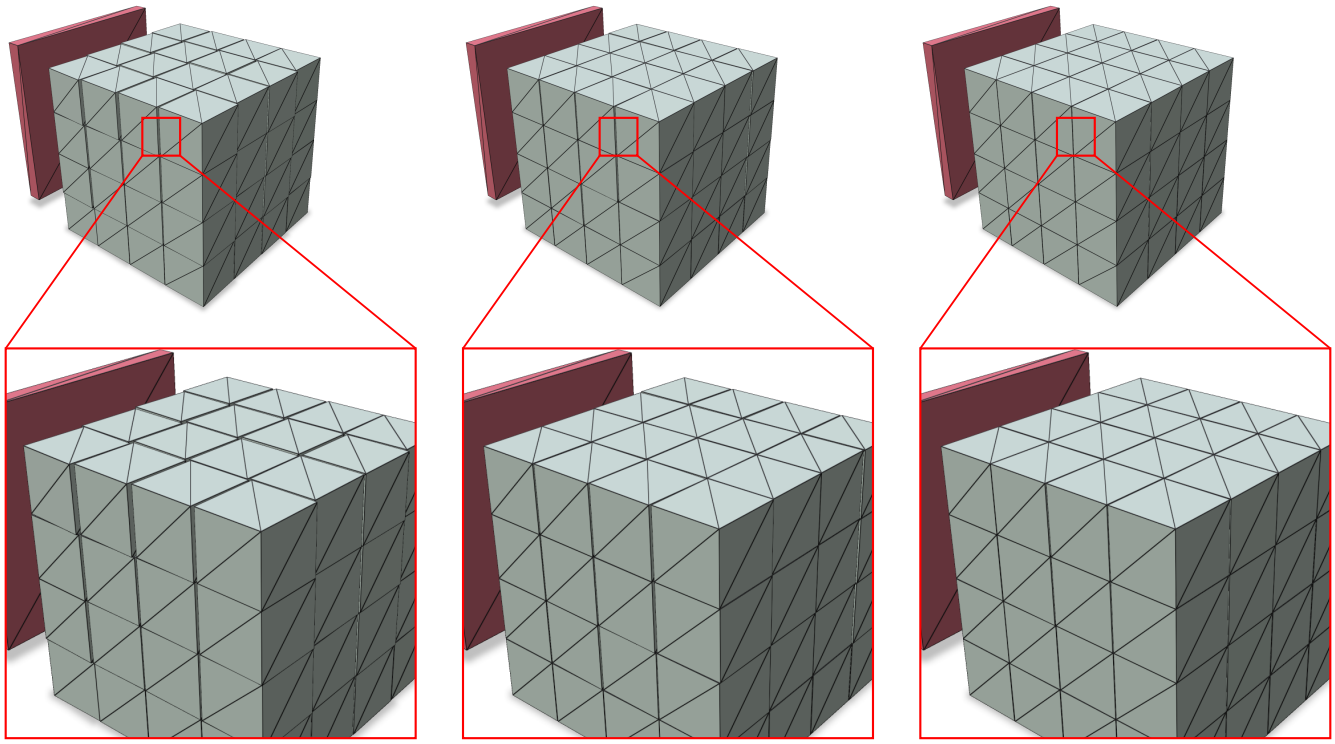
4.3.1 Robustness to Low-Iteration Constraint Solver. To evaluate policy resilience under reduced computation, we benchmark an unloaded Rope Bridge using PGS solver restricted to 25 iterations per simulation frame (Figure 7). Under this moderate iteration budget, the Heuristic Sub-stepping method under-allocates substeps, leading to spurious kinetic energy buildup, large constraint errors, and eventual divergence. In contrast, both Global RL and Subdomain RL maintain numerical stability and tighter joint error bounds, with both methods achieving near-zero kinetic energy.

4.3.2 High-Mass Impact Failure Modes. Figure 8 illustrates an extreme ballistic impact scenario ($m = 1000$ kg, $v_0 = 1.0$ m/s). When the sphere impacts the deck, the Heuristic Sub-stepping method fails to capture contact stiffness, allocating an insufficient sub-step count that results in severe penetration and explosive numerical instability (Figure 8(b)). Our proposed RL method anticipates high-mass contact dynamics from cached generalized impulse trends, increasing temporal sub-stepping to guarantee a stable trajectory throughout (Figure 8(a)).

