OptiMARL: an optimal latent space–based multiagent reinforcement learning method for subgrid-scale model discovery in compressible flows

By D. Brouzet

While recent reinforcement learning (RL) approaches, notably multiagent reinforcement learning (MARL), have shown promise for automated subgrid-scale (SGS) model discovery, they face critical challenges when applied to compressible flows with localized features such as shocks. This study introduces OptiMARL, a novel framework that addresses these limitations through adaptive latent space (LS) control. The key innovation is the deployment of agents in an optimal subspace of the state vector, determined dynamically by a dedicated RL agent. This approach eliminates the need for spatial interpolation and enables precise representation of localized flow features. Application to the one-dimensional (1D) Burgers equation demonstrates superior performance compared with standard approaches such as the fifth-order targeted essentially nonoscillatory scheme (TENO) and standard MARL. The method achieves near-DNS (direct numerical simulation) accuracy in both spectral content and temporal evolution while automatically discovering that projections onto the first and second velocity derivative spaces provide optimal representations of the Smagorinsky constant. As a result, the framework assigns spatially varying Smagorinsky constants that adapt to local flow physics, resulting in sharp shock profiles with minimal numerical oscillations. This proof of concept demonstrates the potential of LS-based RL for developing adaptive and physics-consistent SGS models for compressible and multiphase flow applications.

1. Introduction

SGS model development for turbulent flows traditionally relies on physical insights and dynamical modeling and, for complex applications, may require empirical tuning of parameters. In contrast, data-driven machine learning techniques can discover nonalgebraic closure policies that adapt to a wide range of flow conditions.

RL offers several key advantages over supervised machine learning and *a priori* model development approaches. Unlike supervised learning methods that train on single-step target values from filtered DNS data, RL optimizes long-term cumulative rewards through integration within the filtered equations, thereby accounting for compounding modeling errors and temporal evolution of discrepancies between DNS and large-eddy simulation (LES). Furthermore, RL addresses a critical issue wherein supervised learning models that perform well in *a priori* testing can fail in *a posteriori* testing as a result of inconsistent filtering operators or error accumulation, for instance. Finally, RL does not necessarily require expensive DNS databases for training, as it can learn from statistical measures of flow quantities that may be obtained from experiments rather than simulations.

There are two main categories of SGS modeling. In implicit modeling, the discretization scheme itself acts as the filter function, whereas in the explicit approach, the SGS term must be explicitly modeled but numerical dissipation must be kept to a minimum. Kurz *et al.* (2023) and Beck & Kurz (2023) developed discretization-consistent closure schemes that optimize both explicit eddy-viscosity models and implicit modeling strategies. A significant contribution by Beck & Kurz (2023) is an RL-optimized hybrid discontinuous Galerkin and finite-volume blending scheme that adapts the discretization operator itself as an implicit closure model.

That said, most of the relevant literature has focused exclusively on explicit SGS modeling. Novati *et al.* (2021) introduced MARL for the discovery of SGS models using LES of homogeneous isotropic turbulent flows. They used cooperating RL agents distributed throughout the flow domain to learn optimal turbulence closure policies by maximizing long-term rewards based on matching DNS energy spectra. Kim *et al.* (2022) showed that RL approaches could be beneficial in more complex configurations by applying RL to wall-bounded turbulent flows, where anisotropic near-wall effects are predominant. Their key innovation lies in developing a physics-constrained RL framework that satisfies reflectional invariance and wall boundary conditions without requiring additional training processes. Kim *et al.* (2022) also introduced several training efficiency innovations, including direct reward accumulation, spatially and temporally correlated exploration strategies, and pretraining processes specifically designed to handle the high-dimensional nature of the problem. Other works have focused on Reynolds-averaged Navier–Stokes modeling with RL methods, for instance, by augmenting the Spalart–Allmaras turbulence model with a nonlinear term (Fuchs *et al.* 2024) or by combining symbolic regression and RL to discover explicit algebraic Reynolds stress closures (Tang *et al.* 2020).

Bae & Koumoutsakos (2022) further extended MARL to wall-modeling strategies in turbulent boundary layers, developing a physics-informed architecture that embeds boundary layer physics directly into the RL framework through a log-law-based wall model. Vadrot *et al.* (2023) improved upon this wall-modeling framework by developing state normalization techniques that enable training on a single Reynolds number while achieving generalization across extreme Reynolds number ranges. Zhou *et al.* (2023) also demonstrated the engineering applicability of MARL by successfully applying it to complex separated flows over the Boeing Gaussian bump, showing generalizability to aerodynamic flows with strong pressure gradients.

Despite these promising advances in RL-based turbulence modeling, several critical challenges remain that limit the broader applicability of these methods. First, the extension to more complex flow physics such as compressible and multiphase flows remains largely unexplored. The presence of highly localized shocks and abrupt density changes is expected to lead to difficulties in the RL training process. Second, while some studies either use data from all grid points (Kim *et al.* 2022) or treat training as a localized process (Beck & Kurz 2023; Kurz *et al.* 2023), the dimensionality of the RL problem becomes prohibitively large ($\geq \mathcal{O}(10^5)$), leading to substantial computational costs and fundamental training difficulties. Subel *et al.* (2021) noted that their attempts to train deep neural network (NN) models with dimensions $\mathcal{O}(10^5)$ resulted in inaccurate predictions in *a priori* tests and unstable LES simulations in *a posteriori* tests as a result of overfitting issues. While their use of data augmentation strategies mitigated these problems, more generalizable and automatic methods are needed. Third, the MARL approach that addresses dimensionality issues by reducing the problem to $\mathcal{O}(10)$ agents typically relies on spatial interpolation schemes to extend agent actions to all grid points (Novati

et al. 2021; Bae & Koumoutsakos 2022; Vadrot *et al.* 2023). However, this interpolation strategy is inherently suboptimal and potentially problematic for flows with nonsmooth or highly localized features, where linear interpolation may fail to capture the complex spatial variations in turbulence closure requirements. These limitations highlight the need for methodological advances that can balance computational efficiency and model accuracy.

