Fast & Stable Control of Coupled Solid-Fluid Dynamic Systems Supplemental Material

ANONYMOUS AUTHOR(S)
SUBMISSION ID: 2360

CCS Concepts: • **Computing methodologies** → **Physical simulation**.

Additional Key Words and Phrases: Fluid-solid coupling, optimal control, reinforcement learning

ACM Reference Format:
Anonymous Author(s). 2025. Fast & Stable Control of Coupled Solid-Fluid Dynamic Systems Supplemental Material . *ACM Trans. Graph.* 1, 1, Article 1 (September 2025), 3 pages. https://doi.org/XXXXXXX.XXXXXXX

A Additional Details for the Control Framework

A.1 Additional Algorithm Details

Our control framework for the coupled solid-fluid systems consists of the following components:

- Two actor networks $\pi_1(\cdot;\phi_1)$ and $\pi_2(\cdot;\phi_2)$.
- Two target actor networks $\pi_1(\cdot;\phi_1')$ and $\pi_2(\cdot;\phi_2')$.
- Two value function networks $Q(\cdot;\theta_1)$ and $Q(\cdot;\theta_2)$.
- Two target value function networks $Q(\cdot;\theta_1')$ and $Q(\cdot;\theta_2')$.
- Pink noise for exploration.
- A pretrained autoencoder for fluid velocity field feature.

Velocity Feature Extraction. Prior to the reinforcement learning (RL) training phase, we conduct a comprehensive data collection phase, wherein fluid velocity fields data $\mathbf{X}$ are amassed from a diverse array of simulated trajectories by executing randomized actions. Subsequently, the data $\mathbf{X}$ is employed to train the auto-encoder, with the optimization process guided by the Mean Squared Error (MSE) loss function:

$$\mathcal{L}_{\text{mse}} = \frac{1}{n}\sum_{i=1}^{n} \|x_i - D(E(x_i))\|^2, x \in \mathbf{X} \tag{1}$$

where $E(\cdot)$ denotes the encoder and $D(\cdot)$ denotes the decoder. Then in both the sampling and training phases of RL policy, we employ the low-dimensional latent representations derived from the pre-trained $E(\cdot)$ as the fluid velocity field feature.

Exploration Noise. In the realm of exploration strategies, deterministic methods typically incorporate noise into action selection, often employing Gaussian white noise or Ornstein-Uhlenbeck (OU) noise, which is analogous to Brownian motion [Uhlenbeck and Ornstein 1930] and called red noise. Nevertheless, these noise mechanisms are often insufficient to promote effective exploration. White noise, characterized by its time-independence, often results in prolonged exploration periods, which can lead to insufficient state space coverage and the potential neglect of high-reward regions. In contrast, red noise, a time-dependent stochastic process, significantly enhances exploration efficiency and is therefore recommended as the default exploration strategy for DDPG [Lillicrap et al. 2016]. Nonetheless, the action space constraints are susceptible to being breached due to the indefinite escalation of variance over time. Consequently, we employ pink noise [Eberhard et al. 2023], a stochastic process that serves as an intermediary between white and red noise, to serve as the standard exploration noise for our deterministic off-policy agent.

Prior to introducing pink noise, we first explicate the concept of colored noise. A stochastic process is designated as colored noise with a colour parameter η if the signal $\varepsilon(t)$ extracted from the random process exhibits the property:

$$|\widehat{\varepsilon}(f)|^2 \propto f^{-\eta}, \tag{2}$$

where $\widehat{\varepsilon}(f) = F[\varepsilon(t)](f)$ denotes the Fourier transform of $\varepsilon(t)$, with f being the frequency, and $|\widehat{\varepsilon}(f)|^2$ is termed the power spectral density (PSD). For instance, when $\eta = 0$, the signal is temporally uncorrelated, implying that all frequencies are equally represented. This type of noise is classified as white noise. Conversely, when