4.4 Constraint Error, Sub-steps, and Damping

Across all evaluated scenarios, either the Global RL or Subdomain RL policy consistently outperforms the heuristic baselines, achieving a balance between accuracy and computational efficiency. Animations can be found in the supplementary video.

In adaptive sub-stepping (Table 4), Global RL significantly mitigates temporal overhead by substantially reducing the average



(a) Baseline Diag. Geom. Damping (b) Global RL Damping (c) Subdomain RL Damping

Figure 5: Qualitative assessment of structural alignment on the *Hydraulic Rig* scenario at frame 200. The magnified insets highlight the vertical joint interface on the left face between adjacent blocks. (a) Standard Gauss–Seidel (GS) damping exhibits visible separation and misalignment at the contact interface. In contrast, (b) Global RL and (c) Subdomain RL maintain tighter and more uniform contact alignment, preserving the geometric integrity of the block assembly.

Table 4: Comparison of adaptive sub-stepping across three simulation scenarios. We report the average constraint error and number of substeps produced by the Heuristic method [Giles and Andrews 2024], and our RL-based method with and without subdomain partitioning.

	Heuristic Sub-stepping		Global RL		Subdomain RL	
Scenario	Constraint Error	Substeps	Constraint Error	Substeps	Constraint Error	Substeps
Rope Bridge	1.6e-04	22.9	3.0e-04	11.9	1.2e-03	16.3
Swinging Boxes	2.7e-04	18.7	2.1e-04	12.8	7.6e-05	29.4
Hydraulic Rig	1.8e-02	19.4	8.8e-03	14.7	1.0e-02	10.1

Table 5: Comparison of adaptive damping across three simulation scenarios. We report the average constraint error and damping value $\|\beta\|_1$ produced by the Heuristic method [Andrews et al. 2017], and our RL-based method with and without subdomain partitioning.

	Heuristic Damping		Global RL		Subdomain RL	
Scenario	Constraint Error	Damping	Constraint Error	Damping	Constraint Error	Damping
Rope Bridge	5.8e-03	1.6	3.9e-03	6.6	5.5e-03	7.4
Swinging Boxes	4.4e-03	1.7	3.3e-03	5.5	3.7e-03	8.2
Hydraulic Rig	2.3e-02	0.1	2.4e-02	1.2	2.4e-02	4.7

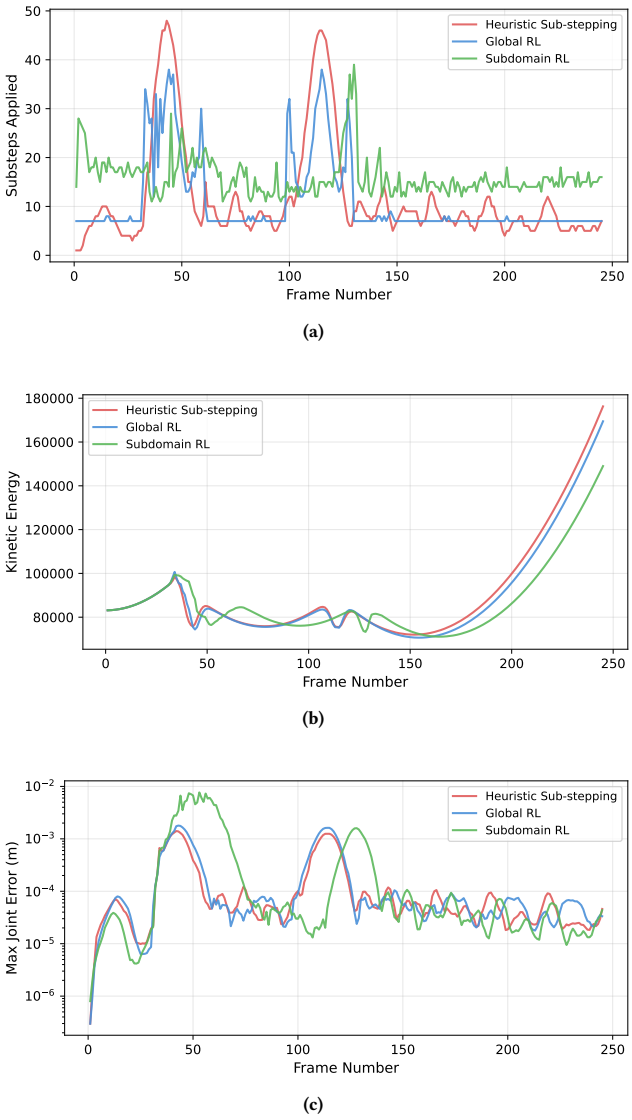


Figure 6: Evaluation of the Rope Bridge example using the Block Principal Pivoting (BPP) method, a direct solver, over an unobserved test trajectory of 250 frames), reporting: (a) sub-steps per frames, (b) total kinetic energy, and (c) max constraint error.

