

A control-theoretic framework on moving-mesh methods for solving PDEs

Milad Banitalebi Dehkordi^a ,^{*} Manas Mejari^a, Hassaan Idrees^b, Marco Paggi^b ,
Dario Piga^a

^a Dalle Molle Institute for Artificial Intelligence (IDSIA USI-SUPSI), Via la Santa 1 CH-6962 Lugano-Viganello, Switzerland

^b IMT School for Advanced Studies Lucca, Piazza S.Francesco, 19, 55100 Lucca LU, Italy

ARTICLE INFO

Keywords:

Partial Differential Equations

r-adaptive mesh

MMPDE

Feedback control

Internal model principle

ABSTRACT

Choosing a good mesh is a central challenge in solving partial differential equations (PDEs), particularly when the solution develops sharp features that move over time. The Moving Mesh PDE (MMPDE) is the standard *r*-adaptive method that relocates a fixed set of mesh nodes to track these features, yet it has been studied almost exclusively through numerical analysis. We show that the MMPDE is, in control-theoretic terms, a proportional (P) feedback controller. This reading reframes mesh adaptation as a control-design problem, and it reveals an inherent limitation of the MMPDE. For PDEs whose solutions develop moving features, such as travelling shocks, fronts, and vortices, a proportional controller cannot keep the mesh matched to the feature. To resolve this, we propose *mesh controllers*, procedural moving-mesh strategies in which (i) the controller structure is set by the Internal Model Principle from the PDE feature motion, (ii) the controller gains are set adaptively when the feature motion varies in time, and (iii) a feedforward term is added when the feature's transport velocity is known from prior physical knowledge of the problem. We develop two instances of mesh controllers, a PI-MMPDE and an Adaptive-PI MMPDE, and test them on two benchmark PDEs: the 2D Burgers and LeVeque flows, where the mesh controllers show substantial improvement over the conventional MMPDE baselines.

1. Introduction

Solving a time-dependent PDE numerically requires representing its solution on a finite set of nodes, a *mesh*, whose placement largely determines the solution accuracy. Sharp solution features, such as steep gradients and moving fronts, demand fine local resolution, whereas smooth regions require far fewer nodes. When these features move, a fixed mesh misallocates resolution. Therefore, it becomes advantageous to relocate the nodes and adapt the mesh as the solution evolves.

This is the aim of *r*-adaptive methods, which reposition a fixed set of nodes to track the solution features, without altering the mesh topology, yielding a favourable cost–accuracy trade-off for travelling shocks, boundary layers, and interfaces [1]. Their standard realisation is the Moving Mesh PDE (MMPDE) [2], in which a relaxation ODE drives the nodes towards *equidistribution*, the state in which each node carries an equal share of the solution variation, as measured by a monitor function *M*. The stability and convergence of this ODE are by now well understood, though studied almost entirely from the numerical-analysis side [1].

From a control-theoretical perspective, the relaxation ODE can be viewed as a proportional (P) feedback controller. This connection has a temporal precedent in the work of Söderlind [3], who linked adaptive time-stepping to PID control. Our work

^{*} Corresponding author.

E-mail address: milad.banitalebi@supsi.ch (M.B. Dehkordi).

develops an analogous control-theoretic perspective for spatial mesh adaptation. We show that reading the MMPDE as a P-controller exposes a limitation that the numerical-analysis view leaves implicit, namely, a proportional controller cannot track a moving reference without a persistent error. Different classes of PDEs exhibit diverse types of feature motion, from stationary layers to translating fronts and oscillating vortices. In each case the optimal target mesh (the one that equidistributes the current solution) moves with the feature, so the MMPDE, being a P-controller, lags behind.

The control-theoretic perspective provides a systematic way to understand and address this limitation by utilising the *Internal Model Principle* (IMP) [4] to determine the minimum controller structure needed for each type of feature motion (Section 3). Our contribution is not the introduction of new control algorithms, but rather showing how established control-theoretic principles can provide new insights into moving-mesh methods for PDEs and guide their systematic improvement. Applying the IMP, we derive control-guided mesh-motion policies, which we term *mesh controllers*. We further develop two extensions that broaden this class: an adaptive variant for features whose motion changes over time (Section 4.2), and a feedforward variant for problems with a known transport velocity (Section 4.3). Together, these elements define a systematic design procedure (Section 4.4) that connects the dynamics of PDE solution features to appropriate mesh-motion laws.

We develop two mesh controllers: the PI-MMPDE and the Adaptive-PI MMPDE and compare them against uniform and P-MMPDE meshes on the 2D Burgers and LeVeque flows (Section 5).

Notation. ∇ , $\nabla \cdot$, and ∇^2 denote the gradient, divergence, and Laplacian, respectively; a subscript ∇_{ξ_i} indicates gradient taken with respect to the i coordinate of a vector ξ . For vectors $a, b \in \mathbb{R}^d$, $\langle a, b \rangle := a^\top b$ is the Euclidean inner product and $\|a\|_2 := \langle a, a \rangle^{1/2}$ denotes the 2-norm. For $x \in \mathbb{R}^d$ and bounds $\ell, u \in \mathbb{R}^d$ with $\ell \leq u$ elementwise, $\text{clip}(x, \ell, u)$ denotes the clip (saturation) function applied componentwise, namely, it projects x onto the box $[\ell_1, u_1] \times \cdots \times [\ell_d, u_d]$, with $\text{clip}(x, \ell, u)_i := \min\{\max\{x_i, \ell_i\}, u_i\}$.

