A Metric Tensor Approach to Data Assimilation on Adaptive Moving Meshes

Cassidy Faye Krause

Submitted to the graduate degree program in Department of Mathematics and the Graduate Faculty of the University of Kansas in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Date defended: January 31, 2022

The Dissertation Committee for Cassidy Faye Krause certifies
that this is the approved version of the following dissertation :

A Metric Tensor Approach to Data Assimilation on Adaptive Moving Meshes

Erik Van Vleck, Chair

Date approved: January 31, 2022

Abstract

Data assimilation (DA) combines noisy, partial data with imperfect physical models in order to make better predictions about the past, current, or future state of some dynamical system. Often the partial differential equations (PDEs) used in these physical models require fine spatial resolution in some areas, and less resolution in others. As such, the physical models benefit from the use of adaptive moving mesh methods, where the location of the spatial grid points can change in time.

The combination of data assimilation procedures with adaptive moving mesh techniques is nontrivial when an ensemble-based DA procedure is employed. Ensemble DA procedures use an ensemble of solutions to the physical model, and the ensemble mean and covariance must be calculated each time an observation is incorporated into the model. Therefore, special care must be taken in the case where each ensemble member is allowed to evolve on its own independent mesh.

This thesis combines ensemble data assimilation techniques with adaptive moving mesh methods through a novel adaptive common mesh. Each ensemble member evolves independently on its own adaptive mesh through the use of a monitor function, or metric tensor. The information from these ensemble metric tensors is used to define an adaptive common mesh at each observation time-step through the metric tensor intersection. Unlike previous works, this method easily generalizes to higher spatial dimensions.

One benefit to this approach is that the common mesh can also be concentrated near observation locations by modifying the monitor function that controls the movement of the adaptive common mesh. This greatly reduces the need to interpolate observation values, which can cause significant errors in the DA predictions. This gives the user a flexible framework where the meshing scheme can either be chosen so that the adaptive mesh sufficiently supports all ensemble members, is concentrated near the observations, or some combination thereof, depending on what is best for their problem.

In addition to the framework for the adaptive common mesh, we also provide a new localization scheme based off the metric tensor intersection. This domain-based localization scheme, through the use of a tuning parameter, gives a wider radius of influence to observations that occur in smooth parts of the numerical solution and a smaller radius of influence to the observations that occur near sharp interfaces or gradients. Through various test problems in one and two spatial dimensions, we show that this localization scheme is competitive with two of the standard domain-based localization schemes widely used.

Though the use of an independent adaptive mesh for each ensemble member can be advantageous, it also can be computationally expensive. Therefore, we also develop a method to have all ensemble members reside on the same mesh, eliminating much of the interpolation. We use a sample of the ensemble members to "look ahead" at each observation step and determine what the next common mesh would be if each of the sample representatives evolved on its own independent mesh. If that new mesh differs significantly from the current mesh, we then use the new mesh for all of the ensemble members. If, on the other hand, it is similar (within some tolerance) to the current mesh, we continue to update via DA and integrate in time on the current mesh.

The efficacy of these methods is demonstrated in both one and two spatial dimensions with the inviscid Burgers equation, in which a steep shock forms and moves across the spatial domain. This is a common test problem for DA with adaptive meshes, because it requires a finer resolution near the shock, and can use a coarser mesh in areas away from the shock.

Finally, we consider a more systematic approach to determining the common mesh. A variational approach can be used to optimize the RMSE, the norm of the innovation, or some other measure of DA success, subject to a linear combination of constraints, such as the equidistribution of arclength or the proximity of common mesh points to the observation locations.

Acknowledgements

I would first like to thank my advisor, Professor Erik Van Vleck, for his continued support and guidance throughout my time at KU. I am grateful that he encouraged me to pursue passions tangential to this thesis through various summer programs, my internship at Sandia National Laboratories, and involvement in the Math Graduate Student Organization, the Association for Women in Math, and various other outreach and teaching activities. This dissertation would not have been possible without his help and encouragement.

I would also like to thank the members of my committee, Professors Weizhang Huang, Xuemin Tu, Weishi Liu, and David Mechem for their insights, thoughtful comments, and engaging discussions. I would especially like to thank Professors Weizhang Huang and David Mechem for taking time to meet with me regularly to discuss the practical implementation and application of these methods. Your perspectives, questions, and advice were incredibly helpful in developing the ideas in this thesis.

The Self Graduate Fellowship played a vital role in my development as a mathematician. The Fellow Development program gave me a plethora of professional skills I would not have otherwise. I would especially like to thank Professors Amy Devitt and Robin Rowland for their impactful communications coaching. I am immensely grateful for Stefani Buchwitz's work in coordinating such a fantastic program, and, of course, I am deeply grateful to the late Al and Lila Self for establishing this fellowship.

I would also like to extend my gratitude to the many members of the broader mathematics community who helped me throughout my graduate experience. Though you may have not impacted my research directly, your influence helped shape me into the mathematician I am today. Specifically, I would like to thank James Swenson, Andrew Steyer, Agnieszka Miedlar, Michelle Wiest, and Jila Niknejad. Your influence extends far beyond the scope of this thesis, and I deeply appreciate your advice, support, and mentorship. I would also like to thank my students - former, current, and future - for reminding me of the excitement and joy of learning mathematics. You inspire me.

To my friends and family - thank you. I would not be here today without your love and support. I would like to thank my parents, Penny, Eric, Dave, and Diane, for instilling in me a love of learning. More importantly, thank you for teaching me compassion and resilience. Thank you, John, for keeping me grounded, and for always believing in me. To the Larson family - Kate, Michael, Jacob, Chelsy, Greta, Perry, and Leah - thank you for taking me in as one of your own. Your friendship means more than I can say.

I wouldn't be here today without every person mentioned above, as well as so many others. I am truly grateful for this experience, and I am excited to begin the next part of my journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Data assimilation (DA) combines noisy data, usually from instrumental uncertainty or issues with scale, with models that are imperfect, due to simplifying assumptions or other inaccuracies, to improve predictions about the past, current, or future state of a system. Typical DA techniques require a prior uncertainty and use the likelihood of observations to calculate the posterior distribution. This Bayesian context provides not only predictions but also quantification of the uncertainty in these predictions. DA originated with numerical weather prediction and is now employed in many scientific and engineering disciplines. With infinite dimensional models such as partial differential equations (PDEs) there are often features that require fine spatial resolution to obtain accurate approximations of model solutions. An alternative to uniform fine meshes is to employ meshes that are coarse in parts of the spatial domain but fine in other parts of the domain. When the regions requiring finer resolution change with time, time-dependent adaptive meshes are advantageous. In the context of data assimilation, employing time-dependent adaptive meshes also has the potential to reduce the dimension needed for good approximation of model solutions.

Combining DA with adaptive moving mesh techniques presents both opportunities and challenges. Many DA techniques employ an ensemble of solutions to make predictions and estimate the uncertainty in the predictions. This introduces a key choice for implementation: either each ensemble solution may evolve on its own independent adaptive mesh, or the ensemble time evolution can happen on a mesh that is common to all ensemble members. In the first case, the mesh for each ensemble member may be tailored to each ensemble solution. However, when incorporating a DA scheme, this presents the challenge of combining ensemble solutions that are supported on different spatial meshes. In the second case, one needs to combine the features of the different ensemble solutions to determine a mesh that provides, on average, good approximation for all ensemble members. In addition, the relative position of the ensemble mesh(es) with respect to potentially time-dependent observation locations may also motivate the choice of ensemble mesh(es).

Our contribution in this paper is to develop a framework and techniques to utilize adaptive meshing techniques for data assimilation with application to PDE models in one and higher space dimensions. This framework allows each ensemble member to evolve on its own independent mesh and presents an adaptive common mesh for the DA update. An adaptive mesh, while not usually uniform in the standard Euclidean metric, can be viewed as uniform in some other metric. This metric is defined by a positive-definite matrix valued monitor function, also called a metric tensor, which controls mesh movement for the ensemble members and is also used to define the new adaptive common mesh. The computation of the metric tensor easily generalizes to higher spatial dimensions. Using an adaptive common mesh based on metric tensors of the ensemble meshes provides a means for determining a mesh that is common to all ensemble members while providing good approximation properties for the ensemble solutions.

The adaptive common mesh is formed through the metric tensor intersection of the metric tensors that monitor the movement of the ensemble meshes. Geometrically, the metric tensor intersection is the same as circumscribing an ellipsoid on an element of two meshes and then finding an ellipsoid that resides in the geometric intersection of the first two ellipsoids. The new element given by that intersecting ellipsoid is the result of the metric tensor intersection. This procedure must be done pairwise, so the resulting ellipsoid is not necessarily maximal. However, a greedy algorithm can be used to find an ordering that seeks to maximize the resulting ellipsoid.

The metric tensor intersection of the ensemble member meshes forms a common mesh that supports all ensemble members with high accuracy. However, the mesh points of this common mesh may not align with the observation locations. If the observation locations do not coincide with nodes in the common mesh, then the location mismatch results in errors from interpolating the observations to the common mesh. This is especially relevant in the case of time-dependent observation locations. With field alignment [24], a variational DA scheme is developed that assimilates both the alignment and amplitude of observations simultaneously. In addition, a two-step approach is developed, where first the locations of the state variables are adjusted to better match the observations. Rather than assume that the observations occur at the same spatial discretization used for the numerical solution, a vector of displacement is employed so that the (interpolated) numerical solution at adjusted nodes is obtained to maximize the posterior distribution of the numerical solution with displacement given the observations. The DA then proceeds with traditional correction based upon the amplitude of the errors of the numerical solution at the adjusted discretization.

The mismatch of observation locations and nodal positions of the common mesh presents another potential opportunity for DA on adaptive moving meshes: the meshes can adapt to concentrate near or align with the observation locations. An observation mesh can be formed by associating a metric tensor with the location of the (potentially time-dependent) observations. Intersecting the common mesh from the ensemble members with this observation mesh provides a new common mesh that is concentrated near observation locations. Interpolation error can have a significant effect on the accuracy of DA schemes, and concentrating the mesh near observation locations reduces the interpolation required, thereby improving the performance of the DA algorithms. Of course, a fixed common mesh can also be used to concentrate the mesh near fixed observation locations, but this approach has the benefit of adapting easily to time-dependent observation locations.

In addition, we develop spatially and temporally dynamic localization schemes, based upon the metric tensor(s) corresponding to the adaptive common mesh. Localization improves DA procedures by ensuring that observations only affect nearby points. Broadly speaking, localization schemes fall into two categories: domain localization and covariance (or R) localization. Domain localization schemes define a spatial radius and use that to define which mesh points are affected by a given observation. Covariance localization schemes use a correlation function to modify the covariance matrix that is used in the DA update, so that the covariance between an observation and the solution values decays to zero as the distance between the observation and the solution values increases.

We develop a domain localization scheme that employs the metric tensor. Employing a fixed, uniform radius of influence for the observations is that the localization scheme may not be effective if there is a steep gradient in the solution. One could predetermine the location of the gradient and adjust the localization scheme accordingly, but if the regions of large gradient are time- dependent, this will usually result in the tuned localization parameter being quite small. However, since the metric tensor provides information about the dynamics of the ensemble solution, it can be used to define an adaptive localization scheme where the localization radius can vary in time and space.

One benefit of using an adaptive moving mesh is that fewer mesh points can be used while still maintaining the same accuracy. Having an adaptive time- dependent common mesh allows for a fewer number of nodes used in the common mesh as compared to a fixed, fine common mesh, increasing the efficiency of the linear algebra, e.g., when updating the mean and covariance with an ensemble Kalman filter. An efficient implementation of the Ensemble Kalman Filter (EnKF) requires $\mathscr{O}((D+N_e)D^2+(M+D)N_e^2)$ flops when $D \ll M$ or more generally, for example when $D \approx M$, $\mathscr{O}((M+D+N_e)N_e^2)$ (see, e.g., [21]) where D is the dimension of the observation space, M is the dimension of the discretized dynamical system, and N_e is the number of ensemble members. In large scale geophysical applications we typically desire $N_e \approx 20$ (in general N_e should be roughly the number of positive and neutral Lyapunov exponents). A reduction in M based upon using fewer mesh points while maintaining or enhancing accuracy results in improved efficiency.

There are several recent works on integrating adaptive spatial meshing techniques with DA, although most of the focus has been on PDE models in one space dimension. This includes methods based on evolving meshes based on the solution of a differential equation, methods in which meshes are updated statically based upon interpolation, and remeshing techniques that add or subtract mesh points as the solution structure changes. In [4], the evolution of meshes was movement of the nodes was determined by the solution of moving mesh differential equations that are coupled to the discretized PDE. The state variables of the PDE were then augmented with the position of the nodes and incorporated into a DA scheme. The test problem consisted of a two- dimensional ice sheet assumed to be radially symmetric; therefore, it reduced to a problem with one spatial dimension. In [2] and [6] common meshes were developed based on a combining through interpolation the ensemble meshes. This allowed for update of mean and covariance for Kalman filter based DA techniques while allowing each ensemble member to evolve on its own independent mesh. That is, at each observational time-step, the ensemble members were interpolated to the common mesh, updated with the DA analysis, and then interpolated back to their respective meshes. Specifically, a uniform, non- conservative mesh was used in [2], with Lagrangian observations in one spatial dimension. Higher spatial dimensions were used in [6], with a fixed common mesh refined near observation locations. [26] uses the same 1D non-conservative adaptive meshing scheme as in [2] and extends this approach through the use of an adaptive common mesh, where, like in [4], the state vector is augmented with the node locations.

The outline of this thesis is as follows. Background of data assimilation and adaptive moving mesh techniques is given in Section 2.1. This includes the framework we develop to include equations describing mesh movement within a DA framework. The development of adaptive meshing techniques for DA is detailed in Section 2.2. Metric tensors are introduced and their connection to non-uniform meshes is discussed. Techniques for combining meshes based on metric tensor intersection and for concentrating ensemble mesh(es) near observation locations are developed. The details of our implementation are in Section 2.3. This includes the discontinuous Galerkin discretization we employ and the specific metric tensor formulation we use to adaptively evolve the ensemble mesh(es). The details of our experimental setup and numerical results for both 1D and 2D inviscid Burgers equations are presented in Section 3.1.

Data assimilation techniques seek to combine models and data to improve predictions and quantify uncertainty typically in a Bayesian context (see, e.g., [5], [18], [25], [27]). Consider a finite dimensional discrete time system for a state vector $u \in \mathbb{R}^M$ ($M > 0$) that evolves based upon

$$u_{n+1} = \Psi(u_n) + \xi_n, \quad \xi_n \sim \mathcal{N}(0, \Sigma), \tag{1.1}$$

where $\Psi(\cdot)$ propagates the state forward in time, n stands for the nth time-step, and ξ_n is assumed to be a normally distributed model error with covariance matrix Σ and mean 0. Equation (1.1) can be used to forecast the state of the dynamical system, but this prediction can often be improved by including the observation $y_{n+1} \in \mathbb{R}^D$ ($D > 0$) given by the data model

$$y_{n+1} = \mathscr{H}(u^t_{n+1}) + \eta_{n+1}, \quad \eta_{n+1} \sim \mathscr{N}(0, R), \tag{1.2}$$

where u^t_{n+1} is the unknown "truth" at time $n+1$, $\mathscr{H} : \mathbb{R}^M \to \mathbb{R}^D$ is the observation operator, and η_{n+1} is assumed to be a normally distributed observation error with covariance matrix R and mean 0. In many applications, $D \ll M$. We wish to determine $\{u_n\}_{n=0}^N$ that satisfies, in some sense, both the physical and data models. Note that implicit in the formulation of the data model (1.2) is that the state variables (u_{n+1}) and data variables (y_{n+1}) are supported on common spatial locations.

Many data assimilation techniques are based upon a Bayesian approach that determine the posterior distribution from the prior distribution and the likelihood. Given an observation y_n at time t_n and a prior estimate $\mathbb{P}(u_n)$ of the state, Bayes' Theorem states that

$$\mathbb{P}(u_n | y_n) \propto \mathbb{P}(y_n | u_n)\mathbb{P}(u_n). \tag{1.3}$$

This procedure extends to the sequential assimilation of observations at multiple times under the assumption that the state is Markovian. Note that since the model noise $\{\xi_j\}_{j=1}^N$ is independent and identically distributed (i.i.d.), the prior can be written as $\mathbb{P}(u_n) = [\prod_{j=1}^n \mathbb{P}(u_j | u_{j-1})]\mathbb{P}(u_0)$, where, e.g., $u_0 \sim \mathscr{N}(u_0^b, P_0^b)$.

For nonlinear models (1.1) available data assimilation techniques include the ensemble Kalman filter (EnKF), particle filters (PF), variational methods such as 4DVar, and hybrid techniques that seek to combine the best features of different types of techniques. Several of these classes of methods are naturally ensemble based (EnKF, PF and their variants, as well as several hybrid methods) while variational methods may employ ensembles of solutions to approximate derivatives or as part of an iterative linear system solver.