This study developed a novel MARL approach for application to compressible turbulent flows using the Burgers equation as a benchmark, which has been the subject of various SGS and reduced-order modeling studies (e.g. Love 1980; Dolaptchiev *et al.* 2013; Maulik & San 2018). The novelty of this methodology, termed OptiMARL, lies in the use of agents in an LS, defined as an optimal subspace of the state vector, and identified by a dedicated single RL agent. Section 2 describes the Burgers equation and related modeling strategies, while Section 3 presents the OptiMARL framework and methodology. The results are discussed in Section 4, and the conclusions are presented in Section 5.

2. Subgrid-scale modeling in the Burgers equation

2.1. *Mathematical framework*

The 1D Burgers equation is written as

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}, \tag{2.1}$$

where u is the velocity field and ν is the kinematic viscosity. To derive its filtered counterpart, akin to LES formulations, a spatial filter operator $G(x, \Delta)$ with filter width Δ is applied to Eq. (2.1), yielding

$$\frac{\partial \overline{u}}{\partial t} + \overline{u}\frac{\partial \overline{u}}{\partial x} = \nu\frac{\partial^2 \overline{u}}{\partial x^2} - \frac{\partial \tau_{sgs}}{\partial x}, \tag{2.2}$$

where $\overline{u}$ denotes the filtered velocity field and τ_{sgs} is the SGS stress that represents the effect of unresolved scales on the resolved scales, expressed in terms of filtered quantities as

$$\tau_{sgs} = \overline{u \cdot u} - \overline{u} \cdot \overline{u}. \tag{2.3}$$

The Burgers equation serves as an instructive model problem that exhibits fundamental similarities to both turbulent and compressible flows. In the inviscid limit and without any forcing term, the Burgers equation develops sharp gradients, analogous to the formation of shock waves in compressible flows. This shock formation mechanism provides an essential framework with which to assess the role of artificial viscosity in numerical schemes for numerical stability. Furthermore, the Burgers equation demonstrates energy transfer mechanisms that parallel those observed in turbulent flows. The nonlinear term drives energy from large scales to small scales through a cascade process, mirroring the Kolmogorov energy cascade in three-dimensional (3D) turbulence. As such, the spectrum in Burgers equation solutions exhibits power-law scaling reminiscent of the Kolmogorov spectrum, even though the former has a -2 logarithmic decay compared with $-5/3$ for the latter. These characteristics make the Burgers equation a valuable benchmark for developing and validating computational methods for both turbulence modeling and shock-capturing schemes.

2.2. *Flow conditions and simulation configuration*

This study considers a high-Reynolds-number regime with $Re = 1/\nu = 15{,}000$. The flow is initialized as a single sawtooth wave with an amplitude ranging from -1 to 1, spanning the periodic computational domain from $x = 0$ to $x = 1$ and discretized using $n_x = 100$ points. The simulations are run for 1000 iterations with a time step $\Delta t = 10^{-3}$.

2.3. *Subgrid-scale modeling*

The SGS stress term in Eq. (2.2) is explicitly modeled to close the system. One widely used approach is the static Smagorinsky (1963) model

$$\tau_{sgs} = -\nu_{sgs}\frac{\partial \bar{u}}{\partial x}, \tag{2.4}$$

$$\nu_{sgs} = C_s\Delta^2\left|\frac{\partial \bar{u}}{\partial x}\right|, \tag{2.5}$$

where ν_{sgs} is the SGS viscosity and C_s is the Smagorinsky constant with a typical value of 0.16. While the standard Smagorinsky model provides a simple closure, C_s may not be universally applicable across different flow configurations and grid resolutions. The dynamic Smagorinsky model (DSM), originally developed by Germano *et al.* (1991), addresses this limitation by dynamically computing C_s based on the resolved scales of the flow. The dynamic procedure introduces a test filter with width $\widehat{\Delta} > \Delta$ to create a hierarchy of filtered fields. The method exploits the assumption of scale similarity to relate the SGS stresses at different filter levels, resulting in the Germano identity

$$L_{ij} = T_{ij} - \widehat{\tau_{sgs}}, \tag{2.6}$$

$$T_{ij} = \widehat{\overline{u_i u_j}} - \widehat{\bar{u}}_i\widehat{\bar{u}}_j, \tag{2.7}$$

where $\widehat{\cdot}$ denotes filtering at scale $\widehat{\Delta}$ and L_{ij} represents the resolved stress tensor. Equation (2.6) allows for the dynamic calculation of the model coefficient through a least-squares minimization procedure in 3D configurations. For 1D configurations, this approach simplifies to