2. MMPDE as a proportional controller

2.1. Setup

Consider a time-dependent PDE whose solution $\mathbf{u} : \Omega \times [0, T] \rightarrow \mathbb{R}^m$ evolves on a spatial domain $\Omega \subset \mathbb{R}^d$. Numerical methods solve such a PDE by discretising it into a system of ODEs at selected points, the *nodes*, of the spatial domain, whose collection forms a computational mesh. The accuracy of the solution depends on where these nodes are placed at each time instant [1]. Adaptive methods such as MMPDE exploit this by relocating the nodes to *optimal* positions as the solution evolves.

In MMPDE, the mesh position $\mathbf{x} \in \Omega$ is augmented as an additional PDE variable alongside $\mathbf{u}$. In particular, the augmented $(\mathbf{u}, \mathbf{x})$ system is solved on a fixed computational grid $\Omega_c \subset \mathbb{R}^d$ having *fixed* reference coordinates ξ . The mesh position $\mathbf{x}$ can then be viewed as a map $\mathbf{x} : \Omega_c \rightarrow \Omega$ that sends each reference coordinate ξ to its physical position $\mathbf{x}(\xi, t)$ at each time instance t .

We now specify what makes a mesh *optimal*. The desired mesh resolution is encoded by a symmetric positive-definite monitor tensor $M(\mathbf{u}, \mathbf{x}) \in \mathbb{R}^{d \times d}$, i.e., $M(\cdot, \cdot)$ marks regions where the nodes should cluster, e.g., steep gradients, shocks, boundary layers etc. The departure from an optimally distributed mesh is measured by the *equidistribution* operator [5],

$$\mathcal{L}_M[\mathbf{x}] := \nabla_\xi \cdot (M(\mathbf{u}, \mathbf{x}) \nabla_\xi \mathbf{x}), \quad (1)$$

and the mesh is *equidistributed* when this residual vanishes,

$$\mathcal{L}_M[\mathbf{x}] = \mathbf{0}. \quad (2)$$

Any nonzero $\mathcal{L}_M[\mathbf{x}]$ means the nodes are mispositioned relative to the resolution demanded by the monitor $M(\mathbf{u}, \mathbf{x})$.

The MMPDE prescribes the mesh dynamics by the relaxation ODE

$$\tau \dot{\mathbf{x}} = \mathcal{L}_M[\mathbf{x}], \quad (3)$$

where $\tau > 0$ is a scalar time constant setting the timescale of adaptation; smaller τ gives faster relaxation towards equidistribution. Solving the augmented $(\mathbf{u}, \mathbf{x})$ PDE system amounts to discretising (3), along with the original PDE, at N nodes $\{\xi_k\}_{k=1}^N$ on the fixed grid Ω_c , in the reference coordinate ξ .

2.2. Gradient-flow structure and P-controller interpretation

In this section we show that, under the frozen-coefficient convention [1], the MMPDE (3) has a gradient-flow structure with respect to a naturally defined equidistribution energy $E[\mathbf{x}]$, where E serves as a Lyapunov function and decays monotonically along all mesh trajectories. Equivalently, the MMPDE admits a control-theoretic interpretation as a single-integrator plant driven by a proportional controller with gain $K_p = 1/\tau$, where the equidistribution residual $\mathbf{e} := \mathcal{L}_M[\mathbf{x}]$ plays the role of the error signal. Node positions thus move towards the equilibrium $\mathcal{L}_M[\mathbf{x}] = \mathbf{0}$, i.e., the equidistributed mesh.

The result is stated formally as follows:

Assumption 1 (Frozen-coefficient convention). The monitor is evaluated at the current mesh, $M := M(\mathbf{u}, \mathbf{x})$, and is held fixed for the perturbation $\mathbf{x} \mapsto \mathbf{x} + \delta \mathbf{x}$.

Proposition 1. The MMPDE $\tau \dot{\mathbf{x}} = \mathcal{L}_M[\mathbf{x}]$ admits a control-theoretic interpretation as the single-integrator plant $\dot{\mathbf{x}} = u$ driven by the proportional feedback law

$$u = K_P \mathbf{e}, \quad K_P := \frac{1}{\tau}, \quad \mathbf{e} := \mathcal{L}_M[\mathbf{x}], \quad (4)$$

where $\mathbf{e}$ is the equidistribution residual. Under [Assumption 1](#), the closed loop is a gradient flow on the equidistribution energy

$$E[\mathbf{x}] := \frac{1}{2} \int_{\Omega_c} \sum_{k=1}^d \mathbf{x}_{\xi_k}^\top M \mathbf{x}_{\xi_k} d\xi, \quad \mathbf{x}_{\xi_k} := \frac{\partial \mathbf{x}}{\partial \xi_k}, \quad (5)$$

where ξ_k is the k th component of ξ and E is a Lyapunov function i.e., $\dot{E} \leq 0$ along all trajectories, with $\dot{E} = 0$ if and only if $\mathbf{e} = \mathbf{0}$.

Proof. Substituting $K_P = 1/\tau$ and $\mathbf{e} = \mathcal{L}_M[\mathbf{x}]$ into $\tau \dot{\mathbf{x}} = \mathcal{L}_M[\mathbf{x}]$ gives $\dot{\mathbf{x}} = K_P \mathbf{e}$, the plant $\dot{\mathbf{x}} = u$ under the proportional law $u = K_P \mathbf{e}$. It remains to establish the gradient-flow structure.