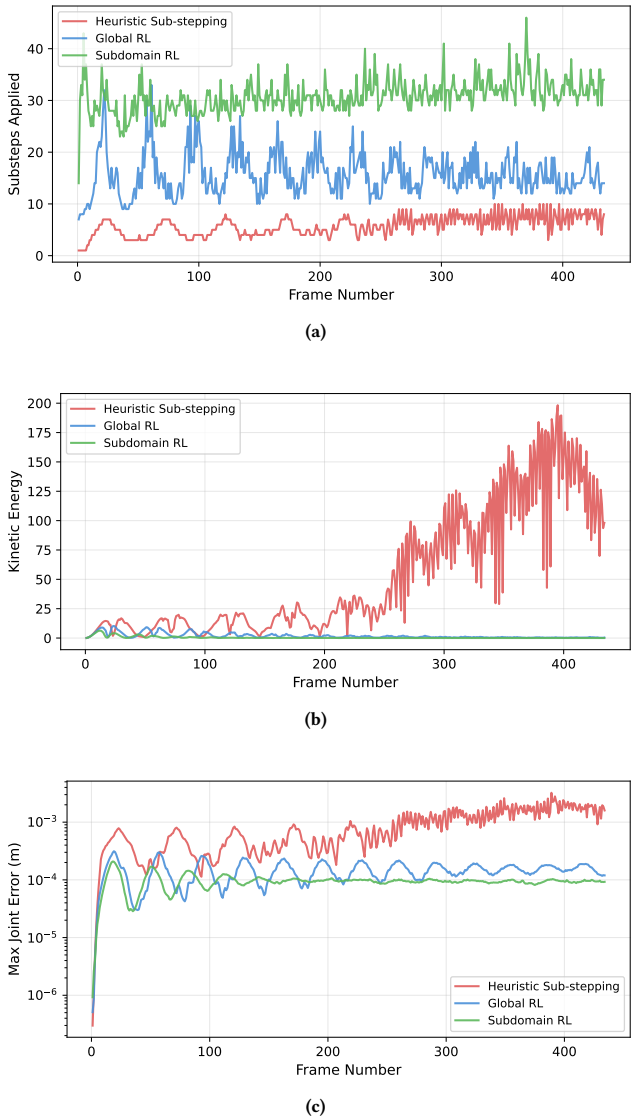


Figure 7: Ablation benchmark on an unloaded Rope Bridge using PGS with 25 iterations, reporting: (a) sub-steps per frame, (b) total kinetic energy, and (c) max constraint error.

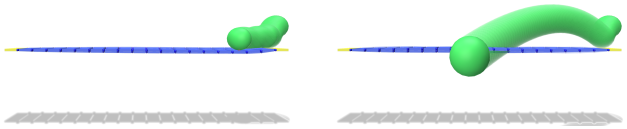
number of required sub-steps per frame. Conversely, Subdomain RL excels at localized error minimization and improves constraint satisfaction. In complex benchmark environments such as the *Hydraulic Rig*, both RL formulations strictly dominate the Heuristic Sub-stepping method across both constraint error reduction and stepping efficiency.

A similar trend holds for adaptive damping (Table 5), where learned policies typically give lower constraint violation compared to the heuristic damping baseline. Global RL Damping achieves the highest constraint accuracy in articulated scenarios. All methods selectively apply numerical dissipation to target transient dynamic instabilities that arise. However, the RL methods produce, on average, higher damping than the Heuristic Damping method, possibly leading to increased energy dissipation in the simulation.

Furthermore, as demonstrated in Figure 9, the learned policies generalize effectively to randomized dynamic environments, maintaining structural stability and strict constraint enforcement under unobserved impact loads.

5 Conclusions

In this work, we presented an RL framework for dynamic parameter adaptation in rigid body simulations characterized by stiff



(a) (b)

Figure 8: Trajectory comparison for a heavy ball ($m = 1000\,\text{kg}$, $v_0 = 1.0\,\text{m/s}$) crossing the Bridge. (a) Proposed RL method (stable): Maintains stability via adaptive sub-stepping. (b) Heuristic method (unstable): Oversized initial time step upon contact causes severe penetration and numerical instability.