Algorithm 1: RL with improved Bellman operator

Initialize critic Q_1, Q_2 and actor π_1, π_2 with $\theta_1, \theta_2, \phi_1, \phi_2$
Initialize target networks $\theta_1' \leftarrow \theta_1$, $\theta_2' \leftarrow \theta_2$, $\phi_1' \leftarrow \phi_1$, $\phi_1' \leftarrow \phi_1$
Initialzie replay buffer $\mathcal{D}$
for $t = 1$ *to* T **do**
  Select action a with pink noise based on π_1 and π_2
  Execute action a, observe reward r, new state s' and done d
  Store transition tuple (s, a, r, s', d) in $\mathcal{D}$ // d is the done flag
  for $i = 1, 2$ **do**
    Sample N transitions $\{(s, a, r, s', d), (s', a', r', s'', d')\}$ from $\mathcal{D}$
    Sample K noises $\epsilon \sim \mathcal{N}(0, \sigma)$
    $\hat{a}' \leftarrow \pi_i(s'; \phi_i') + \text{clip}(\epsilon, -c, c)$
    $\hat{Q}(s', \hat{a}') \leftarrow \min_{j=1,2}\left(Q_j(s', \hat{a}'; \theta_j')\right)$
    Calculate Softmax-$\hat{Q}_\beta(s')$
    $y_i \leftarrow r + \gamma(1-d)$Softmax-$\hat{Q}_\beta(s')$
    Do same with (s', a', r', s'', d') to get y_i'
    $y_i \leftarrow \max\left(y_i, r + \gamma(1-d)\, y_i'\right)$
    Update the critic θ_i according to Bellman loss:
    $\frac{1}{N}\sum_s (Q_i(s, a; \theta_i) - y_i)^2$
    Update actor ϕ_i by policy gradient:
    $\frac{1}{N}\sum_s \left[\nabla_{\phi_i}(\pi(s; \phi_i))\nabla_a Q_i(s, a; \theta_i)|_{a=\pi(s;\phi_i)}\right]$
    Update target networks:
    $\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$, $\phi_i' \leftarrow \tau\phi_i + (1-\tau)\phi_i'$
  end
end

ACM 1557-7368/2025/9-ART1
https://doi.org/XXXXXXX.XXXXXXX

Table 1. Hyperparameter Settings for Reinforcement Learning

Category	Hyperparameter	Value
Policy Optimization	Target policy noise σ	$\mathcal{N}(0,\ 0.2^2)$
	Noise clipping range c	$[-0.5,\ 0.5]$
Network Architecture	Hidden layer dimensions	[400, 300]
	Activation function	ReLU
	Layer normalization	Disabled
Exploration	Random exploration steps	8×10^4
	Noise type	Pink noise
Training	Discount factor γ	0.99
	Replay buffer size	1×10^6
	Batch size	256
	Target update rate τ	0.005
Optimizer	Learning rate α	3×10^{-4}
	Optimizer	Adam
Softmax Operator	Noise sample size K	50
	Softmax temperature β	10
FEDG	Guidance probability	0.3

$\eta = 2$, the noise is characterized as red noise. Pink noise, with $\eta = 1$, strikes an optimal balance by offering an intermediate level of temporal correlation between white and red noise, making it a more suitable default choice for exploration noise in our policy.

Critic Networks. We use dual critic networks $Q(\bullet, \theta_1)$ and $Q(\bullet, \theta_2)$ that simultaneously estimate Q-values (expected cumulative future rewards) for state-action pairs. This dual-estimator design effectively reduces variance in value estimation.

Actor Networks. Similarly, our architecture employs dual actor networks $\pi_1(\bullet, \phi_1)$ and $\pi_2(\bullet, \phi_2)$ that generate action a based on the current environmental state s. During training, these policy networks undergo continuous optimization through gradient ascent to progressively improve action selection quality. At each timestep t, the action a_t is determined through a competitive evaluation process between twin critics, formally defined as:

$$a_t = \pi_k\left(s_t; \phi_k\right), \quad k = \underset{i \in \{1,2\}}{\arg\max}\, Q_i\left(s, \pi_i\left(s_t; \phi_i\right); \theta_i\right). \tag{3}$$

This selection protocol ensures the agent follows the most optimistic evaluation from its paired value estimators. And the action a_t of the fluid driver is constrained to the normalized range $[-1, 1]$.

Target Networks. The target networks are copies of the actor and critic networks. They enhance the stability of deep reinforcement learning by maintaining fixed or slowly updated copies of the actor and critic networks. By decoupling the target values from immediate updates, they prevent harmful feedback loops and reduce the risk of divergence during training. This mechanism provides more consistent temporal difference learning targets, leading to smoother convergence and improved policy performance. Their use is particularly effective in value-based and actor-critic methods, where stable target estimation is crucial for successful learning.

Policy Optimization. The overall flow of the policy optimization algorithm is shown in Algorithm 1. We obtain a more accurate value estimation function, facilitating gradient computation and policy optimization, thereby significantly improving the performance in complex coupled solid-fluid control tasks.