We perturb the mesh map, $\mathbf{x} \mapsto \mathbf{x} + \delta \mathbf{x}$, with M held fixed by [Assumption 1](#). Since E is quadratic in the derivatives $\mathbf{x}_{\xi_k}$ and $M = M^\top$, its first variation is

$$\delta E = \int_{\Omega_c} \sum_{k=1}^d \mathbf{x}_{\xi_k}^\top M \delta \mathbf{x}_{\xi_k} d\xi, \quad \delta \mathbf{x}_{\xi_k} := \frac{\partial(\delta \mathbf{x})}{\partial \xi_k}. \quad (6)$$

Integrating by parts in ξ_k ,

$$\delta E = \int_{\partial \Omega_c} \sum_{k=1}^d (M \mathbf{x}_{\xi_k})^\top \delta \mathbf{x} n_k dS - \int_{\Omega_c} \sum_{k=1}^d \left(\frac{\partial}{\partial \xi_k} (M \mathbf{x}_{\xi_k}) \right)^\top \delta \mathbf{x} d\xi. \quad (7)$$

where dS is the boundary measure on $\partial \Omega_c$ and n_k the k th component of the outward normal. The boundary nodes are fixed, so $\delta \mathbf{x} = \mathbf{0}$ on $\partial \Omega_c$ and the boundary integral vanishes. The remaining sum is the divergence $\sum_k \partial_{\xi_k} (M \mathbf{x}_{\xi_k}) = \nabla_\xi \cdot (M \nabla_\xi \mathbf{x}) = \mathcal{L}_M[\mathbf{x}]$, so

$$\delta E = - \int_{\Omega_c} \mathcal{L}_M^\top[\mathbf{x}] \delta \mathbf{x} d\xi, \quad \frac{\delta E}{\delta \mathbf{x}} = -\mathcal{L}_M[\mathbf{x}] = -\mathbf{e}. \quad (8)$$

The residual is therefore the negative functional gradient of E , so the MMPDE is gradient descent, $\dot{\mathbf{x}} = -\frac{1}{\tau} \delta E / \delta \mathbf{x} = K_P \mathbf{e}$. Differentiating E along the trajectories,

$$\dot{E} = \int_{\Omega_c} \left(\frac{\delta E}{\delta \mathbf{x}} \right)^\top \dot{\mathbf{x}} d\xi = -K_P \int_{\Omega_c} \|\mathbf{e}\|_2^2 d\xi \leq 0, \quad (9)$$

with equality if and only if $\mathbf{e} = \mathbf{0}$, i.e. the equidistribution condition [\(2\)](#). Hence E is a Lyapunov function, and it decreases strictly until the mesh equidistributes. $\square$

[Proposition 1](#) identifies the MMPDE as a proportional controller with no integral action.

Remark 1. Note that the MMPDE numerical scheme follows the same frozen-coefficient convention. At each time step, the monitor is evaluated at the current mesh, the mesh is then adapted with M held fixed, and the PDE is advanced one step. The scheme and [Proposition 1](#) therefore treat the same operator.

2.3. The steady-state lag

The absence of integral action has an immediate consequence for PDEs whose solution feature moves. At each instant, there is an optimal mesh $\mathbf{x}^*(t)$, such that

$$\mathcal{L}_M[\mathbf{x}^*(t)] = \mathbf{0}. \quad (10)$$

Note that $\mathbf{x}^*(t)$ is the target reference signal which the controller tracks. The P-law $\dot{\mathbf{x}} = K_P \mathbf{e}$ drives the mesh towards it, acting whenever $\mathbf{e} \neq \mathbf{0}$ and stopping at equidistribution. For a stationary feature $\mathbf{x}^*$ is time-invariant, so the mesh reaches it and the residual vanishes.

When the feature translates with velocity $\mathbf{v} := \dot{\mathbf{x}}^*(t)$, tracking requires the mesh to move at the same velocity, $\dot{\mathbf{x}} = \mathbf{v}$. Under the P-law $\dot{\mathbf{x}} = K_P \mathbf{e}$, this is possible only with a persistent nonzero residual

$$\mathbf{e}_\infty = \frac{\mathbf{v}}{K_P} = \tau \mathbf{v}. \quad (11)$$

The mesh therefore cannot equidistribute while tracking a moving feature. Instead, it settles at a fixed offset whose residual $\mathbf{e}_\infty$ is proportional to the feature speed $\|\mathbf{v}\|$ and the relaxation time τ .

Note that this lag is structural. No finite gain $K_P = 1/\tau$ removes it. Moreover, since $\mathbf{e}_\infty$ is independent of the node count, mesh refinement by increasing N cannot eliminate it. The offset is largest where the mesh responds most slowly to the residual. Eliminating it requires an integrator in the loop, which is the subject of the internal model principle discussed in the next section.

Table 1
Minimum mesh controller structure for each Feature motion class.

Feature motion	Reference $\mathbf{x}^*(t)$	Controller
Stationary	Constant ($p = 0$)	P
Constant speed	Ramp ($p = 1$)	PI
Accelerating	Parabola ($p = 2$)	PII
Oscillating	Periodic	Resonant

3. The internal model principle as a mesh design tool

We showed that a proportional controller leaves a steady-state lag whenever the target mesh reference signal varies over time. The *Internal Model Principle* (IMP) [4] states exactly when such lag can be removed. A controller drives the steady-state error to zero against a class of reference signals if and only if it contains an internal model of those signals. For a polynomial reference of degree p , that model has p integrators. A P-controller has none, so it rejects constant references ($p = 0$) but lags on a ramp ($p = 1$, as in (11)) or any higher-order signal.

3.1. From feature motion to controller

For our mesh controller, the reference signal is the target mesh $\mathbf{x}^*(t)$. Its motion is inherited from the solution feature, since the mesh that resolves a feature must move with it. The feature's kinematics therefore sets the class of the reference, and the controller is then fixed with utilising IMP.

The most common feature motions fall into four such classes. A stationary feature, like a steady boundary layer, keeps $\mathbf{x}^*$ fixed, so the reference is constant and a P-controller suffices. A feature translating at constant speed, like a travelling wave, moves $\mathbf{x}^*$ at constant velocity, which is a ramp reference, so tracking it requires one integrator, in the form of the PI-controller. An accelerating feature, like a steepening front, moves $\mathbf{x}^*$ with growing speed, which gives a parabolic reference and needs two integrators. An oscillating feature, like a rotating vortex, moves $\mathbf{x}^*$ periodically, so the reference is periodic and calls for a resonant controller. Table 1 summarises the feature motions, their corresponding reference signals $\mathbf{x}^*(t)$, and the required matching controller based on IMP.