Ensemble DA procedures use an ensemble of solutions to make predictions for the physical state via a two-step process. First, the prediction step uses the physical model (1.1) to integrate the ensemble members $\{u_n^{e_i}\}_{i=1}^{N_e}$, where N_e is the number of ensemble members, to make the ensemble forecasts $\{\hat{u}_{n+1}^{e_i}\}_{i=1}^{N_e}$. Second, the analysis step incorporates the observations at time t_{n+1} to adjust the prediction $\{u_{n+1}^{e_i}\}_{i=1}^{N_e}$. EnKF does this through the following sequential process:

$$\text{Prediction:} \quad \begin{cases} \hat{u}_{n+1}^{e_i} &= \Psi(u_n^{e_i}) + \xi_n^{e_i}, \quad e_i = 1,\ldots,N_e \\ \hat{m}_{n+1} &= \frac{1}{N_e}\sum_{i=1}^{N_e} \hat{u}_{n+1}^{e_i}, \\ P_{n+1}^b &= \frac{1}{N_e-1}\sum_{i=1}^{N_e} \left(\hat{u}_{n+1}^{e_i} - \hat{m}_{n+1}\right)\left(\hat{u}_{n+1}^{e_i} - \hat{m}_{n+1}\right)^T, \end{cases} \tag{1.4}$$

$$\text{Analysis:} \quad \begin{cases} \mathbf{K}_{n+1} &= P_{n+1}^b H^T \left(H P_{n+1}^b H^T + R\right)^{-1}, \\ u_{n+1}^{e_i} &= (I - \mathbf{K}_{n+1} H)\,\hat{u}_{n+1}^{e_i} + \mathbf{K}_{n+1} y_{n+1}, \quad e_i = 1,\ldots,N_e \end{cases} \tag{1.5}$$

where I is the identity matrix, H is the linearization of $\mathscr{H}$, and $\mathbf{K}_{n+1}$ is the Kalman gain matrix. The ensemble mean $\hat{m}_{n+1}$ together with the forecast covariance P_{n+1}^b in (1.4) is employed to make predictions with a measure of uncertainty. Again, the implicit assumption here is that all ensemble members reside on the same mesh.

Variational methods such as 4DVar and 3DVar are direct formulations of Bayes' Theorem. The analysis update of a variational DA algorithm can be viewed as the minimizer of a corresponding cost function. For example, when the prior distribution and the observational error model are Gaussian, the cost function is

$$J(u_0) = (u_0 - u_b)^T B^{-1}(u_0 - u_b) + \sum_{n=0}^{N} (y_n - \mathscr{H}(u_n))^T R_n^{-1} (y_n - \mathscr{H}(u_n)), \tag{1.6}$$

where u_n is obtained by the evolution of the model dynamics (1.1) and B is the background error covariance. Note that this cost function is quadratic when both the physical model Ψ and the observation operator $\mathscr{H}$ are linear, so in this case there is a unique optimizer. In the case that either the physical model or observation operator are nonlinear, or if the error distributions are not Gaussian, the posterior distribution may not have a unique optimum. The use of non-Gaussian error distributions results to corresponding terms in the cost function (1.6) which may be approximated at least locally with a quadratic cost function, e.g., with variants such as incremental 4DVar.

Both EnKF and 4DVar are used in large-scale applications. 4DVar provides great flexibility in including terms in the cost function, does not require linear physical model or observation operators, and can make use of sophisticated optimization techniques. EnKF is based upon Gaussian assumptions and parameterizes the prediction and uncertainty in terms of the ensemble mean and sample covariance. Together with local linear approximation of the observation operator, EnKF is a linear solver corresponding to a quadratic cost function. For more on the advantages and disadvantages of the EnKF and 4DVar techniques, see [16, 19].

Hybrid methods seek to combine the advantages of ensemble Kalman filter techniques and variational techniques. An important motivation behind hybrid methods is to incorporate flow-dependent background error covariance matrices (P^b_{n+1}) into a variational setting. A representative hybrid method is the ETKF-4DVar technique in which B in (1.6) is replaced by

$$\tilde{B} = \beta B + (1-\beta) P_b$$

where $0 \leq \beta \leq 1$ is a weighting factor, B is the background error covariance used in 4DVar, and P_b is the error covariance found using the ensemble transform Kalman filter (ETKF), a square root filter similar to the EnKF (1.4), (1.5). For $\beta = 1$ this reduces to the standard 4DVar, while for $\beta = 0$ it is known as 4DVAR-BEN, and for $\beta = 1/2$, the method is known as the ETKF-4DVAR. For further discussion of hybrid methods, see, e.g., [1, 3, 20].

While the previous classes of techniques all rely to some extent on Gaussian assumptions by parameterizing the predicted state and its uncertainty in terms of a mean and covariance, particle filters approximate the posterior distributions in an unstructured way in terms of particles (analogous to ensemble members) and particle weights. For example, the standard or bootstrap particle filter uses the model dynamics (1.1) to make predictions using each particle. The weights $\{w^{e_i}_{n-1}\}_{i=1}^{N_e}$ are updated using the observation y_n. In the bootstrap PF the update is

$$w_n^{e_i} = c\, w_{n-1}^{e_i} \mathbb{P}(y_n | u_n^{e_i}), \tag{1.7}$$

where the likelihood, given Gaussian observational error model, is

$$\mathbb{P}(y_n | u_n) \propto \exp\left[-\frac{1}{2} (y_n - H u_n)^T R^{-1} (y_n - H u_n) \right] \tag{1.8}$$

and c is chosen so that $\sum_{i=1}^{N_e} w_n^{e_i} = 1$. The standard particle filter and many variants suffer from weight degeneracy in which most weights are nearly zero and this decreases the effective number of particles. This leads to the need for a large number of particles (N_e) that increases exponentially with the dimension of the state space (M) and the observation space (D). Several variants including the implicit PF, the equivalent weights PF, and optimal proposal PF have been developed (see, e.g., [27]) to overcome weight degeneracy and the curse of dimensionality in PFs.

Modern DA procedures employ spatial localization and covariance inflation techniques. Localization can be done either through a Schur product (also known as a Hadamard or element-wise product) with the covariance matrix, resulting in observations having little or no effect on distant nodes. It can also be done through domain localization, where the domain is broken into subdomains, and the observations of each subdomain only affects the variables supported within that subdomain. Inflation (either additive or multiplicative) increases the entries of the covariance matrix, preventing degeneracy of the procedure.

The focus of this work is employing adaptive moving meshes with ensemble DA methods. In the following we outline a framework for reducing to the discrete time, finite dimensional DA problem (1.1), (1.2). We start from the simplest cases of discretized ODE and PDE on fixed spatial meshes, and then formulate the DA problem using moving mesh methods in which a differential equation determines the movement of mesh points. We will emphasize the errors that are introduced in these different scenarios.

Chapter 2

Framework for DA on Adaptive Moving Meshes

2.1 DA with Adaptive Moving Meshes

2.1.1 ODEs and Time-Dependent PDEs Discretized on Fixed Mesh

Given an ODE

$$\frac{du}{dt} = F(u,t) \tag{2.1}$$

as the physical model, the use of a time stepping technique and the subsequent addition of additive model error is used to obtain (1.1). Similarly, given a time-dependent PDE, after spatial discretization we obtain a system of ODEs that can then be discretized in time to obtain a discrete time, finite dimensional physical model (1.1) with the addition of model noise.

2.1.2 The Physical PDE Model and Moving Mesh Equations

Consider now the physical model as a time-dependent PDE written abstractly as $\frac{\partial u}{\partial t} = F(u)$ posed on an appropriate function space. PDEs employed in a DA context may be coupled to an equation that evolves the spatial mesh, enabling DA on an adaptive moving mesh. There are two basic approaches to adaptive moving meshes. The first is a rezoning approach, used in [2, 26], which updates the mesh at each time using a given mesh generation and interpolates the PDE solution from the old mesh to the new mesh. The second approach is a quasi-Lagrange approach where the mesh is considered to move continuously with time. In this case, the discretized PDE is supplemented with an advective term to reflect mesh movement. If $u = u(x(t),t)$ satisfies a PDE given by $u_t = F(u)$, then the total derivative $\frac{du}{dt}$ is given by

$$\frac{du}{dt} = \nabla_x u \cdot \frac{dx}{dt} + u_t = \nabla_x u \cdot \frac{dx}{dt} + F(u) \equiv \mathscr{F}(u,x). \tag{2.2}$$

The equation for the mesh movement comes from a variational approach in which a cost function (the meshing functional) is minimized through a gradient flow differential equation:

$$x_t = -\frac{1}{\tau}\nabla_x \mathscr{L}(x,u) \equiv \mathscr{G}(x,u), \tag{2.3}$$

where $\tau \geq 0$ is a user-specified parameter controlling the speed of mesh movement. We note that when $\tau = 0$, (2.3) reduces to an algebraic equation $\mathscr{G}(x,u) = 0$ and when satisfied the mesh instantaneously satisfies a local minimizer of the meshing functional. In practice, first fix x and then update the solution of the PDE by solving (2.2). The solution to (2.2), together with x, gives the mesh velocity defined by (2.3). A new mesh is obtained by integrating (2.3) in time.

In an ensemble-based method, an adaptive mesh PDE discretization can be used for each of the ensemble members. However, the computations of the ensemble mean and covariance only make sense if the values of the ensemble members are taken from the same spatial locations. Therefore, if the ensemble members' meshes can evolve independently via an adaptive moving mesh scheme, special care must be taken to calculate the mean and covariance at each DA step. Previous works have explored two general approaches to this problem. One approach is to interpolate the ensemble solutions to a common mesh at each observational time-step and assimilate the PDE variables only. The common mesh approach is used here, as well as in [2, 6]. Another method is to assimilate both the PDE variables and the common mesh locations, as done in [4].

In [4], they augment the state vector U^n with the mesh points x^n. That is, the new state vector is given by

$$V^n = \begin{bmatrix} U^n \\ x^n \end{bmatrix}. \tag{2.4}$$

The DA update is then performed on this augmented state vector. Note that this approach doubles M in the DA scheme, affecting the computational cost of the DA update. In [2, 26] the mesh movement is based on a Lagrangian flow of form $\frac{dx}{dt} = u$ together with upper and lower bounds on the mesh spacing that are enforced by remeshing and subtracting or adding mesh points. For example, on a spatial domain $\Omega = (0,1)$ this corresponds to a meshing functional (2.3) of the form

$$\mathscr{L}(x,u) = -\int_0^x u(y,t)dy. \tag{2.5}$$

Example 1. *As a concrete example of a PDE and mesh movement equation, consider a 1D reaction diffusion equation:*

$$u_t - u_{xx} = f(u), \quad 0 < x < 1, \quad t > 0 \tag{2.6}$$

with, for example, homogeneous Dirichlet or homogeneous Neumann boundary conditions and smooth initial data.

A non-uniform finite difference discretization in space yields $(U_j \equiv U_j(t) \approx u(x_j,t))$*:*

$$\dot{U}_j \;=\; \left[\frac{U_{j+1}-U_{j-1}}{x_{j+1}-x_{j-1}}\right]\dot{x}_j + \frac{2}{x_{j+1}-x_{j-1}}\left[\frac{U_{j-1}-U_j}{x_j-x_{j-1}} + \frac{U_{j+1}-U_j}{x_{j+1}-x_j}\right] + f(U_j). \tag{2.7}$$

Now consider a moving mesh scheme determined by the equidistribution of arc length of the solution:

$$\tau\dot{x}_j = \sqrt{(x_{j+1}-x_j)^2 + (U_{j+1}-U_j)^2} - \sqrt{(x_{j-1}-x_j)^2 + (U_{j-1}-U_j)^2}. \tag{2.8}$$

After discretizing both (2.7) *and* (2.8) *in time we obtain a discrete time, finite dimensional model* (1.1) *with the state vector U or* $\begin{bmatrix} U \\ x \end{bmatrix}$ *depending on whether only the PDE solution is assimilated or if the mesh locations are as well.*

2.1.3 Observations and Their Locations

Consider D observations at a fixed time t, $\mathbf{y}(t) = (y_1(t), ..., y_D(t))^T \in \mathbb{R}^D$ supported on a potentially time-dependent set of observation locations $\{x_j^o(t)\}_{j=1}^D$. In practice, these observations are related to the unknown truth $\mathbf{u^t}$ through the observation operator, suppressing the time dependence,

$$\mathbf{y} = \mathscr{H}(\mathbf{u^t}) + \eta, \qquad \eta \sim \mathscr{N}(0, R). \tag{2.9}$$

At a fixed time $t = t_n$, if $\{x_j^o(t_n)\}$ are a subset of the nodes of the common mesh, then (2.9) is exactly (1.2), and the DA update proceeds as detailed above. However, it is often the case that the $\{x_j^o(t_n)\}$ are not a subset of the nodes of the common mesh. In this case, it is natural to either interpolate the ensemble members onto a common mesh that contains the observations or interpolate the observations to the common mesh to obtain innovations of the form

$$\mathbf{y} = \mathscr{H}(\mathscr{I}_{co}(\mathbf{u^{e_i}})) + \eta\,, \quad \text{or} \quad \mathscr{I}_{oc}(\mathbf{y}) = \mathscr{I}_{oc}[\mathscr{H}(\mathscr{I}_{co}(\mathbf{u^{e_i}})) + \eta], \tag{2.10}$$

where $\mathscr{I}_{co}$ interpolates from the common mesh to the locations in the domain of the observation operator and $\mathscr{I}_{oc}$ maps from the observation locations to (part of) the common mesh.

2.2 Development of Adaptive Mesh Methods for DA

In this section we develop techniques for DA with adaptive moving meshes. These techniques are based upon the use of a metric tensor that describes the mesh. In particular, a non-uniform mesh is uniform with respect to the metric tensor being employed. The use of metric tensors is applicable not only in one spatial dimension, but also in higher spatial dimensions. We first describe the development of metric tensors using the so-called equidistribution and alignment conditions. Metric tensor intersections are introduced next and are used to combine meshes (e.g., the meshes of ensemble members) and potentially the locations of observations into what is in some sense an "averaged" mesh. We develop techniques for defining metric tensors that will concentrate mesh

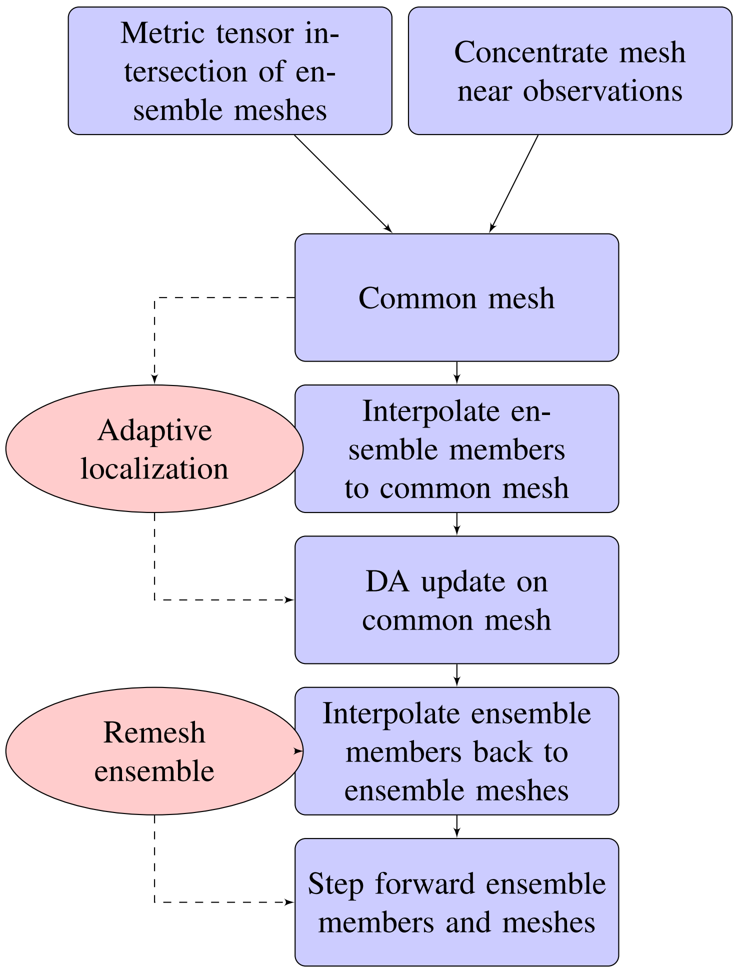
Metric tensor intersection of ensemble meshes
Concentrate mesh near observations
Common mesh
Adaptive localization
Interpolate ensemble members to common mesh
DA update on common mesh
Remesh ensemble
Interpolate ensemble members back to ensemble meshes
Step forward ensemble members and meshes

Figure 2.1: Algorithm for DA on an adaptive moving mesh. Optional steps are in red; required steps are in blue.

points near observation locations or other locations of interest and develop adaptive localization techniques based upon the use of metric tensors. A rough outline of the overall algorithm in given in Figure 2.1.