A.2 Hyperparameters

Table 2. 2D AutoEncoder Encoder Architecture

Layer	Kernel	Stride	#Filters	Activation
Conv1	(7, 7)	(2, 2)	$2 \rightarrow 64$	LeakyReLU (0.1)
Conv2	(7, 7)	(2, 2)	$64 \rightarrow 64$	LeakyReLU (0.1)
Conv3	(5, 5)	(2, 2)	$64 \rightarrow 64$	LeakyReLU (0.1)
Conv4	(5, 5)	(2, 2)	$64 \rightarrow 128$	LeakyReLU (0.1)
Conv5	(5, 5)	(2, 2)	$128 \rightarrow 128$	LeakyReLU (0.1)
Conv6	(2, 2)	(2, 2)	$128 \rightarrow 128$	LeakyReLU (0.1)
Conv7	(2, 2)	(1, 1)	$128 \rightarrow 64$	LeakyReLU (0.1)

Table 3. 3D AutoEncoder Encoder Architecture

Layer	Kernel	Stride	#Filters	Activation
Conv1	(3, 3, 3)	(2, 2, 2)	$3 \rightarrow 64$	LeakyReLU (0.1)
Conv2	(3, 3, 3)	(2, 2, 2)	$64 \rightarrow 64$	LeakyReLU (0.1)
Conv3	(3, 3, 3)	(2, 2, 2)	$64 \rightarrow 128$	LeakyReLU (0.1)
Conv4	(5, 3, 2)	(1, 1, 1)	$128 \rightarrow 128$	LeakyReLU (0.1)
Conv5	(5, 3, 2)	(1, 1, 1)	$128 \rightarrow 128$	LeakyReLU (0.1)
Conv6	(2, 2, 2)	(1, 1, 1)	$128 \rightarrow 64$	LeakyReLU (0.1)

Hyperparameters for Autoencoder. The network architecture of the fluid velocity field autoencoder is detailed in Tables 2 and 3. Each convolutional layer is followed by BatchNorm, and the decoder follows a symmetric structure. The input dimensions are $(2, 128, 128)$ for 2D velocity fields and $(3, 80, 40, 32)$ for 3D fields. During pre-training, the autoencoder is trained with the Adam optimizer at a learning rate of 0.001.

A.2.1 Hyperparameters for RL. For reproducibility, we provide the full hyperparameter settings in Table 1, which includes essential configurations such as network size, optimizer, guidance probability, etc. Note that each task's action a_t in our work is constrained to $[-1, 1]$.

B Experimental Details

B.1 Experimental Settings

All experiments are conducted on a workstation equipped with an NVIDIA GeForce RTX 3090 GPU (24GB VRAM). The software stack includes:

- Python 3.8.12
- Taichi 1.6.0
- PyTorch 1.10.0
- OpenAI Gym 0.26.2

Table 4. Benchmark configurations and computational costs for RL training. In the table, G denotes gravitational acceleration, E represents Young's modulus, and Δt specifies the simulation timestep size. Density Ratio is defined as the ratio of the solid material's density to that of water. Numsteps indicates the number of state transitions collected for policy training, and Cost refers to the total computational time required to complete the training process.

Benchmark	Task	Grid Resolution	#Particles	G	E	Density Ratio	$\Delta t(s)$	Numsteps	Cost
Squeeze	Squeeze with Double Walls	128×128	25K	60	1400	0.4–0.7	2×10^{-4}	5×10^6	50h
	Squeeze with Single Wall	128×128	25K	60	1400	0.4–0.7	2×10^{-4}	5×10^6	47h
	Squeeze Target Balls	128×128	25K	60	1400	0.4–0.7	2×10^{-4}	5×10^6	51h
Scoop	Scoop Balls	128×128	25K	60	1400	0.4–0.7	2×10^{-4}	5×10^6	34h
	Scoop Target Balls	128×128	25K	70	900	0.6	1×10^{-4}	5×10^6	60h
Balance	Single Ball Balance	$80 \times 40 \times 32$	26K	60	800	4	2×10^{-4}	2×10^6	15h
	Double Ball Balance	$80 \times 40 \times 32$	27K	60	800	4	2×10^{-4}	2×10^6	16h
Transport	Transport in X-axis	$80 \times 40 \times 32$	26K	60	800	4	2×10^{-4}	5×10^6	40h
	Transport in 3D Space	$80 \times 40 \times 32$	26K	60	800	4	2×10^{-4}	5×10^6	60h
Music	Single Solid Music Palyer	$80 \times 40 \times 32$	26K	60	800	4	2×10^{-4}	5×10^6	51h
	Double Solid Music Player	$80 \times 40 \times 32$	52K	60	800	4	2×10^{-4}	5×10^6	83h