$$C_s = \frac{\langle L\rangle}{\langle M\rangle}, \tag{2.8}$$

with

$$L = \widehat{\overline{uu}} - \widehat{\bar{u}}\widehat{\bar{u}}, \tag{2.9}$$

$$M = \Delta^2\widehat{\left|\frac{\partial \bar{u}}{\partial x}\right|\frac{\partial \bar{u}}{\partial x}} - (\widehat{\Delta})^2\left|\frac{\partial \widehat{\bar{u}}}{\partial x}\right|\frac{\partial \widehat{\bar{u}}}{\partial x}, \tag{2.10}$$

where $\langle\cdot\rangle$ denotes an averaging operator that is used for convergence and can be temporal or spatial, depending on the case. This dynamic formulation provides better adaptability to local flow conditions, automatically adjusting the SGS viscosity based on the instantaneous state of the resolved field. The DSM theoretically and practically improves the accuracy and robustness of LES across a broader range of applications, from channel flows (Germano *et al.* 1991) to cloud-topped atmospheric boundary layers (Kirkpatrick *et al.* 2006). Here and below, the term Smagorinsky constant is retained even though C_s can be spatially and/or temporally varying.

2.4. *Numerical discretization*

Temporal integration is performed using the fourth-order Runge–Kutta method, with a time step satisfying the Courant–Friedrichs–Lewy stability condition. For spatial discretization, a fourth-order central finite-difference scheme is employed to approximate the spatial derivatives. This high-order central spatial discretization is purposefully chosen to avoid any numerical dissipation that could interfere with the SGS model's role in representing unresolved physics. With this strategy, the SGS term is responsible for both physical modeling and numerical stability in the presence of near-discontinuous shocks.

An alternative approach to numerical stability is to use specifically designed schemes to discretize the shock region. For flows exhibiting sharp gradients or shocklike structures, traditional central-difference schemes may indeed suffer from numerical oscillations and instabilities due to their nondissipative nature. In such cases, shock-capturing schemes maintain numerical stability while preserving solution accuracy. As an example, the TENO approach employs a scale-separation technique that distinguishes between smooth regions and areas containing discontinuities through the evaluation of local smoothness indicators. In smooth regions, TENO schemes automatically revert to central differences, minimizing numerical dissipation. Near discontinuities, the scheme activates adaptive dissipation mechanisms that prevent spurious oscillations while maintaining sharp interface resolution. This targeted approach results in significantly reduced numerical dissipation in smooth flow regions compared with traditional shock-capturing schemes.

3. OptiMARL framework

3.1. *Overview*

Figure 1 illustrates the MARL and OptiMARL frameworks for solving the Burgers equation using NN-based control policies. First, the Burgers equation solver computes the variable $u(x,t)$. The distributed agents compute the local state $s(x_i,t)$ at locations x_i based on the flow state and receive the scalar reward $r(t)$ based on the solution quality relative to the target solution. The NN policy $\pi_{w_1}(a_1|s)$, parameterized by weights w_1, processes these state observations to generate control actions a_1 that enable the computation of a spatially and temporally varying Smagorinsky constant $C_s(x,t)$, which is then used back in the solver. While the Smagorinsky model (Eq. (2.5)) is used as a framework for training and for helping the algorithm identify a stable SGS model, the policy eventually identified by the RL strategy can be significantly more complex because of the C_s variations.

In OptiMARL, the reward is also used as input for an additional single agent that defines the NN policy $\pi_{w_2}(a_2|\cdot)$, parameterized by weights w_2, to generate the discrete action a_2. This action controls which states s_k are used as a basis for the LS Ψ, in which the agents are defined at fixed locations. The spatial location x_i of these agents is therefore updated at every iteration, depending on Ψ and s_k.

Both MARL and OptiMARL approaches create a closed feedback loop that allows the system to learn spatially distributed control strategies while sharing a common policy network. In the context of the application to the Burgers equation, the states s_k are defined as the first four derivatives of u, namely $s_k = \partial^k u/\partial x^k$, where $k \in \{1,2,3,4\}$. Furthermore, C_s is constrained to be within the bounds $[10^{-5}, 5.0]$ to help convergence of the policy. The steps shown in Figure 1 are described in more detail below.

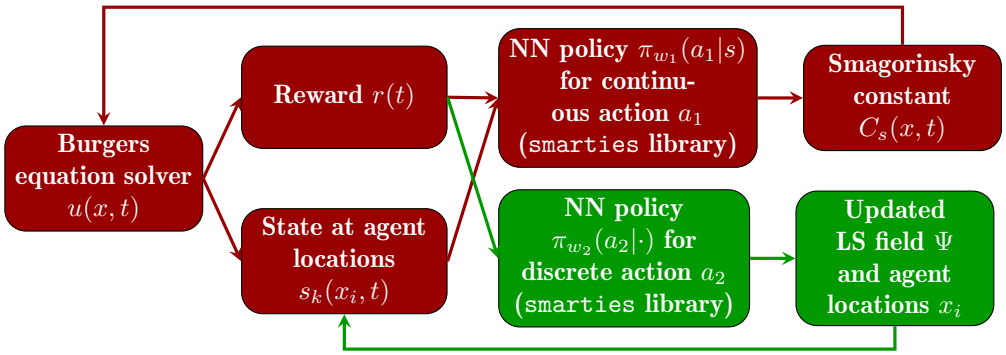

FIGURE 1. Schematic of the MARL (dark red) and OptiMARL (dark red + dark green) frameworks. LS: latent space. NN: neural network. w_1: weight parameters for policy π_{w_1}. w_2: weight parameters for policy π_{w_2}.