2.2.1 Metric Tensors and Adaptive Meshes

In one dimension, a mesh can be defined by determining the size of each of the elements of a mesh. In higher spatial dimensions, however, it is necessary to have conditions to also determine the shape and orientation of each element in addition to its size. For this and following discussions of adaptive mesh techniques, we follow the approach of [13].

Consider a polyhedral domain $\Omega \subset \mathbb{R}^d$ ($d \geq 1$), and let $\hat{K}$ be a reference element such that it is an equilateral d-simplex with unit volume. Then, given a simplicial mesh $\mathscr{T}_h$, for any element $K \in \mathscr{T}_h$, there is a unique invertible affine mapping $F_K : \hat{K} \to K$ such that $K = F_K(\hat{K})$. The size, shape, and orientation of $K \in \mathscr{T}_h$ can be obtained from F_K'.

One way to create adaptive meshes is through the use of a given symmetric positive-definite matrix-valued monitor function, $\mathbb{M} = \mathbb{M}(x)$, which is also sometimes called a metric tensor. This monitor function defines a metric, and a mesh that is uniform in this metric is said to be an $\mathbb{M}$-uniform mesh. That is, a mesh $\mathscr{T}_h$ is $\mathbb{M}$-uniform if and only if all elements have a constant volume and are equilateral in the metric $\mathbb{M}$. These conditions are called the equidistribution and alignment criteria ([13], section 4.1.1). To be specific, let $\mathbb{M}_K$ to be the average of $\mathbb{M}(x)$ over the element K, that is,

$$\mathbb{M}_K = \frac{1}{|K|} \int_K \mathbb{M}(x) dx. \tag{2.11}$$

Then the equidistribution condition is given by

$$\sqrt{\det(\mathbb{M}_K)}\, |K| = \frac{\sigma_h}{N}, \quad \forall K \in \mathscr{T}_h, \tag{2.12}$$

where $\sigma_h = \sum_{K \in \mathscr{T}_h} |K| \sqrt{\det(\mathbb{M}_K)}$ and N is the number of simplexes/elements in $\mathscr{T}_h$. The alignment condition is equivalent to

$$\frac{1}{d} \mathrm{tr} \left(\left(F_K'\right)^{-1} \mathbb{M}_k^{-1} \left(F_K'\right)^{-T} \right) = \det \left(\left(F_K'\right)^{-1} \mathbb{M}_K^{-1} \left(F_K'\right)^{-T} \right)^{\frac{1}{d}}, \quad \forall K \in \mathscr{T}_h \tag{2.13}$$

where $\det(\cdot)$ and $\mathrm{tr}(\cdot)$ denote the determinant and trace of a matrix, respectively. Given a user-prescribed monitor function $\mathbb{M} = \mathbb{M}(x)$, equations (2.12) and (2.13) can be used to define an energy functional, which in turn generates mesh movement; see detail in Section 2.3.1.

2.2.2 Forming the Common Mesh

At each data assimilation step, the common mesh is calculated by taking the metric tensor intersection of the monitor functions for each ensemble member; see, e.g., [29]. The metric tensor intersection is better understood in geometry as the intersection of unit balls in different norms that are associated with symmetric and positive definite (SPD) matrices. Mathematically, if A and B are SPD matrices of the same dimension, then their intersection is denoted by "$\cap$" and defined as

$$A \cap B = P^{-1}\text{diag}\left(\max(1,b_1),\cdots,\max(1,b_d)\right)P^{-T}, \tag{2.14}$$

where P is a nonsingular matrix such that

$$PAP^T = I, \qquad\qquad PBP^T = \text{diag}(b_1,\cdots,b_d). \tag{2.15}$$

It is not difficult to show that the unit ball in the norm associated with $A \cap B$ is contained in the unit balls in the norms associated with A and B. As a consequence, when A and B are two metric tensors (meaning they are SPD-matrix-valued functions), the mesh associated with $A \cap B$ will combine concentration patterns of the meshes associated with A and B.

Consider a DA scheme with N_e ensemble members u^{e_i}, $i = 1,...,N_e$ which are defined on different, independent meshes $\mathscr{T}_h^{e_i}$, $i = 1,...,N_e$. Assume that the ensemble meshes $\mathscr{T}_h^{e_i}$ and the common mesh $\mathscr{T}_h$ have the same number of mesh elements and vertices (denoted N and M, respectively), as well as the same connectivity. That is, they differ only in the location of the vertices. At each observational time-step, first interpolate the ensemble members from the ensemble meshes $\mathscr{T}_h^{e_i}$ to the current common mesh $\mathscr{T}_h$. Then compute the metric tensor for each of the ensemble members, denoted $\mathbb{M}_K^{e_i}$, and obtain a single metric tensor $\mathbb{M}_K^m$ by matrix intersection, $\mathbb{M}_K^m = \mathbb{M}_K^{e_1} \cap \mathbb{M}_K^{e_2} \cap ... \cap \mathbb{M}_K^{e_{N_e}}$, which will be used to generate the common mesh at the next time-step. In practice, this computation can be done sequentially. In 1D, the order does not matter since the metric tensor intersection corresponds to finding the maximum value. However, the order does matter in multi-dimensions and different orderings lead to different final metric tensors. While the examples we present in the sections that follow are robust with respect to the different orderings, we have used the Greedy Algorithm based on minimizing the determinant (which is equivalent to maximizing the area of the ellipsoid from the intersection) to determine an optimal ordering.

2.2.3 Concentration of Mesh Points near Observation Locations

Using metric tensors to define a common mesh gives some amount of control over the location of the mesh points. Specifically, the common mesh can be concentrated near specific points of interest, such as observation locations. In the context of data assimilation, it may be beneficial to have more mesh points near the observation locations $\{x_j^o(t)\}_{j=1}^{N_o}$, where N_o is the number of observations. The locations of the observations do not need to be fixed, as long as the location is known at the observational time t.

One strategy for ensuring that there are common mesh points near the observation locations is to simply fix the observation locations as points in the common mesh. However, this can lead to meshes that are ill-conditioned when the mesh points are not allowed to move freely with the dynamics of the solution. Instead of fixing the observation location as a point in the common mesh, a monitor function $\mathbb{M}_K^O$ can concentrate the mesh near the observation locations. Ideally, $\mathbb{M}_K^O$ is comparable to $\mathbb{M}_K^m$ in some measure at the locations where the mesh requires more points, and quickly decays away from these locations. One such choice is

$$\mathbb{M}_K^O = \sum_{j=1}^{N_o} \chi\left(\|x - x_j^o(t)\|\right) I,$$

where

$$\chi(w) = \left[e^{4w^2} - 1 + \frac{1}{\max\limits_K \sqrt{\det(\mathbb{M}_K^m)}} \right]^{-1}.$$

$$\mathbb{M}_K = \mathbb{M}_K^m \cap \mathbb{M}_K^O. \tag{2.16}$$

If the ensemble meshes alone determine the common mesh, set $\mathbb{M}_K = \mathbb{M}_K^m$. Conversely, if the locations of the observations alone determine the common mesh, set $\mathbb{M}_K = \mathbb{M}_K^O$. Note that this last scenario is similar to [6], where a fixed, non-uniform common mesh is employed based on the observation locations.

2.2.4 Adaptive Localization

A localization scheme can ensure that observations only affect nearby mesh points. There are broadly two categories of localization schemes: domain localization and covariance localization. For a covariance localization scheme, the covariance matrix is adjusted so that the analysis update is less affected by observations that are farther away. Several works explore this type of R- localization scheme, including in an adaptive sense [28, 22, 23]. The second category of localization schemes is domain localization, where the domain is decomposed into several subdomains. Domain localization ensures that an observation only affects the solution at mesh points within the same subdomain. It will not affect the solution outside of the given subdomain. One common domain localization scheme is that used in [15].

We define a metric tensor localization scheme (MT localization) as a domain localization scheme of [15] (cf. Section 2.3.5) but with the localization radii calculated for all mesh node as a function of the determinant of the monitor function as follows. Instead of having one predetermined radius of localization r, at each time-step we compute the localization radius for each node:

$$r_i = L\, e^{-\frac{d_i}{2d_{min}}}, \quad i = 1, \ldots, M \tag{2.17}$$

where $d_i = \min(\det(\mathbb{M}_K^m(x_i)), c)$, $d_{min} = \min_i d_i$, and $c > 0$ and $L > 0$ are the parameters offering some control over the localization regime. It is not difficult to see that

$$L\, e^{-\frac{c}{2d_{min}}} \leq r_i \leq \frac{L}{\sqrt{e}}. \tag{2.18}$$

This shows that the larger the value of L, the larger the localization radius can be. A smaller cutoff value c will increase the lower bound of the localization radius, ensuring that localization can still happen, even given a sharp front.

When combining the concentration of mesh points with the MT localization scheme, the localization radius should be calculated solely based on the meshes from the previous time-step and the ensemble solutions. This is done before the common mesh is concentrated near the observations. If not, the localization radius would be incorrectly computed from the concentration scheme instead of the ensemble solutions to the PDE.

2.2.5 Remeshing ensemble members

At the beginning of each observational time-step, the ensemble members u^{e_i} reside on their corresponding meshes $\mathscr{T}_h^{e_i}$. The analysis is computed on the common mesh, and the updated ensemble members are interpolated back to their individual meshes. Sometimes the analysis update is enough of a perturbation from the forecast that the meshes that worked for the forecast are no longer suitable for the analysis meshes.

If the perturbation is large enough, the forecast meshes might not still be appropriate for the analysis. One option is to add extra smoothing cycles before integrating. If necessary, we can remesh to find a mesh suitable for the analysis ensemble. The new meshes for ensemble members must satisfy the equidistribution and alignment criteria for the updated ensemble, and the result is the analysis ensemble $u^{e_i,a}$ residing on the updated meshes $\mathscr{T}_h^{e_i,a}$.

2.2.6 Algorithm for DA on Adaptive Meshes

The technique that has been developed here for DA with an adaptive moving mesh is summarized in Algorithm 1. Here we assume that all meshes have the same number of elements and nodes and the same connectivity. A main advantage of this is that those meshes can be viewed as deformations to each other and conservative interpolation schemes can be developed relatively easier between those meshes; e.g., see [30] or Section 2.3.3. It is important in the DA computation to conserve ensemble members at the interpolation steps 7 and 8 in Algorithm 1 since, otherwise, the mean zero assumption for the model error in (1.1) will be violated.

We note that it is not a requirement for meshes to have the same number of elements and nodes and the same connectivity for the metric tensor approach to ensemble DA with adaptive meshing. This approach can also be used with a meshing scheme where mesh points may be added or eliminated based on some pre-defined criteria and/or ensemble meshes and the common mesh can have different numbers of elements and nodes. However, precaution should be taken for interpolation between ensemble meshes and the common mesh to ensure conservation of the ensemble members and therefore mean zero of the model error.

Algorithm 1 EnKF on Adaptive Moving Meshes

```
 1: procedure DA UPDATE ON MOVING MESH
 2:     Compute $\mathbb{M}^m$
 3:     Compute $\mathbb{M}^O$
 4:     Take $\mathbb{M} = \mathbb{M}^m, \mathbb{M}^O$ or $\mathbb{M}^m \cap \mathbb{M}^O$
 5:     Compute common mesh $X$
 6:     (Optional) Adapt localization scheme based on $\mathbb{M}^m$
 7:     Interpolate $u^m$ and observations to common mesh $X$
 8:     DA update on common mesh $X$
 9:     Interpolate $u^{m,analysis}$ to individual meshes
10:     (Optional) Remesh ensemble meshes
11:     Integrate solutions forward in time until next observation
12: end procedure
```

2.3 Implementation Details

2.3.1 Defining Mesh Movement

The following is a summary of the moving mesh PDE (MMPDE) approach developed in [8, 9, 11, 12, 13]. The central idea of the MMPDE moving mesh method is to view any nonuniform mesh as a uniform one in some metric $\mathbb{M}$; that is, the elements have a constant volume and are equilateral in the metric $\mathbb{M}$. These conditions are called the equidistribution and alignment criteria; see (2.12) and (2.13). It has been shown that if a mesh begins as nonsingular (that is, the elements have positive volume), it will remain nonsingular for all time under the MMPDE method. Furthermore, the height of each of the elements will be bounded above and below by some positive constants that depend on $\mathbb{M}$ and the initial mesh [10, Theorem 4.1].

The metric tensor $\mathbb{M} = \mathbb{M}(x)$ is used to control the size, shape, and orientation of the elements of the mesh to be generated. Various metric tensors have been developed in [14]. For this paper, we consider a Hessian- based metric tensor defined for each element $K \in \mathscr{T}_h$ as

$$\mathbb{M}_K = \det\left(I + \frac{1}{\alpha_h}|H_K(u)|\right)^{-\frac{1}{d+4}} \left(I + \frac{1}{\alpha_h}|H_K(u)|\right), \tag{2.19}$$

where $H_K(u)$ is the Hessian or a recovered Hessian of the state vector $u \in \mathbb{R}^d$ on the element K; $|H_K(u)| = Q\text{diag}(|\lambda_1|, ..., |\lambda_d|)Q^T$ with $Q\text{diag}(\lambda_1, ..., \lambda_d)Q^T$ being the eigen-decomposition of $H_K(u)$; and α_h is a regularization parameter defined through the following algebraic equation:

$$\sum_{K\in\mathscr{T}_h} |K| \det\left(I + \frac{1}{\alpha_h}|H_K(u)|\right)^{\frac{2}{d+4}} = 2\sum_{K\in\mathscr{T}_h} |K| \det\left(|H_K(u)|\right)^{\frac{2}{d+4}}.$$

The metric tensor given by equation (2.19) is known to be optimal for the L^2-norm of linear interpolation error [14].

To generate the $\mathbb{M}$-uniform mesh $\mathscr{T}_h$, we use here an approach different from (2.3) where the coordinators of the mesh nodes are evolved directly. Instead, we evolve the coordinators of the nodes of an intermediate mesh and obtain the new mesh through this intermediate mesh and interpolation. An advantage of this approach is that its formulation is simpler than that of the direct approach (cf. [9]). To start with, we introduce the reference computational mesh $\hat{\mathscr{T}}_c$ which is uniform in the Euclidean metric, and the computational mesh $\mathscr{T}_c$. The reference computational mesh $\hat{\mathscr{T}}_c = \{\hat{\xi}_j\}_{j=1}^M$ has the same connectivity and the same number of vertices and elements as $\mathscr{T}_h$ and stays fixed in the computation. The computational mesh $\mathscr{T}_c = \{\xi_j\}_{j=1}^M$ serves as an intermediate variable. In this setting, for any element K in $\mathscr{T}_h$, there is a unique corresponding element K_c in $\mathscr{T}_c$. The affine map between K_c and K and its Jacobian matrix are denoted by F_K and F_K', respectively. With this new setting, the equidistribution and alignment criteria for $\mathbb{M}$-uniform meshes have a similar form as those in equations (2.12) and (2.13).

To generate a mesh satisfying these conditions as closely as possible, define an energy functional as

$$\begin{aligned} I_h(\mathscr{T}_h,\mathscr{T}_c) =& \frac{1}{3}\sum_{K\in\mathscr{T}_h} |K|\det(\mathbb{M}_K)^{\frac{1}{2}}\left(\mathrm{tr}((F_K')^{-1}\mathbb{M}_K^{-1}(F_K')^{-T})\right)^{\frac{3d}{4}} \\ &+ \frac{1}{3}d^{\frac{3d}{4}}\sum_{K\in\mathscr{T}_h}|K|\det(\mathbb{M}_K)^{\frac{1}{2}}\left(\det(F_K')\det(\mathbb{M}_K)^{\frac{1}{2}}\right)^{-\frac{3}{2}}, \end{aligned} \tag{2.20}$$

which is a Riemann sum of a continuous functional developed in [7] based on mesh equidistribution and alignment. Note that $I_h(\mathscr{T}_h,\mathscr{T}_c)$ is a function of the coordinates of the nodes of $\mathscr{T}_h$ and $\mathscr{T}_c$.

Taking $\mathscr{T}_h$ as the current mesh $\mathscr{T}_h^m$, the MMPDE approach defines the mesh equation as a gradient system of $I_h(\mathscr{T}_h,\mathscr{T}_c)$,

$$\frac{d\xi_j}{dt} = -\frac{\det(\mathbb{M}(x_j))^{\frac{1}{2}}}{\tau}\left(\frac{\partial I_h(\mathscr{T}_h^m,\mathscr{T}_c)}{\partial \xi_j}\right)^T, \quad j=1,\cdots,M \tag{2.21}$$

where $\partial I_h/\partial\xi_j$ is considered as a row vector, $\tau > 0$ is a parameter used to adjust the response time of mesh movement to the changes in $\mathbb{M}$.