Remark 2. When a controller matches the dominant feature motion in Table 1, the **steady-state** error is already zero, so adding capability cannot improve it. A higher-order controller then adds cost without improving accuracy. Conversely, an under-powered controller leaves a structural lag that no gain can remove.

For the translating shocks and travelling waves most common in practice, the ramp reference is required. The PI is then the minimum sufficient controller, as with a P-controller no choice of relaxation time τ can replace the missing integrator. In the next section, we detail this PI controller termed as the fixed-gain PI-MMPDE.

We remark that Table 1 considers only one dominant feature class, which real simulations rarely guarantee. A feature may steepen, then translate at near-constant speed, then decelerate, passing through the parabola, ramp, and constant rows in a single run. No fixed-gain controller matches all three phases. This motivates the development of an Adaptive-PI that adapts its gains as the class changes. In the next section, we describe the adaptive-PI-MMPDE.

4. Mesh controllers

4.1. PI-MMPDE

The IMP prescribes one integrator to eliminate the ramp lag (11). Augmenting the P-MMPDE with an integral term of gain K_I gives

$$\dot{\mathbf{x}} = K_P \mathbf{e} + K_I \mathbf{z}, \quad (12a)$$

$$\dot{\mathbf{z}} = \mathbf{e}, \quad (12b)$$

where $\mathbf{z}$ accumulates the equidistribution residual. At steady state $\dot{\mathbf{z}} = \mathbf{e} = \mathbf{0}$ forces the residual to vanish, so the mesh reaches exact equidistribution, and the stored term $\mathbf{z} = \mathbf{v}/K_I$ supplies the constant velocity needed to track a feature moving at $\mathbf{v}$. This is the internal model the IMP requires. With the monitor held fixed, the energy E of Proposition 1 serves as a storage function, making the plant passive from the mesh velocity to the residual. Standard passivity arguments then give asymptotic stability of the PI loop for all $K_P, K_I > 0$ [6,7]. A simple choice of gains is $K_P = 1/\tau_{lb}$ and $K_I = K_P^2/4$, where τ_{lb} is a relaxation time, large enough for the mesh update to integrate stably, and is the only free parameter.

Integral action introduces a failure mode the P-MMPDE does not have. When adjacent nodes approach their minimum allowed spacing, the mesh cannot compress further, so the residual there cannot be driven to zero. The integral state $\mathbf{z}$ nonetheless keeps accumulating this residual. Once the feature moves on and the nodes are free to spread again, the over-large $\mathbf{z}$ drives the mesh past its target before unwinding, a transient overshoot. This is the classic integrator windup problem [8].

Two standard remedies apply to overcome this problem. We employ a leaky integrator that replaces the pure accumulation with $\dot{\mathbf{z}} = \mathbf{e} - \mu\mathbf{z}$, adding a decay term $-\mu\mathbf{z}$ ($\mu > 0$) that drains stored action, so $\mathbf{z}$ cannot grow without bound. This comes at the cost of a small steady-state bias, since the leak also acts when the residual is non-zero. The Clegg reset integrator instead accumulates normally but discharges $\mathbf{z}$ the moment it begins to oppose the current residual, which signals that the mesh has overshot [9,10]. We use the reset, as it removes the wound-up state without the steady-state bias of the leaky integrator, as follows,

$$\dot{\mathbf{z}} = \mathbf{e}, \quad \mathbf{z} \leftarrow \mathbf{0} \quad \text{when} \quad \langle \mathbf{z}, \mathbf{e} \rangle < 0, \quad (13)$$

where $\mathbf{z} \leftarrow \mathbf{0}$ denotes the reset to zero.

4.2. Adaptive-PI-MMPDE

We have noted that a single fixed-gain controller cannot match a feature whose class changes over time. A shock may form, translate, then decelerate, and an oscillating feature reverses the residual each cycle. In these cases the best integral gain is itself time-varying, set by the instantaneous feature speed, which is not known in advance. The Adaptive-PI controller addresses this by adapting two gains online in the PI law (12), while keeping the reset integrator (13) for anti-windup.

Funnel proportional gain. The funnel mechanism replaces the fixed proportional gain with one that drives the error towards a prescribed envelope. A time-varying bound $\varphi(t) > 0$ is set on the equidistribution error $\|\mathbf{e}(t)\|$. As the error nears this bound, the gain grows and pushes the error back down. Gain selection then becomes the choice of a tolerance. Rather than tuning K_p , one specifies the target error bound $\varphi(t)$, and the mechanism supplies the gain that pursues it. Here, the constant gain K_p is replaced by the following barrier gain [11]

$$k(t) = \frac{k_0}{1 - (\|\mathbf{e}(t)\|/\varphi(t))^2}, \quad k_0 = K_p. \quad (14)$$

For a relative-degree-one minimum-phase plant, this construction keeps $\|\mathbf{e}(t)\| < \varphi(t)$ whenever $\|\mathbf{e}(0)\| < \varphi(0)$ [11]; our mesh dynamics meets this condition under the frozen-monitor linearisation. In practice, the gain is saturated to avoid stiffness during rapid shock formation, thus the bound is only prescribed rather than strictly guaranteed.

The bound $\varphi(t)$ decays exponentially from an initial value φ_0 to a floor φ_∞ [12],

$$\varphi(t) = \varphi_\infty + (\varphi_0 - \varphi_\infty) e^{-\lambda_\varphi t}, \quad (15)$$

with rate $\lambda_\varphi = 1/T$ for a simulation horizon T , so the bound tightens gradually over the whole run.