3.2. *The* `smarties` *reinforcement learning library*

The `smarties` library is an open-source RL framework specifically designed for high-performance scientific computing applications. The library was developed to facilitate the integration of MARL with computationally intensive simulations, such as computational fluid dynamics solvers. `smarties` efficiently leverages computing resources by separating the task of updating policy parameters from the task of collecting interaction data.

The `smarties` library implements a master–worker architecture where flow simulations are distributed across multiple worker processes that collect interaction data, while a central learning process (master) handles policy parameter updates. Each worker collects experiences organized into episodes for each agent, and when a simulation concludes, the worker sends one episode per agent to the central learning process and receives updated policy parameters. The master process stores episodes in replay memory, which is sampled to update policy parameters according to the Remember-and-Forget Experience Replay (ReF-ER) algorithm. ReF-ER combines the sample efficiency of experience replay with the stability of constrained policy updates, making it particularly suitable for MARL applications.

`smarties` also implements V-RACER, an off-policy actor–critic algorithm that supports continuous state and action spaces and is combined with ReF-ER. V-RACER trains an NN that, given input state s, outputs the mean $\mu_w(s)$ and standard deviation $\sigma_w(s)$ of the policy π_w, along with a state-value estimate $v_w(s)$. The policy gradient estimator is implemented as

$$g^{\text{pol}}(w) = E\left[(\tilde{q} - v_w(s)) \times \frac{\pi_w(a|s)}{P(a|\mu,\sigma)} \times \nabla_w \log \pi_w(a|s)\right], \tag{3.1}$$

where $P(a|\mu,\sigma)$ is the probability of sampling the action a from a Gaussian distribution with statistics μ and σ, and $\tilde{q}_t$ estimates the cumulative rewards using the Retrace algorithm.

ReF-ER addresses the instability issues common in experience replay methods by using importance weights γ to classify experiences as either near policy or far policy and clipping gradients from far-policy samples to zero. The gradient modification rule is

$$\tilde{g}(w) = \begin{cases} \beta\tilde{g}(w) - (1-\beta)g^D(w) & \text{if } \frac{1}{C} < \gamma < C, \\ -(1-\beta)g^D(w) & \text{otherwise,} \end{cases} \tag{3.2}$$

where $g^D(w) = \nabla_w D_{KL}(\pi_w(\cdot|s)\|P(\cdot|\mu,\sigma))$ is a penalization term and the coefficient β is iteratively updated to maintain a constant fraction of samples within the trust region.

The ReF-ER algorithm with hyperparameters $C = 1.5$ and $D = 0.05$ has been demonstrated to stabilize training in multiagent settings (Novati *et al.* 2021). Training runs are typically advanced for 10^7 policy gradient steps, and consistent training progress has been observed regardless of the initial random seed. The library's design makes it particularly suitable for scientific computing applications where large numbers of cooperating agents are required and integration with existing simulation frameworks is necessary. Further details about `smarties` are presented by Novati *et al.* (2021) and Bae & Koumoutsakos (2022).

3.3. *Reward function*

The reward $r(t)$ is computed on the basis of a DNS solution $u^{\mathrm{DNS}}(x,t)$ that is computed on a domain with much higher resolution ($n_x^{\mathrm{DNS}} = 4000$ points)

$$r(t) = \exp\left[-\frac{50}{n_x}\sum_{j=1}^{n_x}\left(u^{\mathrm{DNS}}(x_j,t) - u(x_j,t)\right)^2\right], \tag{3.3}$$

ensuring that $0 < r(t) \leq 1$. The reward is updated at every iteration as

$$r(t) = (1-\alpha)r(t-\Delta t) + \alpha r(t), \tag{3.4}$$

with the parameter $\alpha = 0.2$.

3.4. *Latent space for agent control*

In MARL, the agents have fixed, usually equidistant, spatial locations x_i, and C_s (or whichever physical variable the action represents) is then interpolated at every grid point according to its value at the agent locations. Several limitations are associated with this approach. First, the data sampling at agent locations is suboptimal because it represents only a subset of the complete simulation domain, which can result in slower policy convergence. Second, and more critically, the interpolation procedure can lead to significant misrepresentation of C_s by failing to capture the complex nonlinear spatial variations. Indeed, the action is defined by a normal distribution with mean μ_{w_1} and standard deviation $\sigma^2_{w_1}$

$$a_1(s) \sim \mathcal{N}(\mu_{w_1}(s), \sigma^2_{w_1}(s)). \tag{3.5}$$

Note that the standard deviation is nonzero for training purposes only. In a nontraining scenario, therefore, a_1 is a deterministic function of s. Mapping a_1 to spatiotemporal space leads to

$$a_1(x,t) = (a_1 \circ s)(x,t). \tag{3.6}$$

Equation (3.6) demonstrates that if either the function $a(s)$ or $s(x,t)$ exhibits nonsmooth behavior, the resulting function $a(x,t)$ may also be nonsmooth, potentially leading to problematic spatial interpolation issues.

The OptiMARL methodology, which algorithm can be found in Appendix A, was developed to address these issues. The important details are noted below.