Define the function G associated with the energy (2.20) as

$$G(\mathbb{J},\det(\mathbb{J})) = \frac{1}{3}\det(\mathbb{M}_K)^{\frac{1}{2}}(\mathrm{tr}(\mathbb{J}\mathbb{M}_K^{-1}\mathbb{J}^T))^{\frac{3d}{4}} + \frac{1}{3}d^{\frac{3d}{4}}\det(\mathbb{M}_K)^{\frac{1}{2}}\left(\frac{\det(\mathbb{J})}{\det(\mathbb{M}_K)^{\frac{1}{2}}}\right)^{\frac{3}{2}}, \tag{2.22}$$

where $\mathbb{J} = (F_K')^{-1} = E_{K_c}E_K^{-1}$ and the edge matrices of K and K_c are $E_K = [x_1^K - x_0^K, \cdots, x_d^K - x_0^K]$ and $E_{K_c} = [\xi_1^K - \xi_0^K, \cdots, \xi_d^K - \xi_0^K]$, respectively. Using the notion of scalar-by-matrix differentiation [9], it is not difficult to find the derivatives of G with respect to $\mathbb{J}$ and $\det(\mathbb{J})$ as

$$\frac{\partial G}{\partial \mathbb{J}} = \frac{d}{2}\det(\mathbb{M}_K)^{\frac{1}{2}}(\mathrm{tr}(\mathbb{J}\mathbb{M}_K^{-1}\mathbb{J}^T))^{\frac{3d}{4}-1}\mathbb{M}_K^{-1}\mathbb{J}^T, \tag{2.23}$$

$$\frac{\partial G}{\partial \det(\mathbb{J})} = \frac{1}{2}d^{\frac{3d}{4}}\det(\mathbb{M}_K)^{-\frac{1}{4}}\det(\mathbb{J})^{\frac{1}{2}}. \tag{2.24}$$

Substituting (2.22)-(2.24) into (2.21) yields

$$\frac{d\xi_j}{dt} = \frac{\det(\mathbb{M}(x_j))^{\frac{1}{2}}}{\tau} \sum_{K\in\omega_j} |K| v_{j_K}^K, \quad j = 1, \cdots, M \tag{2.25}$$

where ω_j is the element patch associated with the vertex x_j, j_K is the local index of x_j on K, and $v_{j_K}^K$ is the local velocity contributed by the element K to the vertex j_K. The local velocities $v_{j_K}^K$, $j_K = 1, \cdots, d$ are given by

$$\begin{bmatrix} (v_1^K)^T \\ (v_2^K)^T \\ \vdots \\ (v_d^K)^T \end{bmatrix} = -E_K^{-1}\frac{\partial G}{\partial \mathbb{J}} - \frac{\partial G}{\partial \det(\mathbb{J})} \frac{\det(E_{K_c})}{\det(E_K)} E_{K_c}^{-1}, \quad v_0^K = -\sum_{j_K=1}^{d} v_{j_K}^K. \tag{2.26}$$

Integrating the mesh equation (2.25) over a physical time-step, with the proper modifications for boundary vertices and with the initial mesh $\hat{\mathscr{T}}_c$, yields the new computational mesh $\mathscr{T}_c^{new}$ which forms a correspondence with the current mesh $\mathscr{T}_h^m$, i.e., $\mathscr{T}_h^m = \Phi_h(\mathscr{T}_c)$. The new common mesh $\mathscr{T}_h^{new}$ is defined as $\mathscr{T}_h^{new} = \Phi_h(\hat{\mathscr{T}}_c)$, which can be computed using linear interpolation.

It is common practice in moving mesh computation to smooth the metric tensor for smoother meshes. To this end, we apply a low-pass filter [13] to the metric tensor several sweeps every time it is computed.

2.3.2 DG Discretization

Numerical results will be presented in the following sections that illustrate the DA procedure using the 1D and 2D inviscid Burgers equations. Since the Burgers equation is hyperbolic and can have discontinuous solutions such as shocks, we use the discontinuous Galerkin method (DG) for its spatial discretization. DG is a type of finite element method with the trial and test function spaces consisting of discontinuous, piecewise polynomials. It is known to be a particularly powerful numerical tool for the simulation of hyperbolic problems and has the advantages of high-order accuracy, local conservation, geometric flexibility, suitability for handling mesh-adaptivity, extremely local data structure, high parallel efficiency, and a good theoretical foundation for stability and error estimates. Over the last few decades, the DG method has been used widely in scientific and engineering computation.

Specifically, we use a quasi-Lagrangian moving mesh DG method (MMDG) to solve the Burgers equation on moving meshes [30]. The method treats the mesh movement continuous in time and leads to an extra convective term (cf. equation (2.2)) in the resulting discrete equations to reflect the mesh movement. Importantly, the method is (mass) conservative so that the model error has mean 0. In our computation, we use piecewise linear polynomials (P^1 -DG) for spatial discretization and a third-order Strong Stability Preserving (SSP) Runge-Kutta scheme for temporal discretization. The reader is referred to [30] for details of the MMDG method.

2.3.3 DG Interpolation

From Algorithm 1, we see that interpolation is needed between the ensemble meshes and the common mesh. Since these meshes are assumed to be deformations to each other, we can perform interpolation by solving a differential equation.

To be specific, we consider the deforming meshes $\mathscr{T}_h^{old}$ and $\mathscr{T}_h^{new}$ and the state variable u that needs to be interpolated. We define a deforming mesh $\mathscr{T}_h(\zeta)$ (with $\zeta \in [0,1]$) from $\mathscr{T}_h^{old}$ to $\mathscr{T}_h^{new}$ as a mesh with the nodal positions and velocities as

$$x_i(\zeta) = (1-\zeta)x_i^{old} + \zeta x_i^{new}, \quad i = 1,\ldots,M \tag{2.27}$$

$$\dot{x}_i = x_i^{new} - x_i^{old}, \quad i = 1,\ldots,M. \tag{2.28}$$

Then the interpolation can be viewed as solving the following linear convective PDE on the moving mesh $\mathscr{T}_h(\zeta)$ over $\zeta \in [0,1]$:

$$\frac{\partial u}{\partial \zeta}(x,\zeta) = 0, \quad (x,\zeta) \in \Omega \times (0,1], \tag{2.29}$$

with the initial values $u(x,0)$ as those of u on $\mathscr{T}_h^{old}$.

A DG-interpolation scheme has been studied in [30] where DG and a SSP Runge-Kutta scheme are used to discretized (2.29) in space and time, respectively. The scheme is conservative and positivity-preserving and can be high-order. As mentioned before, the mass conservation is important for interpolation between ensemble meshes and the common mesh to maintain the mean zero feature of the model error. Like for the PDE solver, we use this scheme for interpolation with piecewise linear polynomials (P^1-DG) for spatial discretization and a third-order SSP Runge-Kutta scheme for temporal discretization; see [30] for detail.

2.3.4 DA Implementation

For our numerical experiments we employ a Local Ensemble Transform Kalman Filter (LETKF) based on [15]. In general, an ETKF code uses a linear transform to have control over the resulting sample covariance so that the covariance of the analysis update, P^f, exactly satisfies the identity $P^f_{j+1} = (I - K_{j+1}H)P^b_{j+1}$.

Let $\hat{m}_{n+1}$ be the ensemble mean at time t_{n+1}, and let $\hat{u}^{(i)}_{n+1}$, $i = 1, \ldots N_e$ be the ensemble forecast. Then define the perturbation matrix

$$\hat{X}_{n+1} = \frac{1}{\sqrt{N_e - 1}} \left[\hat{u}^{(1)}_{n+1} - \hat{m}_{n+1}, \hat{u}^{(2)}_{n+1} - \hat{m}_{n+1}, \ldots, \hat{u}^{(N_e)}_{n+1} - \hat{m}_{n+1}\right], \tag{2.30}$$

and the sample covariance is given by $P^b_{n+1} = \hat{X}_{n+1}\hat{X}^T_{n+1}$. Consider the following transformation:

$$T_{n+1} = \left[I + \left(H\hat{X}_{n+1}\right)^T R^{-1} \left(H\hat{X}_{n+1}\right)\right]^{-1}, \tag{2.31}$$

and set $X_{n+1} = \hat{X}_{n+1}T^{\frac{1}{2}}_{n+1}$. Then

$$P^f_{n+1} := X_{n+1}X^T_{n+1} = (I - K_{n+1}H)P^b_{n+1}, \tag{2.32}$$

as desired.

2.3.5 Localization Techniques

We will compare the metric tensor based localization scheme developed in Section 2.2.4 with some commonly used localization schemes. Recall from (1.5) the Kalman gain $\mathbf{K}_{n+1}$. R-localization modifies the Kalman gain through the Schur product of a localization matrix ρ with the covariance matrix. One of the most common ways to define ρ is through the Gaspari-Cohn (GC) correlation function, which is a fifth order polynomial that decays to zero.

$$\mathscr{C}(r) = \begin{cases} -\frac{1}{4}r^5 + \frac{1}{2}r^4 + \frac{5}{8}r^3 - \frac{5}{3}r^2 + 1, & 0 \le r \le 1 \\ \frac{1}{12}r^5 - \frac{1}{2}r^4 + \frac{5}{8}r^3 + \frac{5}{3}r^2 - 5r + 4 - \frac{2}{3}r^{-1}, & 1 < r \le 2 \\ 0, & 2 < r. \end{cases} \tag{2.33}$$

GC localization can be applied to either the model space or the observation space. In the model space, the localization matrix is given by

$$\rho_{ij} = \mathscr{C}(|x_i - x_j|/L), \quad i,j = 1,...,M \tag{2.34}$$

where x_i and x_j are the positions of the ith and jth nodes and $L > 0$ is a pre-determined localization radius. This matrix is Schur-multiplied with the ensemble covariance matrix, resulting in the Kalman update

$$K_{GCmod} = \left(\rho \circ P^b\right) H^T \left(H\left(\rho \circ P^b\right) H^T + R\right)^{-1}. \tag{2.35}$$

For localization in the observation space, the localization matrix must be Schur multiplied to both $\left(P^b H^T\right)$ and $\left(HP^bH^T\right)$. Therefore, two localization matrices are needed,

$$[\rho_1]_{i,j} = \mathscr{C}(|x_i - y_j|/L), \qquad i,j = 1,...,M \tag{2.36}$$

$$[\rho_2]_{i,j} = \mathscr{C}(|y_i - y_j|/L), \qquad i,j = 1,...,M. \tag{2.37}$$

Then the Kalman gain is given by

$$K_{GCobs} = \rho_1 \circ \left(P^b H^T\right)\left(\rho_2 \circ \left(HP^bH^T\right) + R\right)^{-1}. \tag{2.38}$$

We next outline the domain localization scheme developed in [15]. Define r to be the radius of localization. Then for every mesh point x_i, if there is an observation y located at the point x^o such that $\|x_i - x^o\|_2 \leq r$, the analysis is given by the EnKF update in equation (1.5). If not, then the analysis update is equal to the forecast predicted by the model.

In many implementations, this radius of influence is predetermined and constant over time and space. However, there may be instances where this should be dynamic in time and space. For example, if the solution has a traveling front or shock that travels across the domain, there should be a smaller radius of localization near the region where the gradient is large and a larger radius where the solution is relatively constant, and this localization scheme should move across the domain with the traveling front or shock.

To see this, consider a solution $u_n(x) \in \mathbb{R}$ with a single observation at x^o and a large gradient beginning at x_k just past this observation location; that is, $0 < x_k - x^o \ll 1$. For x_i outside of the localization radius, the observation does not affect the analysis, so $u_{n+1}(x_i) = \hat{u}_{n+1}(x_i)$. For x_i close to the observation, consider the EnKF update for a single ensemble member, omitting the ensemble superscripts and time subscripts:

$$u(x_i) = (I - \mathbf{K}H)\hat{u}(x_i) + \mathbf{K}y, \tag{2.39}$$

where $\mathbf{K} = P^bH^T\left(HP^bH^T + R\right)^{-1}$. For the sake of simplicity, assume $R = \alpha I$ and $H = e_k^T$, where e_k denotes the k^{th} unit vector. Then the analysis update at the point x_i close to the observation gives us

$$u^a(x_i) = u^f(x_i) + \mu P^b_{ik} \quad \text{with} \quad \mu = \left[P^b_{kk} + \frac{1}{\alpha}\right](y - x^f_k).$$

In particular, the larger the value of P^b_{ik} is, the greater the difference will be between the analysis and the forecast. This can be especially problematic if P^b_{ik} is large for $i < k$, that is, after the shock. In that case, the analysis will be changed to be much closer to the observations, and the steep front will be smoothed out.

This problem is avoided by using a smaller radius of localization near the shock and a larger radius of localization farther away from it. Fortunately, the monitor function obtained in the adaptive moving mesh algorithm determines where mesh points will be closer together and where they will be spread far apart; by proxy, this shows where the shock or other feature of interest exists. In this way, the monitor function can inform what the localization parameter should be over time and space, enabling a dynamic update of the localization variable.

LETKF employs localization within the ensemble transform Kalman filter so that only model variables located at mesh points within a predetermined radius of an observation will assimilate that observation. This not only localizes the influence of observations but also provides a dimension reduction by creating reduced dimensional subproblems on which assimilation is performed independently.

A covariance-based localization scheme, which uses a Schur product applied to the covariance matrix, is problematic when working in this reduced dimension. For example, the reduced dimension implementation uses $\hat{X}_{n+1}$, taken from the Cholesky decomposition of the covariance matrix P^b_{n+1}, rather than P^b_{n+1} itself. Covariance localization would use a Schur product to adjust this covariance matrix, but in doing so, it could result in a matrix with negative eigenvalues. Therefore, a covariance localization scheme is not easily implemented into the LETKF code. For the experiments where we compare the MT localization scheme to covariance localization schemes like Gaspari-Cohn, we use a traditional ETKF code.

Chapter 3

Numerical Results, Part I

The following presents the application of these methods to the one and two dimensional inviscid Burgers equations. We generate synthetic observations by sampling from a truth run, obtained by solving this equation on an adaptive moving mesh. The ensemble members are initialized as perturbations of the initial conditions. Efficacy of the DA scheme is measured by the root mean squared error (RMSE), which is calculated as

$$\text{RMSE} := \frac{1}{\sqrt{M}} \| u^{\text{truth}} - \bar{u} \|_2, \tag{3.1}$$

where $\bar{u}$ is the analysis mean. A DA procedure is generally considered stable if its asymptotic behavior is on the order of the square root of the norm of the observation error. The RMSE in the experiments that follow is averaged over 10 runs.

3.1 Common Experimental Set-Up

The next two sections explore the use of the LETKF on the one- and two- dimensional inviscid Burgers equation. We perform identical twin experiments where the truth is generated using no model noise and observations are formed from the truth by applying the observation operator and adding noise using the observation error model with covariance matrix $R = 0.01I$. Among the parameters to be tuned are the number of mesh points and the number of ensemble members. Generally speaking, more mesh points correspond to more accurate numerical solutions, lowering the RMSE. However, moving mesh methods generally require fewer mesh points than a fixed, uniform mesh. For the 1D Burgers experiments, we use $M = 50$ mesh points, and for the 2D Burgers experiments, we use a $M = 15 \times 15$ mesh. Increasing the number of mesh points beyond the values chosen had little impact on the RMSE. The rule of thumb for ensemble-based DA schemes is that the number of ensemble members should be large enough to span the unstable subspace. For both the 1D and 2D experiments, we found $N_e = 5$ ensemble members to be sufficient. That is, for larger N_e, we found that there was no substantial improvement in RMSE.

In the 1D Burgers experiments we observe the truth at $D = 5$ locations with an observational time-step of $\Delta t = 0.5$, and then add artificial observation error ($\eta \sim \mathcal{N}(0, 0.01)$). To avoid the observations all occurring in one region of the spatial domain, we space them linearly throughout the domain, and then perturb the locations by a small amount. These observation locations are randomly chosen, but once chosen at the beginning of each trial, they remain fixed. The truth (and observations) are taken from a fine mesh (100 mesh points) to ensure that it is fully resolved. For the 2D Burgers experiments, we use an observational time-step of $\Delta t = 0.5$ with $D = 16$ observation locations that are also uniformly spaced in a 4×4 mesh, and then perturbed. Unless otherwise stated, the observational error covariance R, model error covariance Σ, and prior distribution P_b^0 are all set to $0.01I$.