The initial value is set as $\varphi_0 = c_0 \|\mathbf{e}(0)\|$ with $c_0 > 1$, so that $\varphi(0) > \|\mathbf{e}(0)\|$ and the error starts inside the funnel. The floor φ_∞ is set as $\varphi_\infty = \beta \varphi_0$ with $\beta > 0$ a dimensionless tolerance.

Adaptive integral gain. The integral gain $\hat{K}_I$ is adapted online from the alignment between the integral state $\mathbf{z}$ and the current residual $\mathbf{e}$ [13],

$$\hat{K}_I \leftarrow \text{clip} \left(\hat{K}_I + \gamma \Delta t \frac{\langle \mathbf{z}, \mathbf{e} \rangle}{\|\mathbf{z}\| \|\mathbf{e}\| + \epsilon}, \hat{K}_I^{\min}, \hat{K}_I^{\max} \right), \quad (16)$$

applied once per step, with adaptation rate $\gamma > 0$ and a small $\epsilon > 0$ that keeps the update well defined when $\mathbf{z}$ or $\mathbf{e}$ is near zero. The normalised inner product measures the alignment of the accumulated action with the current error. When the two align ($\langle \mathbf{z}, \mathbf{e} \rangle > 0$) the error has persisted in one direction, so the integrator is under-powered and $\hat{K}_I$ is raised. When they oppose ($\langle \mathbf{z}, \mathbf{e} \rangle < 0$) the integrator has overshot, and $\hat{K}_I$ is lowered. The normalisation makes the update independent of the magnitudes of $\mathbf{z}$ and $\mathbf{e}$. The gain starts at the PI value $\hat{K}_I = K_p^2/4$, so the controller matches the PI-MMPDE at $t = 0$, and is confined to the interval $[\hat{K}_I^{\min}, \hat{K}_I^{\max}]$.

The combined Adaptive-PI system is as follows,

$$\dot{\mathbf{x}} = k(t) \mathbf{e} + \hat{K}_I \mathbf{z}, \quad (17a)$$

$$\dot{\mathbf{z}} = \mathbf{e}, \quad \text{with reset (13)}, \quad (17b)$$

with $k(t)$ as in (14) and (15) and $\hat{K}_I$ given in (16). On overshoot, the reset clears the accumulated state $\mathbf{z}$ while the adapted gain $\hat{K}_I$ is retained, so integral action rebuilds from zero at the current integral gain.

4.3. Feedforward for known transport velocity

The controllers so far are purely feedback. The mesh moves only in response to the measured equidistribution residual $\mathbf{e}$. In some applications, the solution feature, for e.g., transport velocity, is known in advance, as in prescribed-flow problems, and then it can be supplied directly rather than inferred through feedback. This is what the arbitrary Lagrangian–Eulerian (ALE) formulation does [14]. It moves the target mesh with the known velocity, which in control terms is a feedforward term added to the mesh dynamics,

$$\dot{\mathbf{x}} = k(t) \mathbf{e} + \hat{K}_I \mathbf{z} + \mathbf{w}, \quad (18)$$

where $\mathbf{w}$ is the transport velocity. The feedback still drives the mesh towards equidistribution from the monitor. The feedforward supplies the known transport, so the feedback only has to correct what the feedforward misses.

Algorithm 1 Mesh Controller design**Input:** Dominant feature motion(s); known transport $\mathbf{w}$ if available**Output:** Mesh controller for the problem

- 1: Map the feature motion to its reference class and base controller via Table 1.
- 2: **if** the feature class changes over time **then** adopt the adaptive form, adapting the gains online as in (17); **else** keep fixed gains.
- 3: **if** the transport velocity $\mathbf{w}$ is known **then** add the feedforward term as in (18).
- 4: Retain the reset (13) for anti-windup.

4.4. Mesh controller design procedure

The two controllers above are instances of a general procedure. For a given problem, the controller is selected by Algorithm 1.

As an example, take a problem with oscillating features under a known transport velocity. The periodic row of Table 1 sets a resonant base controller. The time variation calls for the adaptive form, and the known transport adds feedforward, giving a feedforward adaptive resonant controller. This paper builds and validates the PI and Adaptive-PI cases, but the same procedure applies to the other architectures of Table 1. Our main contribution is the mesh controller design procedure, with PI-MMPDE, adaptive-PI-MMPDE controllers being its special cases.

5. Numerical results

We test the framework on two PDE benchmarks: the 2D viscous Burgers' equation and the LeVeque deformational flow. Both are two-dimensional, with solution $\mathbf{u}$ on $\Omega = [0, 1]^2$ and coordinates $\mathbf{x} = (x_1, x_2)$. Throughout, we use the standard gradient monitor

$$M(\mathbf{u}, \mathbf{x}) = \sqrt{1 + \alpha \|\nabla_{\mathbf{x}} \mathbf{u}\|^2} I_2, \quad (19)$$

where $\alpha > 0$ is a tunable sensitivity parameter, set to $\alpha = 1$ in all experiments. The monitor concentrates nodes where the solution varies sharply [1]. On each benchmark, we compare four mesh strategies: a fixed uniform mesh, the P-MMPDE (current practice), the PI-MMPDE, and the Adaptive-PI-MMPDE. A coarse mesh with $n_x \times n_x$ nodes is denoted as the reduced-order model (ROM). We evaluate the ROM against a mesh with 100×100 nodes, denoting it as the full-order model (FOM). The fixed uniform mesh strategy uses the same $n_x \times n_x$ nodes without adaptation. All controller gains are set analytically as detailed in Section 4. For the Adaptive-PI-MMPDE, funnel parameters are set to $c_0 = 3$, $\beta = 0.05$, and $\gamma = 20000$, shared across both benchmarks. We report the mean relative error between the FOM and ROM solutions,

$$\bar{\varepsilon} = \frac{1}{N_t} \sum_{n=1}^{N_t} \frac{\|\mathbf{u}_{\text{FOM}}(t_n) - \mathbf{u}_{\text{ROM}}(t_n)\|}{\|\mathbf{u}_{\text{FOM}}(t_n)\|} \quad (20)$$

averaged over N_t time steps and 10 random initial conditions per resolution, for varying node sizes $n_x \in \{10, 15, 20, 25, 30\}$. The Python code used to reproduce the results is available online.¹