(*a*) Inputs. The user selects N_{LS}, the number of dimensions of the LS Ψ, satisfying the condition $1 \leq N_{\mathrm{LS}} \leq N_s$, where N_s is the number of states. The total number of agents must satisfy the relation $N_{\mathrm{agents}} = n^{N_{\mathrm{LS}}}$, with $n \in \mathbb{N}$.

(*b*) Initialization. The agent locations in Ψ are then set as an equidistant grid of dimension N_{LS} with minimum and maximum values at zero and one, respectively.

	TENO5	DSM	MARL	OptiMARL
N_{agents}	N/A	N/A	3	4
N_{LS}	N/A	N/A	N/A	2

TABLE 1. Modeling approaches considered.

(*c*) Action generation. While N_{agents} agents govern the continuous action a_1, a single additional agent is dedicated to the discrete action a_2, which takes a value $\in [1, C(N_s, N_{\text{LS}})]$, where $C(N_s, N_{\text{LS}})$ is the binomial coefficient representing the cardinality of all N_{LS}-element subsets of an N_s-element set.

(*d*) LS update. The action a_2 serves as a combinatorial index that uniquely identifies, through a bijective mapping from integers to combinations, an N_{LS}-element subset of states with indices k_{LS}. Ψ is then computed as the field using the normalized state vectors $s_{k_{\text{LS}}}$ as the basis.

(*e*) Smagorinsky constant computation. The Smagorinsky constant is then linearly interpolated in Ψ space for all grid points based on the value a_1. A simple mapping $\Psi \rightarrow x$ enables the computation of $C_s(x, t)$.

(*f*) Agent spatial location update. Since there is no guarantee that the agent location in Ψ x_{LS} corresponds to the state at a grid point, the agent location in physical space is computed as the one closest to x_{LS} in an L_2 sense. This approximation is necessary to feed the RL algorithm a physical state present in the simulation.

4. Results

This study employs four distinct modeling approaches, as summarized in Table 1. The first approach utilizes the fifth-order TENO (TENO5) scheme from Fu *et al.* (2016) with a threshold constant $C_T = 1 \times 10^{-5}$ and without any explicit SGS model. The second approach implements the DSM using a box test filter of size $\widehat{\Delta} = 2\Delta x$. Through systematic testing, a spatial averaging operator over the full domain was determined to be optimal and is therefore employed in this study. The third approach applies MARL with three agents distributed equidistantly across the computational domain, and the fourth approach (OptiMARL) utilizes four agents positioned on a two-dimensional LS with two agents per dimension. For validation purposes, a reference DNS solution is computed on a domain discretized with 4000 grid points using the TENO5 scheme.

We begin by examining the convergence behavior of the cumulative reward $r(t)$ (Eq. (3.3)) in Figure 2 as a function of cumulative time steps, representing 1000 simulations. The curves represent the median values over a moving window of 50 time steps, while the shaded areas indicate the 20–80 interdecile range. For comparison, the rewards obtained with TENO5, DSM, and DNS are indicated by horizontal reference lines. The standard MARL policy exhibits convergence difficulties, as evidenced by persistent oscillations in the median value and a substantial 20–80 interdecile range. Its overall performance remains comparable to that of DSM. In contrast, the OptiMARL policy demonstrates robust convergence to a high reward value approaching the theoretical maximum within approximately 7×10^5 time steps. The algorithm requires several tens of thousands of time steps to identify the optimal LS, resulting in a sharp increase in cumulative reward. Ultimately, OptiMARL achieves a performance significantly superior to that of all

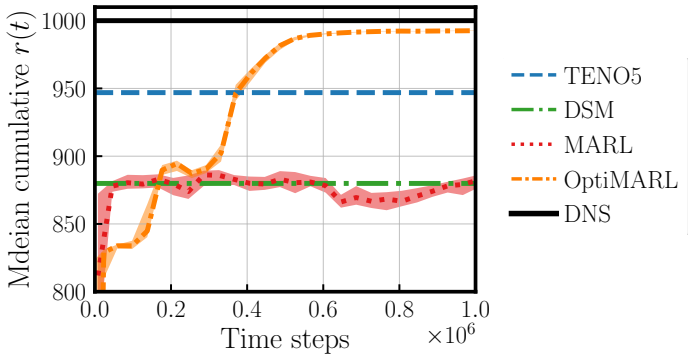

FIGURE 2. Median cumulative reward of the RL policies as a function of RL time steps for MARL and OptiMARL with representation of the 20–80 interdecile range (more than 50 time steps) by the shaded areas. The DNS, TENO5, and DSM rewards are also shown for reference.

other approaches, including TENO5, thereby demonstrating the algorithm's potential to identify breakthrough models in compressible flow applications.

The remainder of this section analyzes the results obtained with the MARL and OptiMARL policies identified at the end of training after 10^6 time steps. The u solution initialized with a sawtooth wave exhibits self-similar behavior, as the normalized solution $u/u_{\max}$ is temporally quasi-steady. Figure 3 displays $u/u_{\max}$ at $t = t_{end}$ for the four modeling approaches (top row), along with a magnified view (middle row). The absolute residual $|u-u^{\mathrm{DNS}}|$ is presented in the bottom row. The MARL policy produces significant numerical oscillations, similar to DSM, and results in shock smoothing. Consequently, residuals are elevated at the shock location. The OptiMARL policy yields a significantly sharper solution compared with all other approaches, as evidenced by the reduced residuals at the shock location. While numerical oscillations are present, they are considerably less pronounced than those observed with the MARL or DSM approach.