For both the 1D and the 2D cases, the localization radius and inflation factors are tuned simultaneously. For MT localization, this involves determining the parameter L shown in equation (2.17), which directly controls the maximum radius of localization. In the 1D case, we choose not to artificially limit the minimum value of the MT localization radius by choosing c larger than the maximum of $\mathbb{M}_K$ in (2.17); numerical results suggest that $c = 8$ is sufficient. In the 2D case, we keep $c = 8$ and note that this does affect the minimum localization radius, but that it also results in a stable DA scheme. For the GC localization schemes, this tuning experiment involves tuning the parameter L as given in equations (2.34), (2.36), and (2.37). After each of the localization schemes has been tuned, the time series RMSE of the different localization schemes is directly compared. Finally, we compare results when choosing $\mathbb{M}^m$, $\mathbb{M}^O$, or $\mathbb{M}^m \cap \mathbb{M}^O$ for the common mesh. When considering long-run RMSE results, we consider the RMSE only after the DA scheme has stabilized so as to evaluate the asymptotic behavior of the DA procedure. Based on numeric results, we present the RMSE for 1D Burgers on the time interval $[25,100]$ and for 2D Burgers on the time interval $[15,50]$. These experiments and the parameters used can be found in Table 3.1.

Experiment	**Description**	**Model**	$\Sigma, \mathbf{R}$	**Inflation Factor**	**M**	**Loc. Scheme**	**Mesh Choice**
1	Localization and Inflation Tuning	(3.2)	$0.01 \cdot I$	varies	50	varies	$\mathbb{M}^m \cap \mathbb{M}^O$
2	Compare Loc. Schemes	(3.2)	$0.01 \cdot I$	varies	50	varies	$\mathbb{M}^m \cap \mathbb{M}^O$
3	Compare Meshes	(3.2)	$0.01 \cdot I$	1.1	50	MT	varies
4	Compare Errors	(3.2)	varies	1.1	50	MT	$\mathbb{M}^m \cap \mathbb{M}^O$
5	Localization and Inflation Tuning	(3.4)	$0.01 \cdot I$	varies	225	varies	$\mathbb{M}^m \cap \mathbb{M}^O$
6	Compare Loc. Schemes	(3.4)	$0.01 \cdot I$	varies	225	varies	$\mathbb{M}^m \cap \mathbb{M}^O$
7	Compare Meshes	(3.4)	$0.01 \cdot I$	1.1	225	MT	varies
8	Compare Errors	(3.4)	varies	1.1	225	MT	$\mathbb{M}^m \cap \mathbb{M}^O$
9	Noisy Data	(3.4)	$0.01 \cdot I$; data varies	1.1	225	MT	$\mathbb{M}^m \cap \mathbb{M}^O$
10	Interpolation	(3.4)	$0.01 \cdot I$	1.1	225	MT	$\mathbb{M}^m \cap \mathbb{M}^O$

Table 3.1: Summary of experiments.

3.2 Inviscid Burgers Equation - A One-Dimensional Example

Consider the one-dimensional inviscid Burgers equation as given by

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0, \quad x \in [0,S), \quad t \in (0,T], \tag{3.2}$$

with initial condition

$$u(x,0) = \frac{1}{2} + \sin\left(\frac{2\pi}{S}x\right) \tag{3.3}$$

and periodic boundary conditions. As time progresses, a shock forms and propagates to the right. This traveling shock makes the 1D inviscid Burgers equation a good candidate for an adaptive moving mesh, as more mesh points are needed near the location of the shock to sufficiently resolve the numerical solution. For the experiments that follow, we consider the spatial domain $[0,20)$ so that $S = 20$ over the time interval $(0,100]$.

3.2.1 Experiment 1: Tuning localization and inflation

The localization parameters for the various localization schemes are tuned simultaneously with the inflation parameter. Equation (2.33) combined with either (2.34) or (2.36)-(2.37) implies that choosing a GC localization parameter of $L = 10$ for the GC localization would imply that the entire domain is affected by the localization scheme. Therefore, in this experiment, the GC localization parameters vary from $L = 0.5$ to $L = 10$. Similarly, a localization parameter of $L = 20$ means that the entire domain could potentially be affected by the MT localization scheme. For the MT localization tuning, L varies from $L = 0.5$ to $L = 20$. For all three localization schemes, the multiplicative inflation factor varies from $\rho = 1$ (no inflation) to $\rho = 1.5$.

As seen in Figure 3.1, the MT localization scheme is robust with respect to the tuning of the localization parameter. The GC localization requires much more careful tuning in both the model space and observation space. More specifically, the GC localization schemes work well when the localization parameter L is less than or equal to 1. If the localization parameter is greater than 1, the

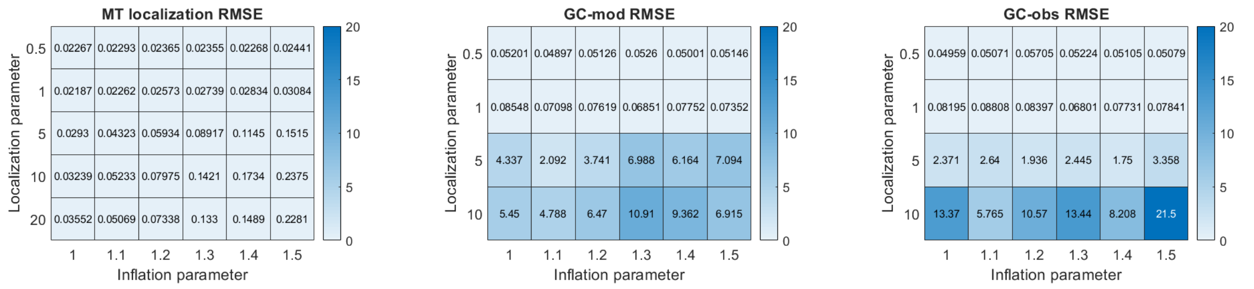

Figure 3.1: Results for simultaneous tuning of localization and inflation for the MT, GC-mod, and GC-obs localization schemes.

Schur product of the localization matrix and the covariance matrix is no longer positive definite, and the localization scheme performs poorly.

As explained in Section 2.2.4, The MT localization scheme is a domain localization scheme based off $\mathbb{M}_K^m$, the metric tensor intersection of the metric tensors for the ensemble meshes. Since each of the ensemble meshes will have a concentration of points near a shock or large gradient, the adaptive common mesh will also have more mesh points where the solution requires a finer resolution. MT localization reduces the localization radius at these types of interfaces. The relationship between the distance between nodes and the localization variable is shown in Figure 3.2.

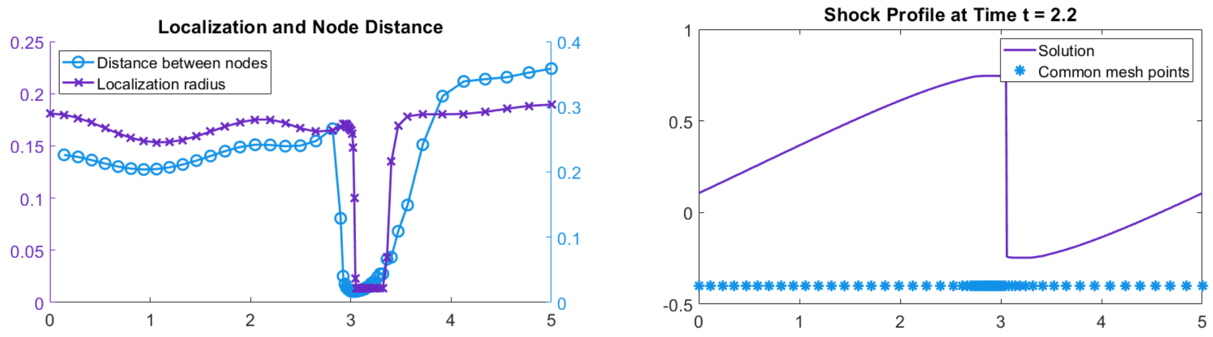

Figure 3.2: Relationship between nodal distance and adaptive localization parameter. The distance between the nodes is positively correlated with the localization radius.

3.2.2 Experiment 2: Localization Scheme Comparison

As shown in Experiment 1, the MT localization is much more robust with respect to the tuning parameters than either of the GC localization schemes. Consider the following tuned localization schemes:

- MT localization with multiplicative inflation parameter 1.1 and localization parameter 1.
- GC-mod localization with multiplicative inflation parameter 1.1 and localization parameter 0.5.
- GC-obs localization with multiplicative inflation parameter 1.0 and localization parameter 0.5.

We compare the time series RMSE of each of these localization schemes in Figure 3.3. In these experiments, $\Sigma = 0.01I$, so the localization scheme is considered a success if it is on the order of 0.1. Both the tuned GC-mod localization and the MT localization can be considered a success in this metric, but the MT localization has a considerably lower RMSE than both GC localization schemes.

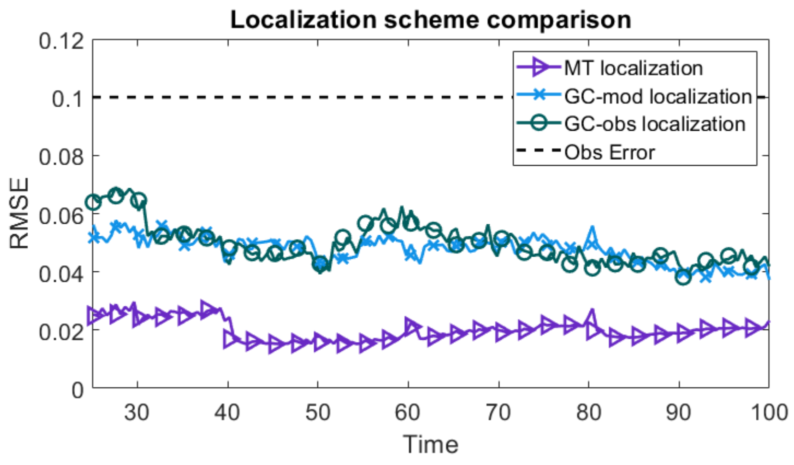

Figure 3.3: Time series RMSE comparison for the different localization schemes for the 1D inviscid Burgers equation. Each trial uses the tuned localization parameters found in Experiment 1.

3.2.3 Experiment 3: Choice of common mesh

We consider three choices for the metric tensor associated with the common mesh $\mathbb{M} \in \{\mathbb{M}^O, \mathbb{M}^m, \mathbb{M}^m \cap \mathbb{M}^O\}$. A comparison of these common meshes is shown in Figure 3.4, using the tuned localization and inflation parameters found in Experiment 1. While all three choices for the common mesh produce stable DA procedures, using $\mathbb{M}^O$ or $\mathbb{M}^m \cap \mathbb{M}^O$ will concentrate the common mesh near the observation locations, reducing the interpolation error. If the interpolation of observations must be avoided at all costs, the user can also specify the observation locations as fixed points in the common mesh. However, this can lead to increased skewness and potential singularity in the mesh, as shown in Figure 3.5.

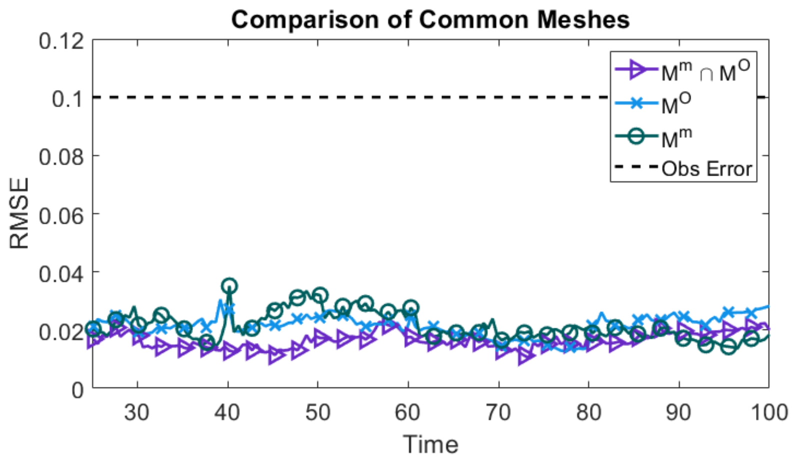

Figure 3.4: Time series RMSE for tuned DA with the 1D inviscid Burgers equation using different choices for the common mesh.

The time evolution of the meshes for the 1D inviscid Burgers equation are shown in Figure 3.5. To better illustrate this phenomenon, we consider a reduced spatial domain of $[0,5]$ $(S = 5)$ and update the initial condition accordingly. As the shock forms and propagates to the right, the mesh points of $\mathbb{M}^m$ evolve with the shock, as shown in the leftmost plot of Figure 3.5. However, if an observation location, say at $x = 3.8$, is fixed in the common mesh, it will prohibit the nodes before it from moving past it, as shown in the middle plot of Figure 3.5. Choosing $\mathbb{M}^m \cap \mathbb{M}^O$ achieves the goal of following the solution dynamics while still concentrating the mesh near the observation location. (Note that if $\mathbb{M} = \mathbb{M}^O$ and the location is static, this common mesh will not change in time.)

As shown in Figure 3.5, there are fewer mesh points near the observation in the ensemble mesh with or without the fixed point than there are in the mesh that also concentrates the mesh points near the observation. Figure 3.6 tallies the number of mesh points within the radius 0.5 of the shock

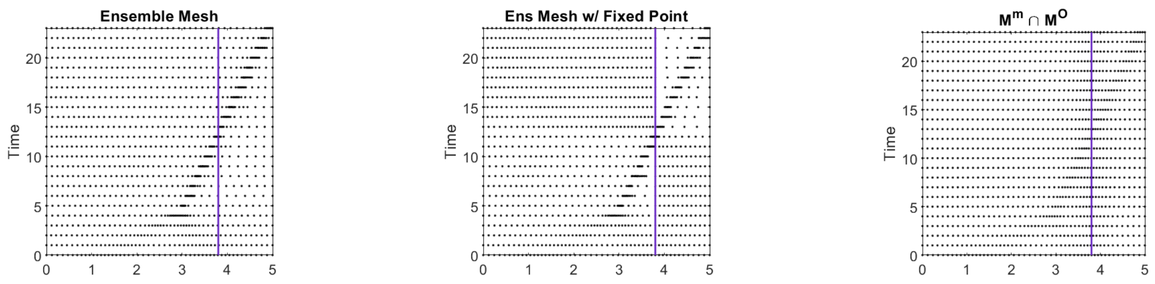

Figure 3.5: A comparison of meshing variations. In the leftmost plot, the mesh evolves naturally with the metric tensor intersection of the ensemble members. In the middle plot, the observation location $x = 3.8$ is fixed in the mesh to reduce observation interpolation error. In the right plot, the mesh evolves according to the metric tensor intersection, but it is also concentrated near the location $x = 3.8$.

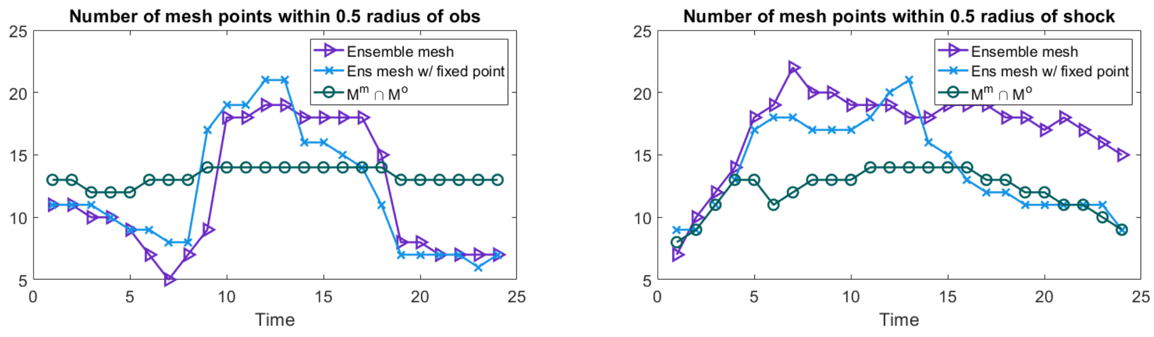

Figure 3.6: The number of mesh points near the observations varies widely for the ensemble mesh with or without the fixed point. The number of points near the observation has less variance when the mesh points are also concentrated near the observation location.

and of the observation for each of the above meshes. The average and variance of these values is given in Table 3.2. While the meshes formed from the ensemble members, with and without the fixed point, do have an adequate number of mesh points near the observation at times, that only coincides with the shock passing through the observation point. At other times, there are relatively few mesh points near that observation. This is easily seen in the large variances in Table 3.2.

	Ensemble Mesh	**Ens Mesh w/ Fixed Point**	$\mathbb{M}^{\mathbf{m}} \cap \mathbb{M}^{\mathbf{O}}$
Near Shock	$(17.13, 11.51)$	$(14.04, 13.61)$	$(12.13, 3.16)$
Near Observation	$(11.83, 25.28)$	$(11.96, 23.87)$	$(13.29, 0.48)$

Table 3.2: Mean and variance of the number of points near (within a 0.5 radius of) the shock and observation for each of the three meshes listed above.