This configuration ensures full hardware acceleration and software compatibility throughout our experiments. During training, we evaluate the policy 10 times every 25,000 timesteps. For the curves, we conduct experiments with at least 3 different random seeds and report the mean performance, with the shaded regions indicating 95% confidence intervals.

B.2 Benchmark Details

For all the benchmarks, we use a right-handed coordinate system in both 2D and 3D spaces. In 2D, the x-axis points right and the y-axis points up. In 3D, the z-axis points outward, perpendicular to the xy-plane. The simulation parameters and RL training statistics are included in the benchmark details listed in table 4. We then provide further details on each benchmark task.

B.2.1 Squeeze Benchmark (2D). For the squeeze with double walls task, the state s has a dimension of 88, consisting of: (1) 64 fluid velocity field features, (2) 4 wall features (positions and velocities of two walls along the x-axis), and (3) 20 ball features (positions and velocities of 5 balls in both x and y directions). In comparison, the squeeze with single wall task has a state dimension of 86 (with one less wall's features). The episode terminates only when reaching the maximal length for above tasks. The squeeze target balls task features an expanded state dimension of 93, which includes additional binary indicators representing whether each of the five balls is good or bad. It is terminated when a bad ball enters the net.

B.2.2 Scoop Benchmark (2D). The scoop balls task uses a state s representation comprising 90 dimensions: (1) 64 fluid dynamics features; (2) 20 ball state features (positions and velocities); and (3) 6 spoon features (2D position, 2D velocity, angular position, and angular velocity). The action space is three-dimensional, controlling the spoon's linear accelerations in the x and y directions, as well as its angular acceleration. For the scoop balls task, which has a state space of 118 dimensions, we implemented Hindsight Experience Replay (HER) for all off-policy RL algorithms during training.

B.2.3 Balance Benchmark (3D). For the single ball balance task, the state s has 81 dimensions. The reward function coefficients are configured as follows: $\omega_1 = 2.5$, $\omega_2 = 1.5$ and $\omega_3 = 25$. A distance threshold of $\bar{d}_{xy} = 0.05$ (equal to the ball radius) is used to guide the fluid spout towards the target. The maximum episode length is set to 5,000 time steps. The double ball balance variant extends the dimensionality of the state to 90, while using the same threshold of $\bar{d}_{xy}$=0.05, but with a maximum episode length of just 1,000 steps.

B.2.4 Transport Benchmark (3D). For the transport in x-axis task, we employ a 90-dimensional state representation with a target distance threshold of $\bar{d}_x = 0.17$. The maximum episode length is configured at 2,000 time steps, with the $p_b^\star$ being updated every 250 time steps during training. The transport in 3D space variant maintains similar configurations but with adjusted distance thresholds: $\bar{d}_x = 0.4$ and $\bar{d}_y = 0.12$.

B.2.5 Music Benchmark (3D). For the single solid music player task, we employ an 88-dimensional state representation with an inter-note interval of 100 interaction timesteps. The reward function coefficients are configured as μ_1=35, μ_2=0.1, μ_3=10, and μ_4=25. In our FEDG setting, the agent is required to position itself beneath target keys within a 20% temporal error margin. The maximum episode duration is set to 3,000 timesteps. The double solid music player variant extends the state dimensionality to 112, with the same configuration parameters.

REFERENCES

Onno Eberhard, Jakob Hollenstein, Cristina Pinneri, and Georg Martius. 2023. Pink Noise Is All You Need: Colored Noise Exploration in Deep Reinforcement Learning. In *Proceedings of the Eleventh International Conference on Learning Representations (ICLR 2023)*. OpenReview.net, Kigali, Rwanda. https://openreview.net/forum?id=hQ9V5QN27eS

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, (ICLR 2016)*. Open Publishing, San Juan, Puerto Rico.

G. E. Uhlenbeck and L. S. Ornstein. 1930. On the Theory of the Brownian Motion. *Phys. Rev.* 36 (Sep 1930), 823–841. Issue 5. https://doi.org/10.1103/PhysRev.36.823