Figure 4 presents a comparative analysis of the spatial spectra of $u(x, t_{end})$ as a function of the normalized wavenumber k/k_{max}. The MARL policy exhibits the poorest performance, demonstrating significant dissipation at high wavenumbers. The OptiMARL policy shows a slight energy buildup at the highest wavenumbers but performs extremely well overall in the high-wavenumber range. The results are virtually indistinguishable from DNS up to $k/k_{\max} = 0.7$, confirming its superior performance relative to the other modeling approaches.

Given that the results presented thus far have been normalized by $u_{\max}$, it is instructive to examine the temporal decay of $u_{\max}$ in Figure 5, which illustrates the dissipation process from large to small scales. The MARL policy and the DSM exhibit higher-than-expected values, partly attributable to the numerical oscillations observed in Figure 3. While TENO5 produces results closely matching those of DNS, the OptiMARL policy decay is indistinguishable from the reference solution. Combined with the spectra shown in Figure 4, these results suggest that the overall energy cascade is accurately captured.

The improved agreement observed with OptiMARL versus MARL can be understood by examining the spatial variations of C_s, as displayed in Figure 6. Due to the fixed spatial locations of the agents and the linear interpolation between them, MARL cannot assign C_s values at the shock location that differ significantly from the remainder of the domain. An exception would occur if an agent were positioned at the shock location, but that is an unlikely scenario. Similarly, because a spatial averaging operator over

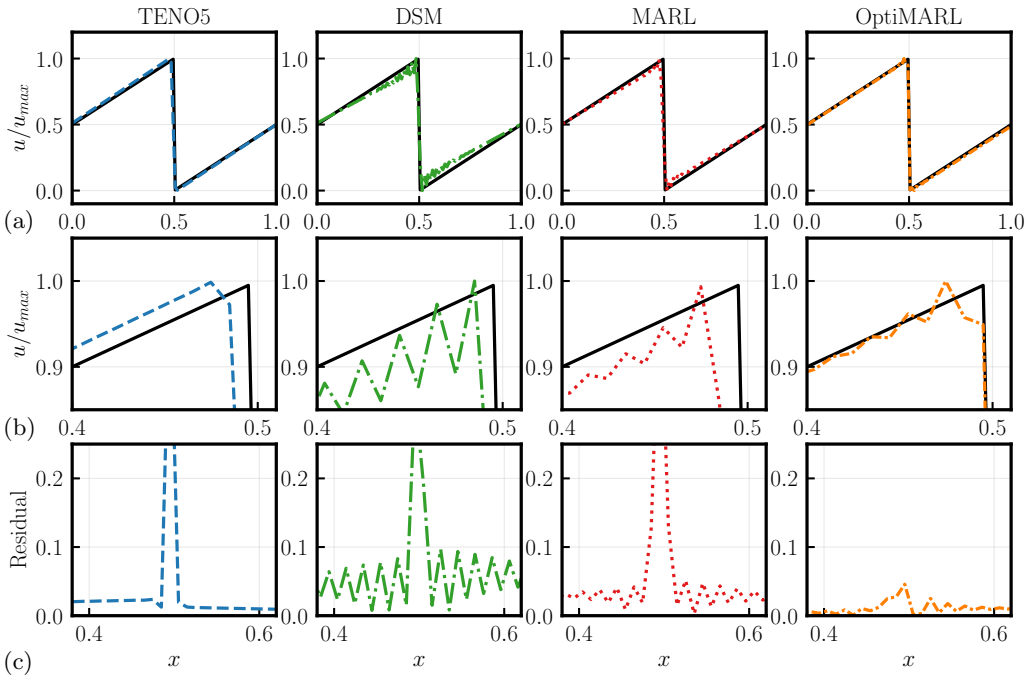

FIGURE 3. DNS (solid black lines) and LES (noncontinuous color lines) for (a) $u(x, t_{end})$. (b) Magnified view. (c) Residual compared with the DNS solution.

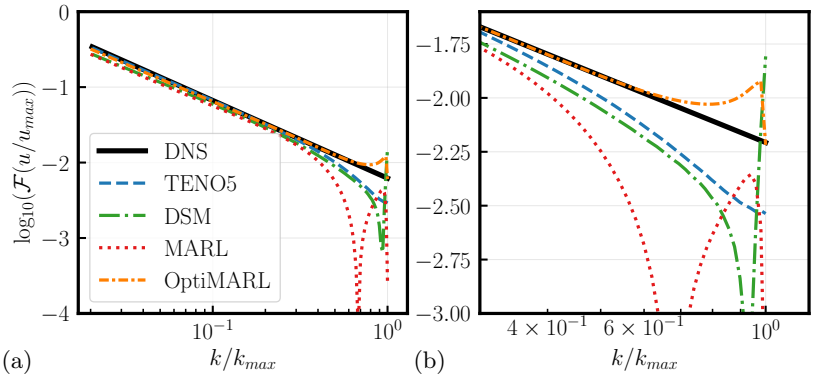

FIGURE 4. (a) DNS (solid black line) and LES (noncontinuous color lines) spectra of $u(x, t_{end})$. (b) Magnified view.

the full domain is employed, C_s in the DSM approach is temporally but not spatially varying. In contrast, OptiMARL assigns substantially lower values to the shock region ($C_s < 0.5$) compared with the remainder of the domain ($C_s \approx 1.5$). These reduced C_s values account for the sharper shock profile observed in Figure 3. Figure 6 also illustrates the clear advantage of the adaptive agent positioning and interpolation within the LS; the C_s value depends solely on the local Ψ state, completely eliminating the spatial dependency that is detrimental to shocklike profiles.