The benefit of having mesh points concentrated near observations is evident in the case where

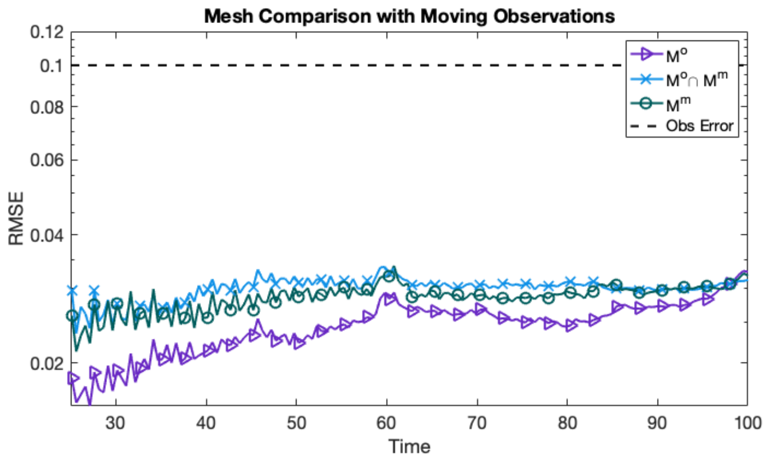

Figure 3.7: The common mesh based on observation locations performs better than the other choices of common mesh in the case where we have observations that are located away from the shock.

observations are sparse and occur in places where the mesh points would otherwise not be concentrated. For example, consider the 1D inviscid Burgers equation with two observation locations located away from the shock, and suppose that the location of the observations move at the same rate as the shock as it propagates forward in time. In that case, the observation locations may not see a concentration of mesh points pass through unless the user specifically concentrates the mesh in that area. This is the setup of the experiment shown in Figure 3.7. Over time, the solution flattens due to numerical dissipation, and all choices of the common mesh perform equally well. However, the mesh based on observation locations outperforms the other choices of common mesh before the shock decays.

3.2.4 Experiment 4: Comparison of error covariances

Using the tuned MT localization scheme from experiment 1, we test the robustness of this DA procedure on (3.2) with different error covariances. The results in Figure 3.8 show stable RMSE, especially for larger error covariances. The spikes in the RMSE for smaller error covariance correspond to times in which the shock passes through the boundary. In this implementation the PDE satisfies periodic boundary conditions while the mesh satisfies Dirichlet boundary conditions so that when shock passes through the boundary, the PDE is not approximated as accurately.

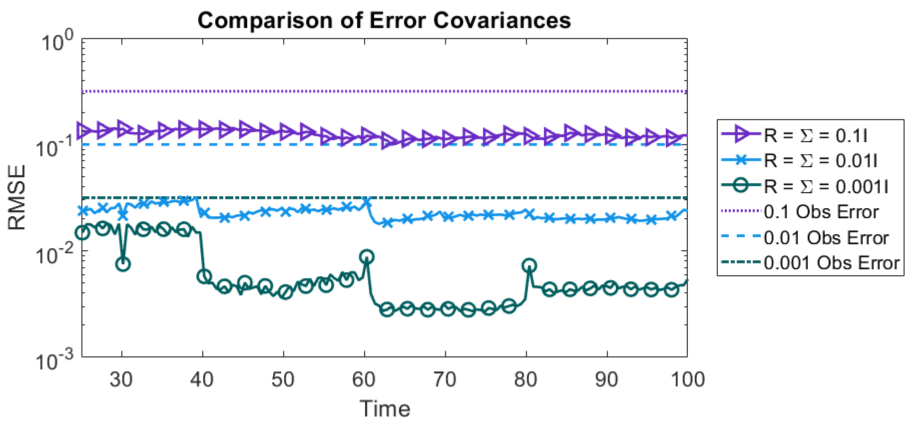

Figure 3.8: Time series RMSE for tuned MT localization with different error covariances.

3.3 Inviscid Burgers Equation - A Two-Dimensional Example

Consider the two-dimensional Burgers equation

$$u_t + uu_x + uu_y = 0, \tag{3.4}$$

with $\Omega = (-0.5, 1) \times (-0.5, 1)$, $t \in (0, 5]$, and periodic boundary conditions. Given the initial condition

$$u = \exp(-\gamma(x^2 + y^2)),$$

with $\gamma = -\log(10^{-16})$, the solution will have a Gaussian bump that propagates diagonally to the upper right corner of the domain.

The choice of concentration with the common mesh is easily seen in 2 dimensions. Figure 3.9 shows a snapshot of the numerical solution of the 2D inviscid Burgers equation (3.4) at time $t = 1.6667$. At this time, the front has fully formed and is beginning to propagate to the upper left corner. There are five observations sampled along the line $x = 0.75$. The truth is shown in the leftmost panel, followed by the three choices for the common meshes computed from $\mathbb{M}_K^m$, $\mathbb{M}_K^O$, and $\mathbb{M}_K^m \cap \mathbb{M}_K^O$. The common mesh taken from $\mathbb{M}_K^m$ focuses the mesh points near the front, while the common mesh from $\mathbb{M}_K^O$ concentrates the mesh near the line of observations. The choice on the far right ($\mathbb{M}_K^m \cap \mathbb{M}_K^O$) concentrates the mesh near both of these attributes.

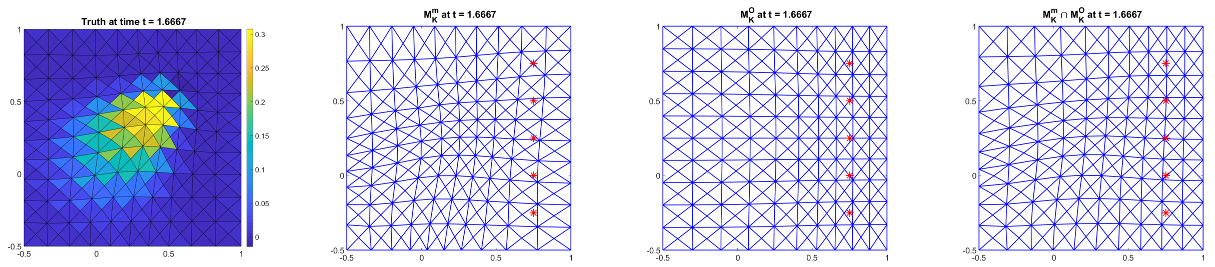

Figure 3.9: A comparison of the different choices of common mesh.

3.3.1 Experiment 5: Tuning localization and inflation

Like in the 1-dimensional case, the localization parameters for the various localization schemes are tuned simultaneously with the inflation parameter. Here we consider a reduced parameter space, using an inflation factor $\rho \in \{1, 1.05, 1.1\}$ and localization parameter $L \in \{0.5, 1.0\}$. The results are presented in Figure 3.10.

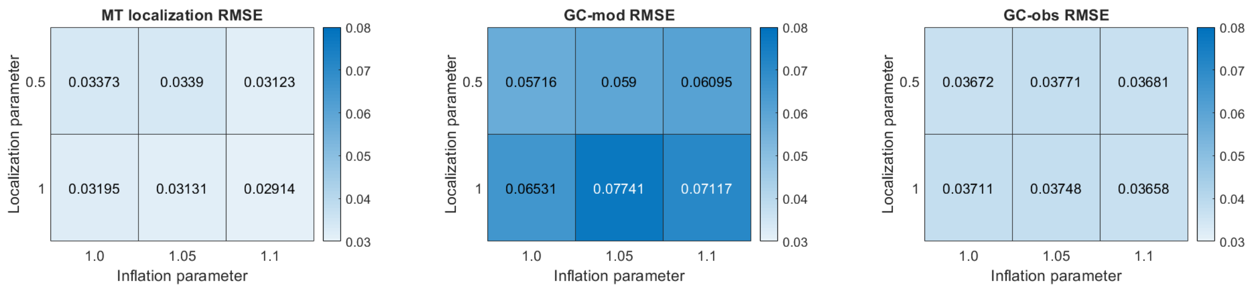

Figure 3.10: Results for simultaneous tuning of localization and inflation for the MT, GC-mod, and GC-obs localization schemes.

Just like in the 1D case, the MT localization scheme is robust with respect to the tuning of the localization parameter.

3.3.2 Experiment 6: Localization scheme comparison

Consider the following localization schemes, tuned in Experiment 5:

- MT localization with multiplicative inflation parameter 1.1 and localization parameter 1.0.
- GC-mod localization with multiplicative inflation parameter 1.0 and localization parameter 0.5.

- GC-obs localization with multiplicative inflation parameter 1.1 and localization parameter 1.

The time series RMSE of each of these localization schemes in shown in Figure 3.11. In these experiments, $\Sigma = 0.01I$, so the localization scheme is considered a success if it is on the order of 0.1.

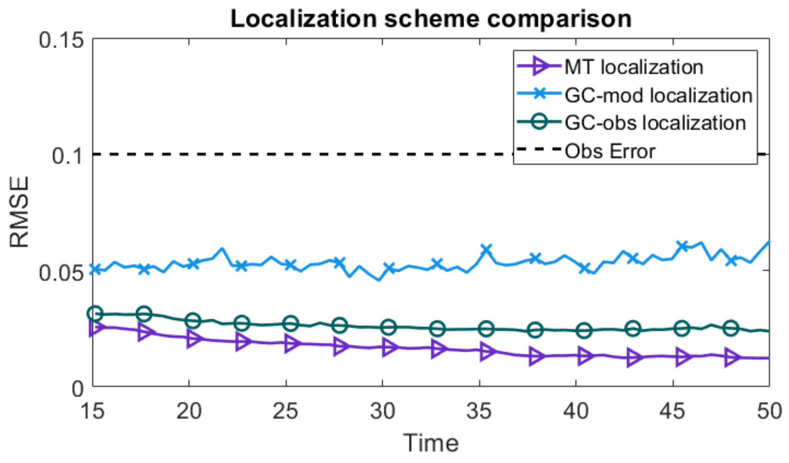

Figure 3.11: Time series RMSE comparison for the different localization schemes for 2D inviscid Burgers. Each trial uses the optimal localization parameters that were tuned in Experiment 5.

3.3.3 Experiment 7: Choice of common mesh

Again we compare the choice of the common mesh for the tuned MT localization scheme: $\mathbb{M} \in \{\mathbb{M}^O, \mathbb{M}^m, \mathbb{M}^m \cap \mathbb{M}^O\}$. A comparison of these common meshes is shown in Figure 3.12, using the tuned localization and inflation parameters found in Experiment 5. All choices for the common mesh yield a stable DA procedure.

3.3.4 Experiment 8: Comparison of error covariances

Using the tuned MT localization scheme from Experiment 5, we test the robustness of this DA procedure on the 2D inviscid Burgers (3.4) with different error covariances. For each experiment, we set the model error, observation error, and initial error covariances equal to a scalar multiple of the identity. The results in Figure 3.13 show stable RMSE.

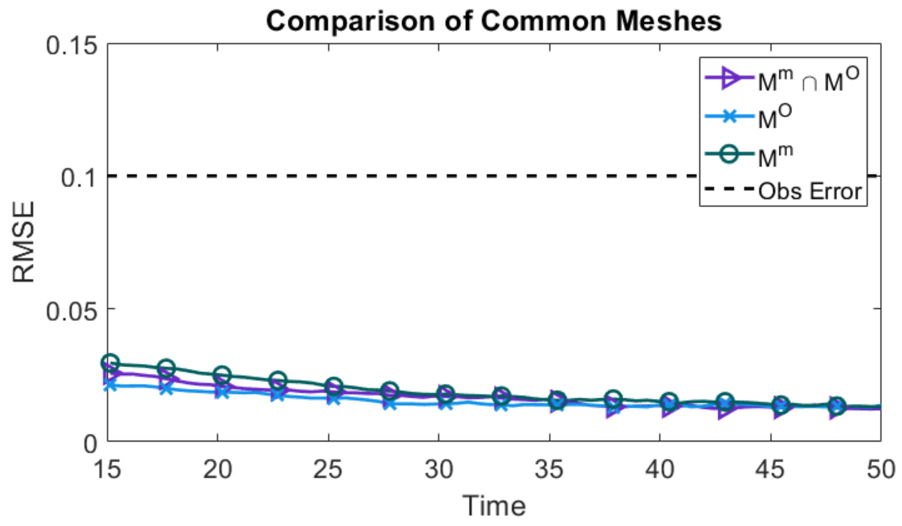

Figure 3.12: Time series RMSE for tuned DA with 2D inviscid Burgers equation using different choices for the common mesh.

3.3.5 Experiment 9: 2D Burgers with Noisy Data

To test this configuration in a more difficult regime, consider (3.4) with noisy data. That is, suppose the data error covariance is believed to be $R = 0.01I$, but the data is actually sampled with an error covariance of $R_{truth} = \alpha^2 I$, where $\alpha \geq 0.1$. For large values of α, (e.g., $\alpha \geq 100$), the analysis update causes discontinuities in the numerical solution that the time integrator cannot overcome. For moderately large values of α (e.g., $\alpha = 50$) this is sometimes an issue, but often is not. We have found that our approach works consistently for $\alpha \leq 30$, and produces a stable DA algorithm for these values. Note that in these experiments, the initial perturbation and the model error continue to be sampled from $\Sigma = 0.01I$. The results for an observational time-step of $\Delta t = 0.5$ are shown in Figure 3.14.

3.3.6 Experiment 10: Choice of Interpolation

In all of the experiments above, we use the DG interpolation, as detailed in Section 2.3.2. There is not much difference in accuracy in the 1D case, but the choice of interpolation makes a significant difference in the 2D case. Figure 3.15 shows the difference in RMSE when using the linear interpolation instead of the DG interpolation.

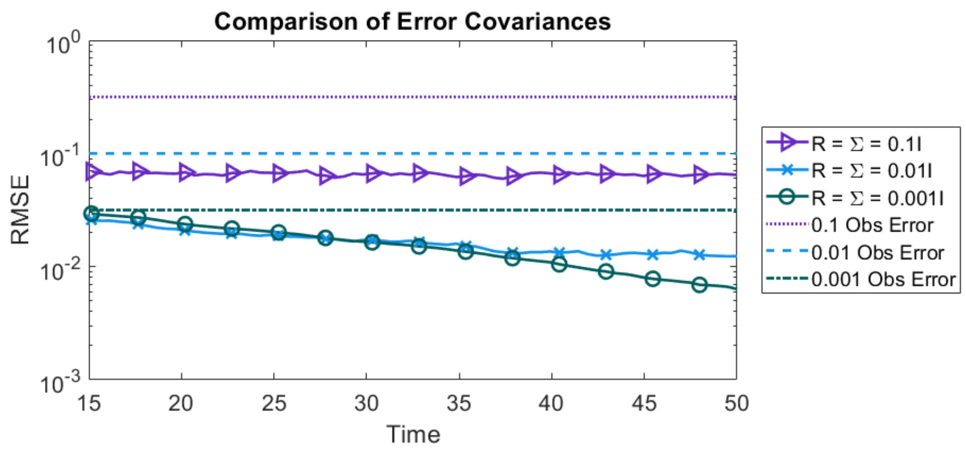

Figure 3.13: Time series RMSE for 2D Burgers with tuned MT localization and different error covariances.

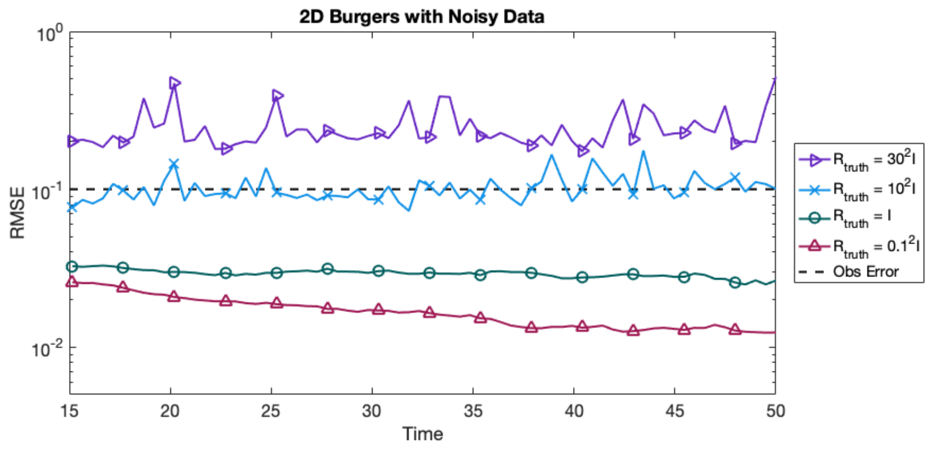

Figure 3.14: Time series RMSE for 2D Burgers with noisy data. It is assumed that the observation error has a normal distribution with mean 0 and i covariance matrix assumed to be $R = 0.01I$, but the synthetic observations are produced from the truth using an error covariance matrix R_{truth}.