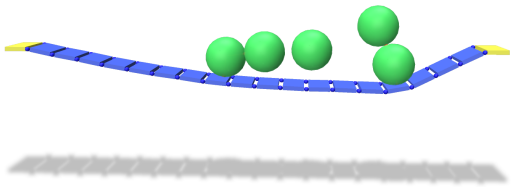


Figure 9: Generalization capability of the learned damping policy in a randomized dynamic environment featuring multiple balls dropping onto the bridge.

constraints, large mass ratios, and non-smooth frictional contacts. By dynamically modulating localized damping and temporal sub-stepping across spatial subdomains, our approach maintains numerical stability while reducing computational overhead compared to conservative geometric heuristics. Empirical evaluations across complex multi-body scenarios demonstrate that both our global and subdomain control strategies successfully preserve system kinetic energy—preventing numerical explosion during high-energy impacts without introducing artificial stiffness or globally over-damping physical motion. Furthermore, both strategies exhibit strong generalization capabilities across unseen initial conditions, establishing learning-based control as a scalable and efficient paradigm for stiff physics-based rigid body simulations with constraints.

Acknowledgments

The Bellairs Workshop on Computer Animation was instrumental in the conception of the research presented in this paper. This work was partially supported by the NSERC grants RGPIN 2024-04523, RGPIN 2025-05982, and ALLRP 571411-21, and FRQNT 2023-NOVA-315274. This work was also supported in part by resources provided by the Digital Research Alliance of Canada and Calcul Quebec.

References

S. Andrews, K. Erleben, and Z. Ferguson. 2022. Contact and friction simulation for computer graphics. In *ACM SIGGRAPH 2022 Courses* (Vancouver, British Columbia, Canada) *(SIGGRAPH '22)*. Association for Computing Machinery, New York, NY, USA, Article 3, 172 pages. doi:10.1145/3532720.3535640

S. Andrews, M. Teichmann, and P. G. Kry. 2017. Geometric Stiffness for Real-time Constrained Multibody Dynamics. *Computer Graphics Forum* 36, 2 (2017), 235–246. doi:10.1111/cgf.13122

K. Arnavaz, M. K. Nielsen, P. G. Kry, M. Macklin, and K. Erleben. 2023. Differentiable depth for real2sim calibration of soft body simulations. *Computer Graphics Forum* 42, 1 (2023), 277–289. doi:10.1111/cgf.14751

M. Cline and D. Pai. 2003. Post-stabilization for rigid body simulation with contact and constraints. In *2003 IEEE International Conference on Robotics and Automation*, Vol. 3. 3744–3751. doi:10.1109/ROBOT.2003.1242171

D. Dinev, T. Liu, J. Li, B. Thomaszewski, and L. Kavan. 2018. FEPR: fast energy projection for real-time simulation of deformable objects. *ACM Trans. Graph.* 37, 4, Article 79 (2018), 12 pages. doi:10.1145/3197517.3201277

K. Erleben. 2007. Velocity-based shock propagation for multibody dynamics animation. *ACM Trans. Graph.* 26, 2 (2007). doi:10.1145/1243980.1243986

C. Giles and S. Andrews. 2024. Adaptive Sub-stepping for Constrained Rigid Body Simulations. In *Proceedings of the 17th ACM SIGGRAPH Conference on Motion, Interaction, and Games* (Arlington, VA, USA) *(MIG '24)*. Association for Computing Machinery, New York, NY, USA, Article 5, 6 pages. doi:10.1145/3677388.3696331

D. Harmon, E. Vouga, B. Smith, R. Tamstorf, and E. Grinspun. 2009. Asynchronous contact mechanics. *ACM Trans. Graph.* 28, 3, Article 87 (2009), 12 pages. doi:10.1145/1531326.1531393

J. Ichnowski, P. Jain, B. Stellato, G. Banjac, M. Luo, F. Borrelli, J. E. Gonzalez, I. Stoica, and K. Goldberg. 2021. Accelerating quadratic optimization with reinforcement learning. In *Proceedings of the 35th International Conference on Neural Information Processing Systems (NIPS '21)*. Curran Associates Inc., Red Hook, NY, USA, Article 1610, 13 pages. doi:10.48550/arXiv.2107.10847