To better understand the spatial variations of C_s assigned by the OptiMARL policy, Figure 7 shows the agent positions (hollow squares) and LES data points (filled circles)

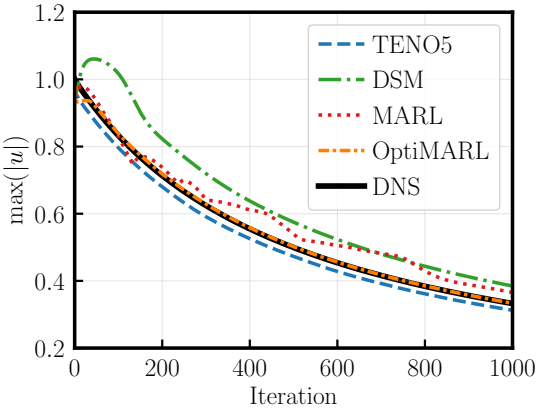

FIGURE 5. Temporal evolution of the maximum absolute value of u.

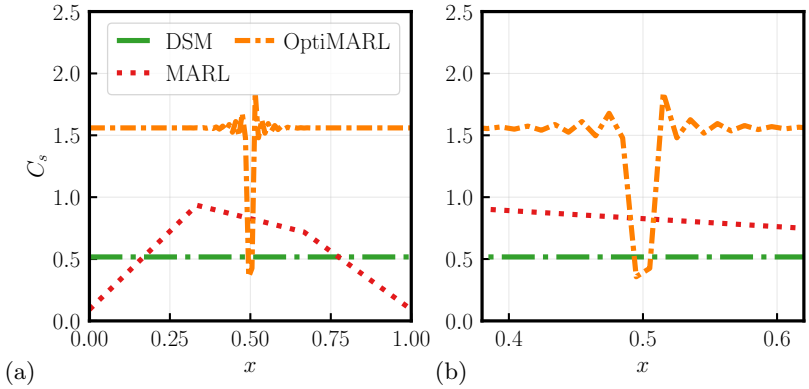

FIGURE 6. (a) Smagorinsky constant at $t = t_{end}$ with the DSM approach, MARL policy, and OptiMARL policy. (b) Magnified view.

within Ψ space. The color coding represents the assigned C_s values. As described in Section 3.4, the LS basis vectors are normalized to range from zero to one. Since two agents per dimension are employed in the current configuration, the agents are positioned at the Ψ extrema. With the presently identified policy, the first LS dimension s_0 corresponds to $\partial u/\partial x$, while the second dimension s_1 represents $\partial^2 u/\partial x^2$. Shock locations are characterized by the minimum value of $\partial u/\partial x$ (i.e., $s_0 = 0$), with only two data points present in the simulation. These points are positioned almost exactly at the locations of the two leftmost agents, enabling precise local C_s adjustment at the shock, as shown in Figure 6. Most of the data points are clustered near $s_0 = 1$, but the substantially different C_s values at the two rightmost agents introduce a dependency on s_1. Overall, the LS-based agent control enabled by OptiMARL leads to a more physically consistent assignment of the Smagorinsky constant based on local flow states, demonstrating the potential for SGS model development in compressible and multiphase flows.

5. Conclusions

This study presents OptiMARL, a novel MARL framework that employs adaptive LS control for SGS modeling in compressible flows. The key innovation lies in using a

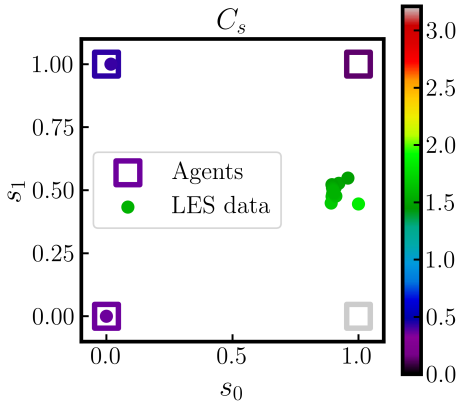

FIGURE 7. Representation of the agent locations and LES data in LS.

dedicated RL agent to identify optimal state-space projections, allowing turbulence closure agents to operate in a dynamically selected subspace rather than at fixed physical locations. The application to the 1D Burgers equation demonstrates that OptiMARL significantly outperforms both traditional approaches (DSM, TENO5) and standard MARL. The framework achieves near-DNS accuracy by automatically identifying the first and second spatial derivatives of the solution to constitute the optimal LS for this problem. This enables precise localization of the Smagorinsky constant at shock locations, resulting in sharper shock profiles and reduced numerical oscillations. The results demonstrate that OptiMARL successfully addresses some limitations of current MARL approaches and establishes its potential for turbulence modeling in flows with localized features. Future work should extend this methodology to 3D compressible turbulence or multiphase flows and investigate the transferability of learned policies across different flow configurations.