‘

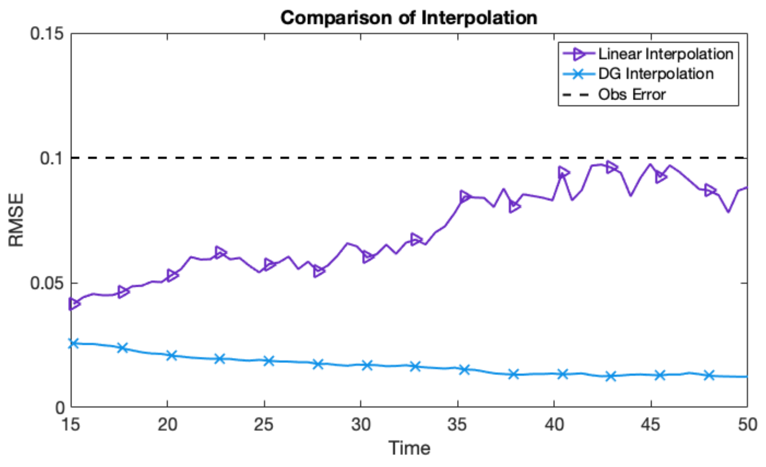

Figure 3.15: Linear interpolation vs. DG interpolation for DG discretization of 2D inviscid Burgers.

Chapter 4

Development of Single Ensemble Mesh

Abstract

In [17], we developed a method of combining an ensemble-based data assimilation (DA) scheme with an adaptive moving mesh. In this method, each ensemble member evolved on its own independent adaptive moving mesh. At each observational time-step, the ensemble members were interpolated to a common mesh for the DA update.

However, this process can be computationally expensive, and it relies heavily on interpolation. Here we present an alternative: an ensemble based DA scheme in which all ensemble members reside on the same adaptive moving mesh. There are two components to reducing the computational cost. First, we use a sample of ensemble members are chosen to compute the adaptive moving mesh at each observational time-step. Since we use a representative sample instead of the entire ensemble, this greatly reduces the computational cost. The second part of increasing the efficiency is that after the new mesh is determined, it is compared to the current mesh. If they differ sufficiently (as determined by some user-defined threshold), the mesh should be updated. One option is to set the mesh upon which all ensemble members reside to be the new mesh defined by the sample of ensemble members. That is the approach we take in the experiments that follow. It is worth noting, however, that the new sample mesh can be combined in some way with the current mesh (for example, by metric tensor intersection) if so desired. That would yield a mesh that works well at the beginning of the time interval due to the influence of the old mesh, as well as a mesh that works well at the next observational time-step, due to the influence of the new mesh. If it turns out that

the new sample mesh does not differ much from the current mesh, the last mesh is still used until the next observation time-step, eliminating unnecessary interpolation.

We demonstrate the efficacy of this approach with the 1D inviscid Burgers equation and various observation scenarios. There is a trade off between the number of mesh updates (which correlates directly with the computational cost) and the accuracy of the numerical solution.

4.1 Introduction

Recall that with the moving mesh PDE (MMPDE) method, we couple a time-dependent PDE to an equation that evolves the spatial mesh via a quasi-Lagrange approach where the mesh is considered to move continuously with time. Consider a general time-dependent PDE given by

$$\frac{du}{dt} = \nabla_x u \cdot \frac{dx}{dt} + u_t = \nabla_x u \cdot \frac{dx}{dt} + F(u) \equiv \mathscr{F}(u,x). \tag{4.1}$$

The equation for the mesh movement comes from a variational approach in which a cost function (the meshing functional) is minimized through a gradient flow differential equation:

$$x_t = -\frac{1}{\tau}\nabla_x \mathscr{L}(x,u) \equiv \mathscr{G}(x,u), \tag{4.2}$$

where $\tau \geq 0$ is a user-specified parameter controlling the speed of mesh movement. The MMPDE method is employed by first fixing x and then updating the solution of the PDE. The solution to (4.1), together with x, gives the mesh velocity defined by (4.2). A new mesh is obtained by integrating (4.2) in time.

In [17], we consider an ensemble-based DA scheme that uses the MMPDE method. Specifically, we implemented a Local Ensemble Transform Kalman Filter (LETKF) [15] where each ensemble member evolved on its own independent mesh. In this scenario, the iterative approach of solving the solution and then updating the mesh must be executed for each ensemble member. The computational cost of computing the mesh for each ensemble member can be expensive, depending on the number of mesh points used and the number of ensemble members in the DA scheme. (Typically, one requires that the number of ensemble members is at least the dimension of the unstable subspace for the PDE.) This cost seems especially unnecessary when one considers that for some problems there is relatively little variation in the ensemble meshes.

For example, consider the one-dimensional inviscid Burgers equation as given by

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0, \quad x \in [0,S), \quad t \in (0,T], \tag{4.3}$$

with initial condition

$$u(x,0) = \frac{1}{2} + 10\sin\left(\frac{2\pi}{S}x\right) \tag{4.4}$$

and periodic boundary conditions, with $S = 50$ and $T = 500$. We implement the LETKF with 25 ensemble members, 50 mesh points on each mesh, and 5 equally spaced fixed observations. Additionally, we consider the tuning from [17]: observation and model noise of 0.01, the MT localization scheme with parameter $L = 1$, the common mesh based on the intersection of ensemble meshes, and inflation parameter of 1.1. For this problem, the ensemble meshes exhibit very little variation between themselves. The variance is computed as

$$V = \frac{1}{N_e - 1}\sum_{i=1}^{N_e} |X_i - \mu|^2,$$

where N_e is the number of ensemble members, X_i is the vector containing the locations of the mesh points for the i^{th} ensemble member, and μ is the average of the X_i, computed as

$$\mu = \frac{1}{N_e}\sum_{i=1}^{N_e} X_i.$$

The variance V remains small throughout the time integration. Taking a spatial average, we see that it remains on the order of 10^{-4} for the majority of the experiment, and it generally decreases as the solutions experience more numerical dissipation. Figure 4.1 shows the evolution of $\bar{V}$ throughout the time integration.

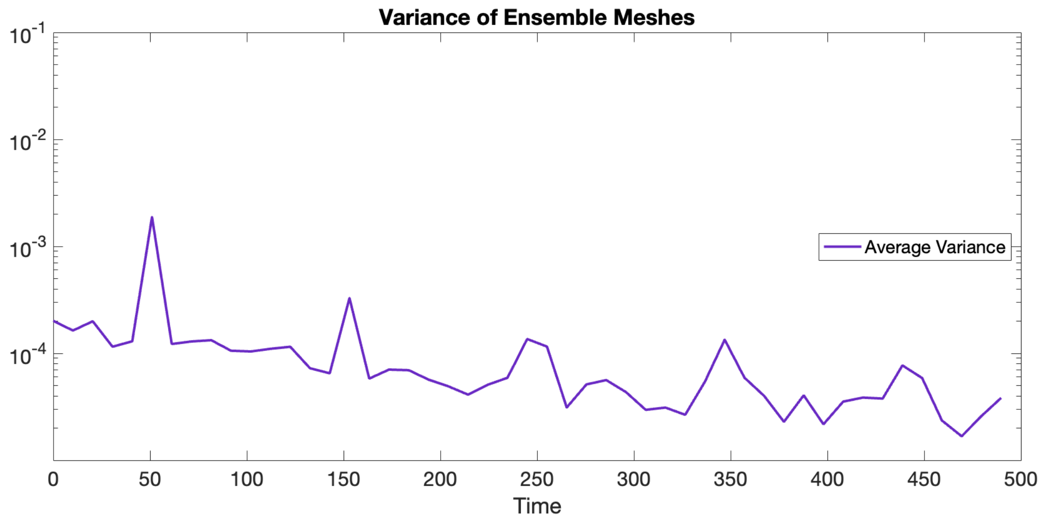

Figure 4.1: Spatially averaged variance of the ensemble meshes for the 1D inviscid Burgers equation. The ensemble meshes show very few differences, suggesting that a single mesh may suffice for all ensemble members.

4.2 The Look-Ahead Method

Since there is little variance of the ensemble meshes in problems like (4.3), we propose an adaptation to the ensemble-based DA scheme with the MMPDE method that was discussed in [17]. In this so-called "look-ahead" method, we use a single mesh for all ensemble members, and use a sample of ensemble members to determine if and when the mesh needs to be updated.

The main benefit to this method is that it reduces the computational cost. One reason is that is reduces the amount of interpolation required, since all ensemble members are residing on the same mesh. There is no need to interpolate each ensemble member to the common mesh for the DA update and then back again to the individual meshes. The second reduction in computational cost comes from the fact that only a sample of ensemble members are used to compute the mesh updates. That is, instead of creating an adaptive moving mesh for all ensemble members, we can achieve a similar result using only a fraction of the ensemble members. Integrating all of the ensemble members in time on a fixed mesh is a third cost saver, as it is much cheaper to integrate on a fixed mesh than it is to integrate on a moving mesh. This method also eliminates the potential need for remeshing, which can occur if the DA update results in ensemble meshes that are no longer suitable for the ensemble members.

The main idea of the look-ahead method is to use a subset of the ensemble members as a representative sampling of the entire ensemble of solutions. At each observational time-step, these sample ensemble members are given their own individual and independent meshes, based on each member's numerical solution. They are then pushed forward to the next observational time-step. We compute the metric tensor intersection of the metric tensors governing each of the new ensemble meshes, $\mathbb{M}^m$, and use that metric tensor for the MT localization scheme. Just like the ordinary ensemble-DA with the MMPDE scheme used in [17], we now have the choice of creating a mesh using $\mathbb{M}^m$, $\mathbb{M}^O$, or $\mathbb{M}^m \cap \mathbb{M}^O$. If the new mesh created is substantially different than the current mesh (determined by some user-defined threshold) then we update the single ensemble mesh to be the new mesh and proceed with the DA step. Otherwise, we retain the current single ensemble mesh, avoiding interpolation, and proceed on the current mesh. That is, once the user defines some threshold δ, we consider

$$\|X^{old} - X^{new}\|_2 < \delta.$$

If the above inequality holds true, we continue using $X = X^{old}$ for all ensemble members and for the common mesh. Otherwise, we interpolate all ensemble members to $X = X^{new}$ and continue with the DA update. A summary of this process is given in Algorithm 2.

4.3 Numerical Results

4.3.1 Common experimental set-up

As shown in [17], the MMPDE method works well with an ensemble- based data assimilation scheme regardless of choice of a common mesh. In the experiments that follow, we compare the efficacy and efficiency for the look-ahead method as compared to the approach used in [17]. In particular, we consider numerical results for Equation (4.3) using 25 ensemble members, 5

Algorithm 2 Look-ahead method for ensemble DA procedures on adaptive moving meshes.

```
1: procedure AT EACH OBSERVATIONAL TIME-STEP:
2:     All ensemble members currently reside on a mesh governed by the monitor function $\mathbb{M}$.
3:     Interpolate a sample of ensemble members onto individual meshes based on each member's solution.
4:     Integrate this sample of ensemble members in time to the next observational time-step with a moving mesh.
5:     Compute the new metric tensor intersection for the sample ensemble meshes, $\mathbb{M}^m$.
6:     Use $\mathbb{M}^m$ from the sample of ensemble members for the MT localization scheme.
7:     if $\|X^{old} - X^{new}\|_2 > \delta$ for some user-prescribed threshold $\delta$ then
8:         Interpolate all ensemble members to $\mathbb{M}^m$
9:         $\mathbb{M} \leftarrow \mathbb{M}^m$
10:    end if
11:    Do DA update
12:    Integrate all ensemble members to next observational time-step
13: end procedure
```

members for the look- ahead method, and 50 mesh points. We use the parameters tuned from [17]. That is, $\Sigma = R = 0.01$, the initial perturbation is 0.1, we use the MT-localization scheme with $L = 1$, and the inflation factor is given by $\rho = 1.1$.

We perform identical twin experiments where the truth is generated using no model noise, and the observations are formed by applying the observation operator and adding noise. The observation timescale is given by $\Delta t = 10$. We evaluate the efficacy of the DA procedure by computing the root mean squared error (RMSE) at each time step, where

$$RMSE = \frac{1}{\sqrt{M}} \|u^{truth} - \bar{u}\|_2, \tag{4.5}$$

with $\bar{u}$ being the analysis mean.

We consider three different observation scenarios with an observation time-step of $\Delta t = 10$. Note that while this is similar to the experimental setup in [17], the observation timescale is much larger to exacerbate the difference in meshes between observation time-steps. In order to increase the size of the observation time-step while maintaining a solution that integrates for numerous time-steps before numerically dissipating, we also adjust the spatial domain and the initial condition compared to the experiments in [17].

In the first observation scenario, we use 5 observations spaced evenly throughout the spatial domain. Since the observation locations are fixed, $\mathbb{M}^O$ will not change during the time integration. Therefore, we consider only two choices of common mesh: $\mathbb{M}^m$ and $\mathbb{M}^m \cap \mathbb{M}^O$. The second observation scenario uses only two observation points, located at the top and bottom of the shock. These observations move with the numerical solution, so that the top and bottom of the shock are always observed. The third observation scenario also uses two observations that move at the same speed as the shock, but they are positioned far away from the shock; that is, in this third scenario, the shock itself is never observed. The summary of experiments is given in Table 4.1.

Experiment	**Observation Scenario**	**Mesh Choice**
1	Fixed Obs	$\mathbb{M}^m$
2	Fixed Obs	$\mathbb{M}^m \cap \mathbb{M}^O$
3	Obs at Shock	$\mathbb{M}^m$
4	Obs at Shock	$\mathbb{M}^O$
5	Obs at Shock	$\mathbb{M}^m \cap \mathbb{M}^O$
6	Obs away from Shock	$\mathbb{M}^m$
7	Obs away from Shock	$\mathbb{M}^O$
8	Obs away from Shock	$\mathbb{M}^m \cap \mathbb{M}^O$

Table 4.1: Summary of experiments.

4.3.2 Observation Scenario 1: Fixed Observations

We begin by looking at the observation scenario where we have 5 observations whose locations are fixed in time and are equally distributed throughout the spatial domain. To put the results into context, first consider the baseline results obtained through the methods used in [17]. That is, in Figure 4.2, we use the tuning parameters as in [17] and we let each ensemble member evolve on its own mesh. The results for the various choices of common mesh ($\mathbb{M}^m$, $\mathbb{M}^O$, and $\mathbb{M}^m \cap \mathbb{M}^O$) are shown in Figure 4.2. These results serve as a baseline: it is our goal to achieve similar results via the look-ahead method, which is computationally cheaper.

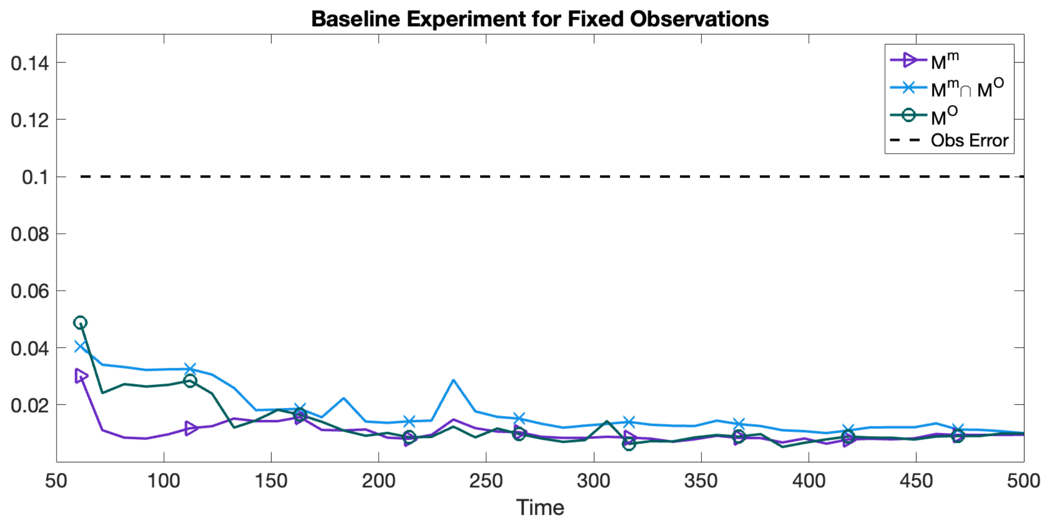

Figure 4.2: The MMPDE method works well with an ensemble Kalman filter. The results above are taken from an experiment with 25 ensemble members, 5 fixed observations, and parameters chosen from the tuning experiments in [17].

4.3.2.1 Experiment 1: Look-ahead Method with Fixed Observations and $\mathbb{M}^m$.

We repeat the baseline experiment, but instead of letting all 25 ensemble members evolve on their own independent meshes, we use 5 sample ensemble members to create a mesh upon which all ensemble members will evolve. Note that since we are considering fixed observations in this first observation scenario, the observation mesh will not ever change; that is, this is essentially just doing DA on a fixed mesh. Therefore, we do not consider $\mathbb{M}^O$ in this observation scenario.

In particular, we look at the DA results for various update tolerances δ. Recall that if the look-ahead method produces a mesh that is significantly different than the current mesh (given by a user-prescribed tolerance), the mesh will be updated. Otherwise, the current mesh will remain the same. Generally speaking, the larger the tolerance is, the fewer times the mesh will need to be updated. This results in a trade off between computational efficiency due to reduced mesh computations and potentially having a mesh that does not sufficiently resolve the numerical solution over the entire observation time interval.