5.1. Benchmark 1: 2D viscous Burgers' equation

The first benchmark is the 2D viscous Burgers' equation,

$$\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla_{\mathbf{x}}) \mathbf{u} = \nu \nabla_{\mathbf{x}}^2 \mathbf{u}, \quad (21)$$

a standard shock test. Here the inertial term steepens an initially smooth solution into a sharp front, which then translates at constant speed. We take homogeneous Dirichlet conditions and $\nu = 0.005$. Each initial condition is set to be a random superposition of Gaussian bumps. From Table 1, the translating front is a ramp reference, and the PI-MMPDE is the minimum sufficient controller.

Fig. 1 illustrates how the mesh tracks the shock front with $n_x = 20$. The ROM (middle panel) is able to accurately reproduce the speed field $\sqrt{u^2 + v^2}$ of a FOM (having 10^4 nodes) (top panel), with only 400 nodes. This is because those nodes concentrate on the shock and coarsen elsewhere, as the mesh controller dictates.²

Table 2 reports the mean relative errors obtained with the three mesh controllers. All three adaptive strategies improve on the uniform mesh, though that margin decreases with resolution n_x . Note that the PI-MMPDE increasingly outperforms the P-MMPDE as the number of nodes grows, from 0.6 to 14.0 percentage points as n_x grows from 10 to 30. PI-MMPDE removes the ramp lag (11), whereas P-MMPDE fails, and this advantage is clearest where the discretisation error is smallest. The Adaptive-PI matches the performance of the PI-MMPDE to within a percent.

Fig. 2 shows the same effect in the equidistribution energy E of Proposition 1. The time-averaged energy (left) tracks the error trend of Table 2. The PI reduction over P grows with resolution, as a finer mesh exposes the structural ramp lag that discretisation error otherwise masks. The energy history at $n_x = 30$ (right) shows PI holding E below P through the translating phase, consistent with the persistent-residual prediction of (11).

¹ <https://github.com/MiladBnit/Mesh-Controller>

² Animated visualisation of the figure is available in the GitHub repository.

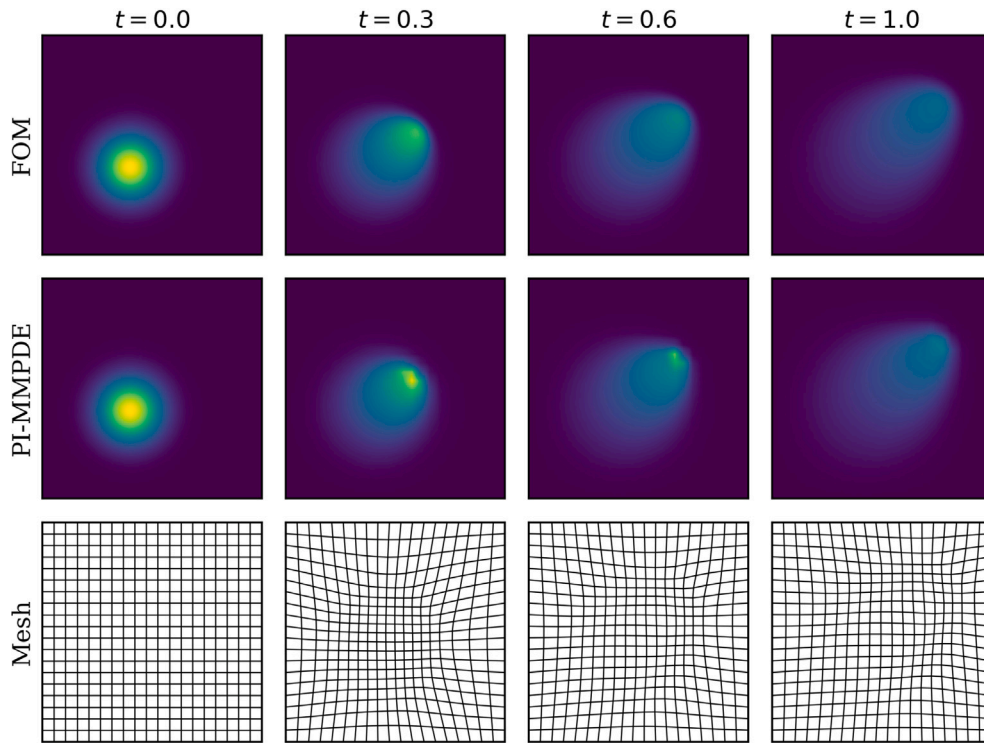


Fig. 1. 2D Burgers benchmark ($n_x = 20$, $\nu = 0.005$, $T = 1$; snapshots at $t \in \{0.0, 0.3, 0.6, 1.0\}$). Top row: FOM speed $\sqrt{u^2 + v^2}$. Middle: PI-MMPDE ROM reconstruction. Bottom: 20×20 adaptive mesh.