K. M. Jatavallabhula, M. Macklin, F. Golemo, V. Voleti, L. Petrini, M. Weiss, B. Considine, J. Parent-Lévesque, K. Xie, K. Erleben, et al. 2021. gradsim: Differentiable simulation for system identification and visuomotor control. *International Conference on Learning Representations (ICLR)* (2021). doi:10.48550/arXiv.2104.02646

A. Kuznetsov, P. Shvechikov, A. Grishin, and D. Vetrov. 2020. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*. JMLR.org, Article 515, 11 pages. doi:10.48550/arXiv.2005.04269

M. Li, Z. Ferguson, T. Schneider, T. Langlois, D. Zorin, D. Panozzo, C. Jiang, and D. M. Kaufman. 2020. Incremental potential contact: intersection-and inversion-free, large-deformation dynamics. *ACM Trans. Graph.* 39, 4, Article 49 (2020), 20 pages. doi:10.1145/3386569.3392425

Y. Liu and S. Andrews. 2022. Graph Partitioning Algorithms for Rigid Body Simulations. In *Eurographics 2022 - Short Papers*. The Eurographics Association. doi:10.2312/egs.20221036

M. Macklin, K. Erleben, M. Müller, N. Chentanez, S. Jeschke, and T. Y. Kim. 2020. Primal/dual descent methods for dynamics. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Virtual Event, Canada) *(SCA '20)*. Eurographics Association, Goslar, DEU, Article 9, 12 pages. doi:10.1111/cgf.14104

M. Macklin, K. Storey, M. Lu, P. Terdiman, N. Chentanez, S. Jeschke, and M. Müller. 2019. Small steps in physics simulation. In *Proceedings of the 18th annual ACM siggraph/eurographics symposium on computer animation*. 1–7.

S. Pawar and R. Maulik. 2021. Distributed deep reinforcement learning for simulation control. *Machine Learning: Science and Technology* 2, 2 (2021), 025029. doi:10.1088/2632-2153/abdaf8

A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. 2021. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research* 22, 268 (2021), 1–8.

D. M. Riley, A. Stathas, D. Gutiérrez-Oribio, and I. Stefanou. 2025. Reinforcement Learning-based adaptive time-integration for nonsmooth dynamics: DM Riley et al. *Machine Learning for Computational Science and Engineering* 1, 2 (2025), 44. doi:10.1007/s44379-025-00048-6

T. G. Ritto, S. Beregi, and D. A. W. Barton. 2022. Reinforcement learning and approximate Bayesian computation for model selection and parameter calibration applied to a nonlinear dynamical system. *Mechanical Systems and Signal Processing* 181 (2022), 109485. doi:10.1016/j.ymssp.2022.109485

V. Saz Ulibarrena and S. Portegies Zwart. 2025. Reinforcement learning for adaptive time-stepping in the chaotic gravitational three-body problem. *Communications in Nonlinear Science and Numerical Simulation* 145 (2025), 108723. doi:10.1016/j.cnsns.2025.108723

Y. Tian, M. A. Chao, C. Kulkarni, K. Goebel, and O. Fink. 2022. Real-time model calibration with deep reinforcement learning. *Mechanical Systems and Signal Processing* 165 (2022), 108284. doi:10.1016/j.ymssp.2021.108284

R. Tonge, F. Benevolenski, and A. Voroshilov. 2012. Mass splitting for jitter-free parallel rigid body simulation. *ACM Trans. Graph.* 31, 4, Article 105 (2012), 8 pages. doi:10.1145/2185520.2185601

M. Tournier, M. Nesme, B. Gilles, and F. Faure. 2015. Stable constrained dynamics. *ACM Trans. Graph.* 34, 4, Article 132 (2015), 10 pages. doi:10.1145/2766969

K. Um, R. Brand, Y. R. Fei, P. Holl, and N. Thuerey. 2020. Solver-in-the-Loop: Learning from Differentiable Physics to Interact with Iterative PDE-Solvers. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 6111–6122.

Z. Xiong, A. Leach, G. Thomas, and S. Sueda. 2025. Two-Pass Shock Propagation for Stable Stacking with Gauss-Seidel. *Proc. ACM Comput. Graph. Interact. Tech.* 8, 4, Article 46 (2025), 18 pages. doi:10.1145/3747867

Z. Zhang. 2025. Application of deep reinforcement learning in parameter optimization and refinement of turbulence models. *Scientific Reports* 15, 1 (2025), 25236. doi:10.1038/s41598-025-09958-6