REFERENCES

BAE, J. & KOUMOUTSAKOS, P. 2022 Scientific multi-agent reinforcement learning for wall-models of turbulent flows. *Nat. Commun.* **13**, 1443.

BECK, A. & KURZ, M. 2023 Toward discretization-consistent closure schemes for large eddy simulation using reinforcement learning. *Phys. Fluids* **35**, 125122.

DOLAPTCHIEV, S., ACHATZ, U. & TIMOFEYEV, I. 2013 Stochastic closure for local averages in the finite-difference discretization of the forced Burgers equation. *Theor. Comp. Fluid Dyn.* **27**, 297–317.

FU, L., HU, X. & ADAMS, N. 2016 A family of high-order targeted ENO schemes for compressible-fluid simulations. *J. Comput. Phys.* **305**, 333–359.

FUCHS, L., VON SALDERN, J., KAISER, T. & OBERLEITHNER, K. 2024 Deep reinforcement learning-augmented Spalart–Allmaras turbulence model: application to a turbulent round jet flow. *Fluids* **9**, 88.

GERMANO, M., PIOMELLI, U., MOIN, P. & CABOT, W. H. 1991 A dynamic subgrid-scale eddy viscosity model. *Phys. Fluids A-Fluid* **3**, 1760–1765.

KIM, J., KIM, H., KIM, J. & LEE, C. 2022 Deep reinforcement learning for large-eddy simulation modeling in wall-bounded turbulence. *Phys. Fluids* **34**, 105132.

KIRKPATRICK, M., ACKERMAN, A., STEVENS, D. & MANSOUR, N. 2006 On the ap-

plication of the dynamic Smagorinsky model to large-eddy simulations of the cloud-topped atmospheric boundary layer. *J. Atmos. Sci.* **63**, 526–546.

Kurz, M., Offenhaüser, P. & Beck, A. 2023 Deep reinforcement learning for turbulence modeling in large eddy simulations. *Int. J. Heat Fluid Fl.* **99**, 109094.

Love, M. D. 1980 Subgrid modelling studies with Burgers' equation. *J. Fluid Mech.* **100**, 87–110.

Maulik, R. & San, O. 2018 Explicit and implicit LES closures for Burgers turbulence. *J. Comput. Appl. Math.* **327**, 12–40.

Novati, G., de Laroussilhe, H. & Koumoutsakos, P. 2021 Automating turbulence modelling by multi-agent reinforcement learning. *Nat. Mach. Intell.* **3**, 87–96.

Smagorinsky, J. 1963 General circulation experiments with the primitive equations: I. the basic experiment. *Mon. Weather Rev.* **91**, 99–164.

Subel, A., Chattopadhyay, A., Guan, Y. & Hassanzadeh, P. 2021 Data-driven subgrid-scale modeling of forced Burgers turbulence using deep learning with generalization to higher Reynolds numbers via transfer learning. *Phys. Fluids* **33**, 031702.

Tang, H., Rabault, J., Kuhnle, A., Wang, Y. & Wang, T. 2020 Robust active flow control over a range of Reynolds numbers using an artificial neural network trained through deep reinforcement learning. *Phys. Fluids* **32**, 053605.

Vadrot, A., Yang, X., Bae, J. & Abkar, M. 2023 Log-law recovery through reinforcement-learning wall model for large eddy simulation. *Phys. Fluids* **35**, 055122.

Zhou, D., Whitmore, M., Griffin, K. & Bae, J. 2023 Large-eddy simulation of flow over Boeing Gaussian bump using multi-agent reinforcement learning wall model. *AIAA Paper 2023-3985* .

Appendix A. Detailed description of the OptiMARL algorithm

Algorithm 1 OptiMARL latent space (LS) control algorithm

// Inputs
Number of agents N_{agents}
LS dimension N_{LS}
// Initialization
Compute LS agent locations $\{x_{\text{LS},i}\}_{i=1}^{N_{\text{agents}}}$ on equidistant grid in $[0,1]^{N_{\text{LS}}}$
Set $t = 0$
// Main loop
while $t < t_{end}$ **do**
  // Action generation
  for $i = 0$ to N_{agents} **do**
    Compute continuous action $a_1(x_{\text{LS},i}, t)$
  end for
  Compute discrete action $a_2(t)$
  // LS update
  Compute LS indices: $k_{\text{LS}}(t) \leftarrow f(a_2(t))$
  Compute local states $\{s_k(x,t)\}_{k=1}^{N_s}$
  Compute the LS field $\Psi(t)$:
    $\Psi(t) = [(s_k - \min(s_k))/(\max(s_k) - \min(s_k)) : k \in k_{\text{LS}}(t)] : \mathbb{R}^{N_s} \rightarrow \mathbb{R}^{N_{LS}}$
  // Smagorinsky constant computation
  Linearly interpolate in Ψ-space $a_1(x_{\text{LS},i}, t)$ to obtain $C_s(\Psi)$
  Map back $C_s(\Psi)$ to physical space $C_s(x,t)$
  // Agent spatial location update
  for $i = 0$ to N_{agents} **do**
    $x_i = \arg\min_x \|\Psi(t) - x_{\text{LS},i}\|_2$
  end for
  // Reward and feedback
  Compute reward $r(t)$
  Send $(s_k(x_i,t), r(t))$ to agents
  Send $r(t)$ to discrete agent
  // Time advancement
  Advance solution in time
  Update time: $t \leftarrow t + \Delta t$
end while=0