Table 2

Monte Carlo results of the 2D viscous Burgers benchmark (Improvement is measured against the uniform mesh).

n_x	$\bar{\varepsilon}$				Improvement (%)	
	Uniform	P	PI	Adp	P	PI
10	0.3182	0.2803	0.2785	0.2786	+11.9	+12.5
15	0.2097	0.1751	0.1705	0.1705	+16.5	+18.7
20	0.1356	0.1158	0.1087	0.1090	+14.6	+19.8
25	0.0895	0.0801	0.0710	0.0719	+10.5	+20.7
30	0.0590	0.0563	0.0481	0.0490	+4.4	+18.4

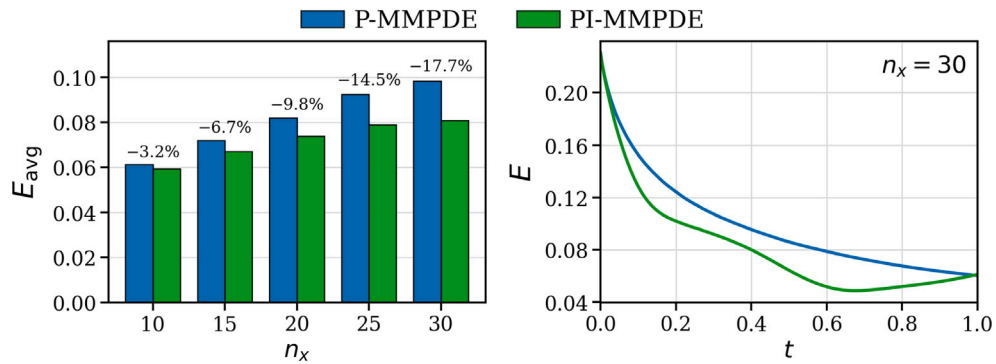


Fig. 2. Equidistribution energy E for the 2D Burgers benchmark, P-MMPDE versus PI-MMPDE. Left: time-averaged energy E_{avg} across resolutions n_x , with the PI reduction over P growing from 3.2% at $n_x = 10$ to 17.7% at $n_x = 30$. Right: energy history at $n_x = 30$, where PI holds E below P through the translating phase.

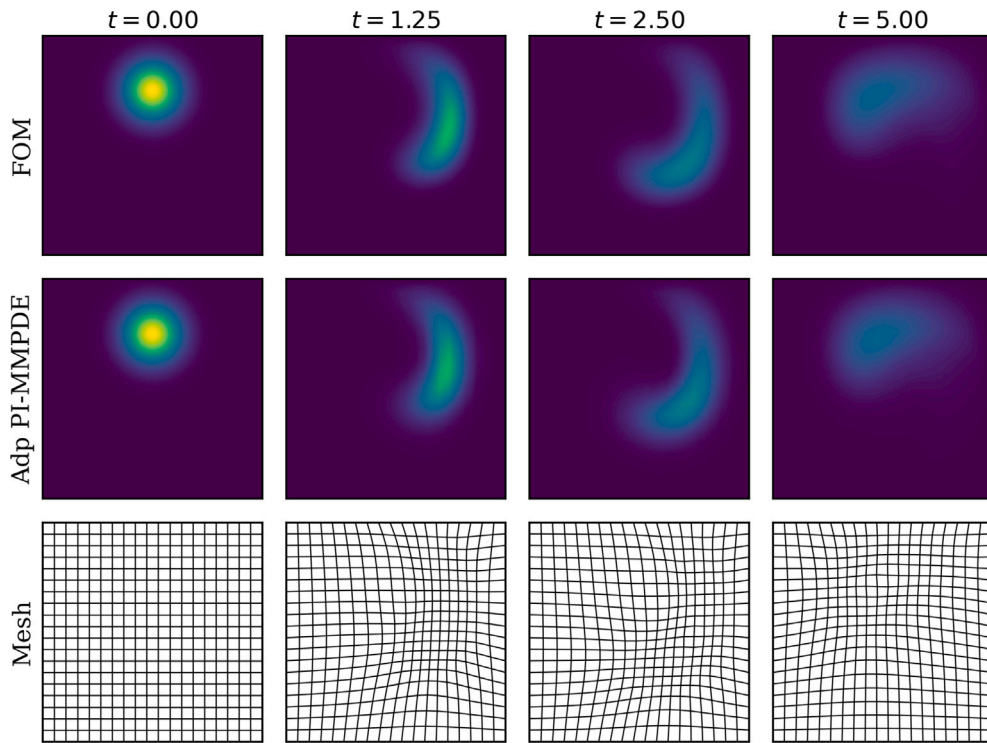


Fig. 3. LeVeque deformational flow benchmark ($n_x = 20$, $\nu = 0.001$, $T = 5$; snapshots at $t \in \{0.0, 1.25, 2.50, 5.00\}$). Top row: FOM speed field. Middle: Adaptive PI-MMPDE ROM. Bottom: 20×20 adaptive mesh.

Table 3

Monte Carlo results of the LeVeque deformational flow benchmark.

n_x	$\bar{\epsilon}$				Improvement (%)	
	Uniform	P	PI	Adp	P	Adp
10	0.3754	0.3052	0.3040	0.3039	+18.7	+19.1
15	0.1993	0.1435	0.1417	0.1414	+28.0	+29.0
20	0.1164	0.0834	0.0812	0.0810	+28.4	+30.5
25	0.0742	0.0558	0.0535	0.0533	+24.8	+28.1
30	0.0503	0.0410	0.0389	0.0389	+18.5	+22.7

5.2. Benchmark 2: LeVeque deformational flow

In the second benchmark, a given velocity $\mathbf{w}$ advects the solution $\mathbf{u}$,

$$\partial_t \mathbf{u} + (\mathbf{w} \cdot \nabla_{\mathbf{x}}) \mathbf{u} = \nu \nabla_{\mathbf{x}}^2 \mathbf{u}, \quad (22)$$

with $\nu = 0.001$. The velocity $\mathbf{w} = (w_1, w_2)$ is the LeVeque deformational flow [15], in coordinates $\mathbf{x} = (x_1, x_2)$,

$$w_1 = -\sin^2(\pi x_1) \sin(2\pi x_2) \cos \frac{\pi t}{T}, \quad (23a)$$

$$w_2 = +\sin(2\pi x_1) \sin^2(\pi x_2) \cos \frac{\pi t}{T}. \quad (23b)$$

with $T = 5$. The flow carries a Gaussian blob (initial condition) into a thin filament at $t = T/2$, then carries it back to its initial position by $t = T$. The feature geometry changes over time, so no fixed K_I is optimal throughout.

Fig. 3 shows the adaptive mesh tracking this motion at $n_x = 20$. It concentrates on the filament at maximum deformation and relaxes as the blob reforms.

Table 3 reports the errors. The Adaptive-PI improves on the P-MMPDE by 0.4 to 4.2 percentage points, mostly through the integral action it shares with the fixed-gain PI. The adaptation adds a modest improvement on top, at negligible cost. The PI and Adaptive-PI schemes require only the extra integral state $\mathbf{z}$ and a few scalar operations per step for the gains, and run in approximately the same time as the P-MMPDE.

6. Conclusions and future works

We have shown that the MMPDE is a proportional feedback controller acting on the equidistribution residual. This reading reframes mesh adaptation as a control-design problem and exposes an inherent insufficiency when applying MMPDE on PDEs with moving features. To resolve this, we propose *mesh controllers* as procedural moving-mesh strategies. For a given PDE, the base structure of the controller (P, PI, and so on) is determined from the feature motion through the IMP. This structure is then refined, with the gains made adaptive where the solution features vary in time, and a feedforward term added when the dominant transport velocity is known.

The most immediate open direction is the extension to unstructured meshes. In such cases, nodes are not confined to a structured grid, and the mesh can tangle as elements invert. Tangling is a hard constraint, and keeping the mesh valid amounts to keeping the element volumes positive. For the standard MMPDE, mesh nonsingularity has been established under a coercivity condition on the meshing functional [16], with the fully discrete result additionally requiring a temporal scheme that preserves monotone energy decrease. Integral action forgoes precisely this monotonicity, since the mesh dynamics are then no longer a gradient flow on E , so the guarantee does not transfer directly to the controllers proposed here. Establishing the analogous condition for mesh controllers, and relating it to the coercivity condition of [16], remains open.

In control terms, non-tangling is a state constraint of the obstacle-avoidance type. Each node must avoid the configurations in which an element degenerates. Control barrier functions provide the standard resolution for such constraints [17,18], encoding the valid configurations as a safe set and filtering the mesh velocity to keep the mesh inside it. The same formulation can enforce further conditions on the shape and quality of each element, so a single barrier could keep the mesh both valid and well-shaped while the controller tracks features aggressively.

A further direction concerns the case of unknown transport velocity. When it is not known in advance, an adaptive feedforward term could infer it from the solution and supply it alongside the equidistribution feedback.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The code required to generate the results are provided in the Github Repo (link provided in the paper).

References

- [1] Huang W, Russell RD. Adaptive moving mesh methods. vol. 174, Springer Science & Business Media; 2010.
- [2] Huang W, Ren Y, Russell RD. Moving mesh partial differential equations (MMPDEs) based on the equidistribution principle. *SIAM J Numer Anal* 1994;31(3):709–30.
- [3] Söderlind G. Automatic control and adaptive time-stepping. *Numer Algorithms* 2002;31(1):281–310.
- [4] Francis BA, Wonham WM. The internal model principle of control theory. *Automatica* 1976;12(5):457–65.
- [5] Winslow AM. Numerical solution of the quasilinear Poisson equation in a nonuniform triangle mesh. *J Comput Phys* 1966;1(2):149–72.
- [6] Van der Schaft A. L2-gain and passivity techniques in nonlinear control. Springer; 2000.
- [7] Ortega R, Loria A, Nicklasson PJ, Sira-Ramirez H. Euler-Lagrange systems. In: Passivity-based control of Euler-Lagrange systems: mechanical, electrical and electromechanical applications. Springer; 1998, p. 15–37.
- [8] Åström K, Hägglund T. Advanced PID control. ISA - The Instrumentation, Systems and Automation Society; 2006.
- [9] Clegg JC. A nonlinear integrator for servomechanisms. *Trans Am Inst Electr Eng Part II: Appl Ind* 1958;77(1):41–2.
- [10] Beker O, Hollot C, Chait Y, Han H. Fundamental properties of reset control systems. *Automatica* 2004;40(6):905–15.
- [11] Ilchmann A, Ryan EP, Townsend P. Tracking with prescribed transient behavior for nonlinear systems of known relative degree. *SIAM J Control Optim* 2007;46(1):210–30.
- [12] Bechlioulis CP, Rovithakis GA. Robust adaptive control of feedback linearizable MIMO nonlinear systems with prescribed performance. *IEEE Trans Autom Control* 2008;53(9):2090–9.
- [13] Ioannou PA, Sun J. Robust adaptive control. Courier Corporation; 2012.
- [14] Donea J, Huerta A, Ponthot J-P, Rodríguez-Ferran A. Arbitrary Lagrangian–Eulerian methods. *Encycl Comput Mech* 2004.
- [15] Leveque RJ. High-resolution conservative algorithms for advection in incompressible flow. *SIAM J Numer Anal* 1996;33(2):627–65.
- [16] Huang W, Kamenski L. On the mesh nonsingularity of the moving mesh PDE method. *Math Comp* 2018;87(312):1887–911.
- [17] Ames AD, Xu X, Grizzle JW, Tabuada P. Control barrier function based quadratic programs for safety critical systems. *IEEE Trans Autom Control* 2017;62(8):3861–76.
- [18] Wang L, Ames AD, Egerstedt M. Safety barrier certificates for collisions-free multirobot systems. *IEEE Trans Robot* 2017;33(3):661–74. <http://dx.doi.org/10.1109/TRO.2017.2659727>.