As shown in Figure 4.3, the look-ahead method produces accurate results regardless of the tolerance chosen.

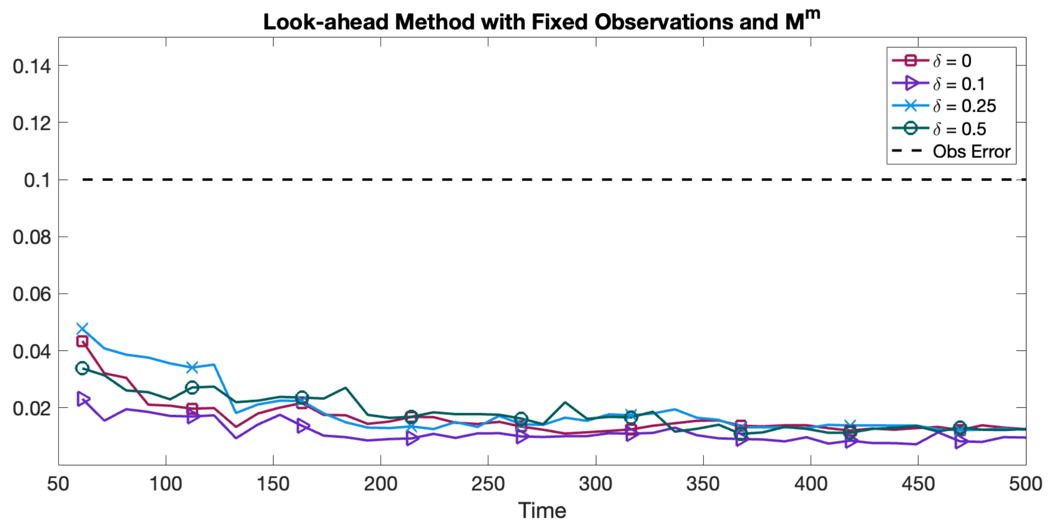

Figure 4.3: Look-ahead method for the 1D inviscid Burgers equation with 25 ensemble members and 5 sample representatives used to create a mesh via $\mathbb{M}^m$. We consider several tolerances δ for the mesh update.

4.3.2.2 Experiment 2: Look-ahead Method with Fixed Observations and $\mathbb{M}^m \cap \mathbb{M}^O$.

Here we repeat Experiment 1, but we use the 5 sample ensemble members to create a common mesh via $\mathbb{M}^m \cap \mathbb{M}^O$. The results are similar to those obtained with $\mathbb{M}^m$, as shown in Figure 4.4.

Given that the observations are fixed in this scenario, once the solution dissipates, there is little need to update the mesh. The benefit to the look-ahead method in this case is that we can achieve very accurate results with much fewer mesh updates. The experiments with a lower mesh tolerance will be updated more frequently, resulting in a longer computation time. However, as shown in Figures 4.3 and 4.4, we can obtain results that are competitive with the baseline experiment (Figure 4.2) with even higher tolerances δ. In fact, as we increase δ, we see barely any difference in the RMSE, yet there is a drastic difference in the number of mesh updates. This allows us to get similar results to the baseline experiment that used an independent mesh for each ensemble member in just a fraction of the time, and without unnecessary interpolation. We also see that for this observation scenario, there is very little difference in RMSE between using a mesh formed by $\mathbb{M}^m$ or a mesh formed by $\mathbb{M}^m \cap \mathbb{M}^O$. Recall that in this observation scenario, since the observations are fixed, $\mathbb{M}^O$ will not update, so we exclude that experiment. See Table 4.2 for details.

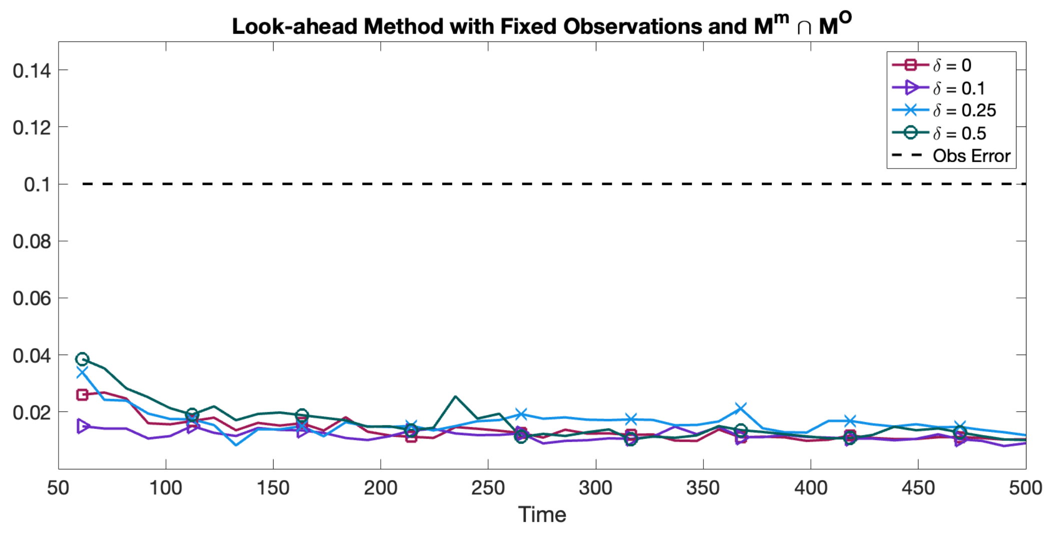

Figure 4.4: Look-ahead method for the 1D inviscid Burgers equation with 25 ensemble members and 5 sample representatives used to create a mesh via $\mathbb{M}^m \cap \mathbb{M}^O$. We consider several tolerances δ for the mesh update.

Common Mesh	Mesh Tolerance	# Updates	Average RMSE
$\mathbb{M}^m$	0.0	49	0.0185
$\mathbb{M}^m$	0.1	39	0.0148
$\mathbb{M}^m$	0.25	20	0.0217
$\mathbb{M}^m$	0.5	11	0.0200
$\mathbb{M}^m \cap \mathbb{M}^O$	0.0	49	0.0154
$\mathbb{M}^m \cap \mathbb{M}^O$	0.1	49	0.0145
$\mathbb{M}^m \cap \mathbb{M}^O$	0.25	18	0.0178
$\mathbb{M}^m \cap \mathbb{M}^O$	0.5	9	0.0181

Table 4.2: Number of updates for mesh based on mesh tolerance and mesh type.

4.3.3 Observation Scenario 2: Moving Observations with Shock

In the next set of experiments, we consider an ensemble of 25 numerical solutions to (4.3), but now we have two observations that move with the shock instead of five fixed observations. Note that since the observations are moving in this scenario, the mesh given by $\mathbb{M}^O$ is no longer a fixed mesh. Therefore, we consider three possibilities for the single mesh for all ensemble members: the meshes generated by $\mathbb{M}^m$, $\mathbb{M}^O$, and $\mathbb{M}^m \cap \mathbb{M}^O$, respectively.

Again, to put these results into context, we begin by looking at a baseline result obtained from the methods in [17] where we have 25 ensemble members all evolving on their own independent meshes and interpolating to a common mesh at each observational time-step. We look at a common mesh obtained from $\mathbb{M}^m$, $\mathbb{M}^O$, and $\mathbb{M}^m \cap \mathbb{M}^O$. Just like the first set of experiments, we use the tuning parameters from [17] and the MT-localization scheme. The results of this baseline experiment are shown in Figure 4.5.

This observation scenario is more challenging than the first one in some sense, because we are only observing two locations instead of five. However, we are still able to achieve good results, in part because the location of the observations is right at the shock.

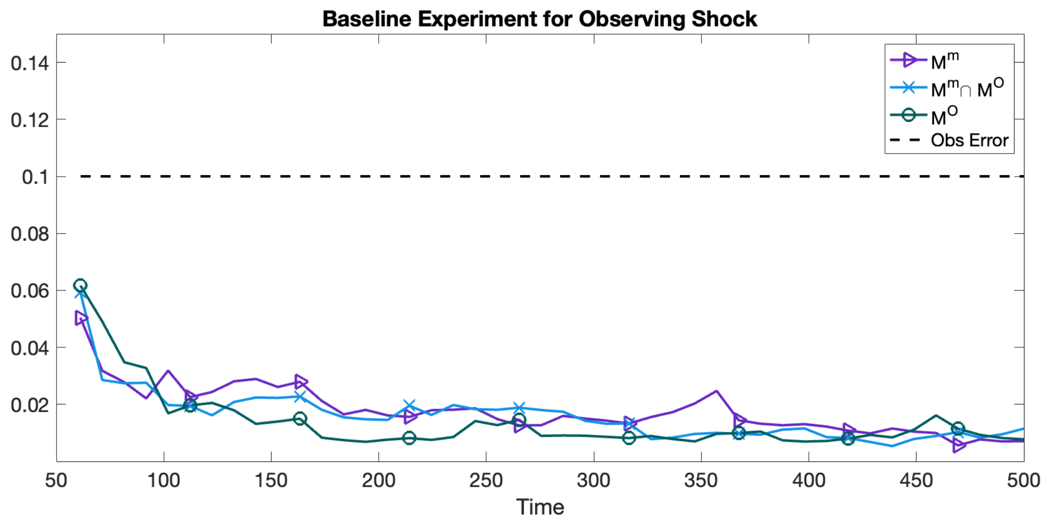

Figure 4.5: The MMPDE method works well with an ensemble Kalman filter. The results above are taken from an experiment with 25 ensemble members, 2 observations that move with the shock, and parameters chosen from the tuning experiments in [17].

4.3.3.1 Experiment 3: Look-ahead Method Observing Shock with $\mathbb{M}^m$.

We begin by implementing the look-ahead method using 5 ensemble members to compute the mesh $\mathbb{M}^m$ and letting all ensemble members evolve on $\mathbb{M}^m$. We consider various choices of δ. For the most part, we see that regardless of choice of δ, we still achieve results that are similar to the baseline experiment where each ensemble member evolved on its own mesh.

However, unlike the first observation scenario where all choices of δ provided accurate results, we do start to see a decrease in accuracy, particularly early on in the time integration, where the choice of δ does make a difference. Though the RMSE results would all suggest that this method yields a stable DA algorithm - that is, the RMSE is on the order of the observation error - we do see a considerable difference between $\delta = 0.5$ and smaller choices for δ.

The results are shown in Figure 4.6.

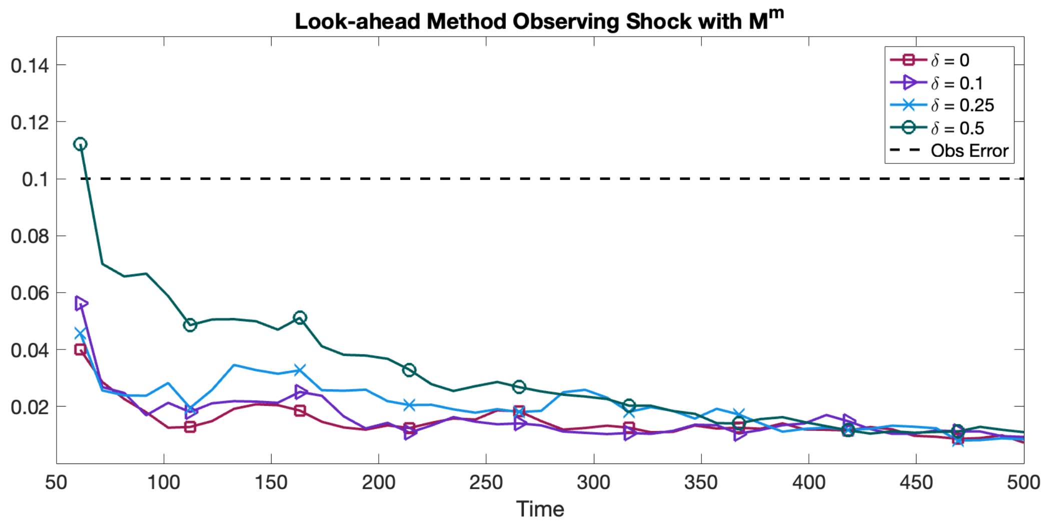

Figure 4.6: Look-ahead method for the 1D inviscid Burgers equation with 25 ensemble members and 5 sample representatives used to create a mesh via $\mathbb{M}^m$. We consider several tolerances δ for the mesh update.

4.3.3.2 Experiment 4: Look-ahead Method Observing Shock with $\mathbb{M}^O$.

As previously mentioned, in this observation scenario we have moving observations, unlike the first set of experiments. This results in a time-dependent mesh $\mathbb{M}^O$, so we include a set of experiments using $\mathbb{M}^O$ as the mesh for all ensemble members.

Even though we have extended the observation time scale to be long enough that we see significant differences in the location of the shock, we see that we still need to use significantly larger δ values in order to see a reduction in the number of mesh updates. The RMSE results are shown in Figure 4.7, and the number of updates is given in Table 4.3.

4.3.3.3 Experiment 5: Look-ahead Method Observing Shock with $\mathbb{M}^m \cap \mathbb{M}^O$.

Again, we consider numerical solutions to (4.3) using 25 ensemble members and 5 of them as a sample representative to compute a single mesh for all ensemble members to use. This time,

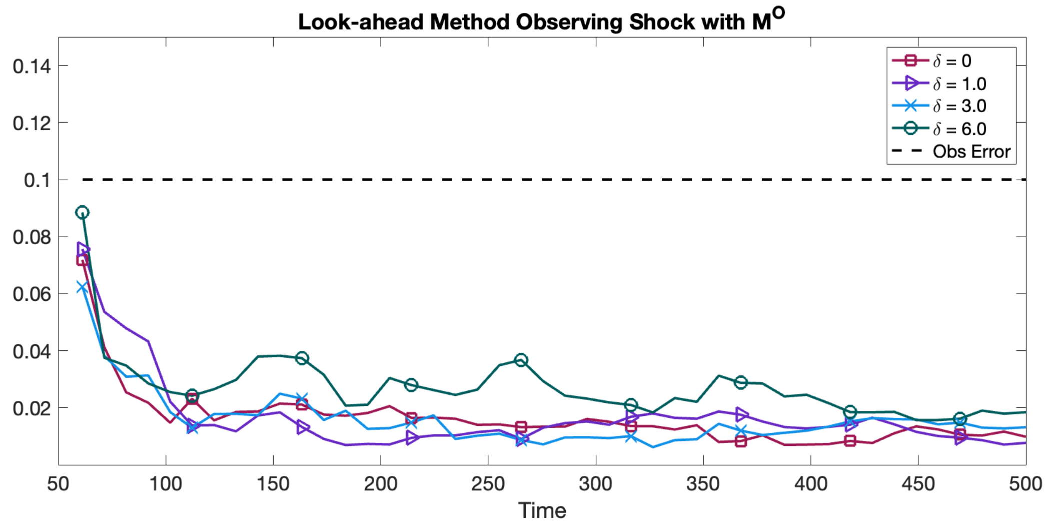

Figure 4.7: Look-ahead method for the 1D inviscid Burgers equation with 25 ensemble members and 5 sample representatives used to create a mesh via $\mathbb{M}^O$. We consider several tolerances δ for the mesh update, and note that they are larger than when we used the mesh given by $\mathbb{M}^m$.

however, we use $\mathbb{M}^m \cap \mathbb{M}^O$ as the metric tensor to compute the mesh.

Similar to Experiment 4, using $\mathbb{M}^m \cap \mathbb{M}^O$ as the mesh results in needing to choose a larger δ in order to see a reduction in the number of updates. We are still able to achieve RMSE values that are on the order of the observation error, even for the larger δ values, but the RMSE is larger than baseline experiments, at least at first. The RMSE results are shown in Figure 4.8, and the number of updates is listed in Table 4.3.

Common Mesh	Mesh Tolerance	# Updates	Average RMSE
$\mathbb{M}^m$	0.0	49	0.0160
$\mathbb{M}^m$	0.1	35	0.0177
$\mathbb{M}^m$	0.25	23	0.0215
$\mathbb{M}^m$	0.5	11	0.0376
$\mathbb{M}^m \cap \mathbb{M}^O$	0.0	49	0.0168
$\mathbb{M}^m \cap \mathbb{M}^O$	1.0	49	0.0175
$\mathbb{M}^m \cap \mathbb{M}^O$	3.0	42	0.0209
$\mathbb{M}^m \cap \mathbb{M}^O$	5.0	25	0.0338
$\mathbb{M}^O$	0.0	49	0.0213
$\mathbb{M}^O$	1.0	49	0.0221
$\mathbb{M}^O$	3.0	46	0.0203
$\mathbb{M}^O$	6.0	31	0.0305

Table 4.3: Time-averaged RMSE and number of mesh updates for observing the shock in the 1D inviscid Burgers equation, using the look-ahead method.

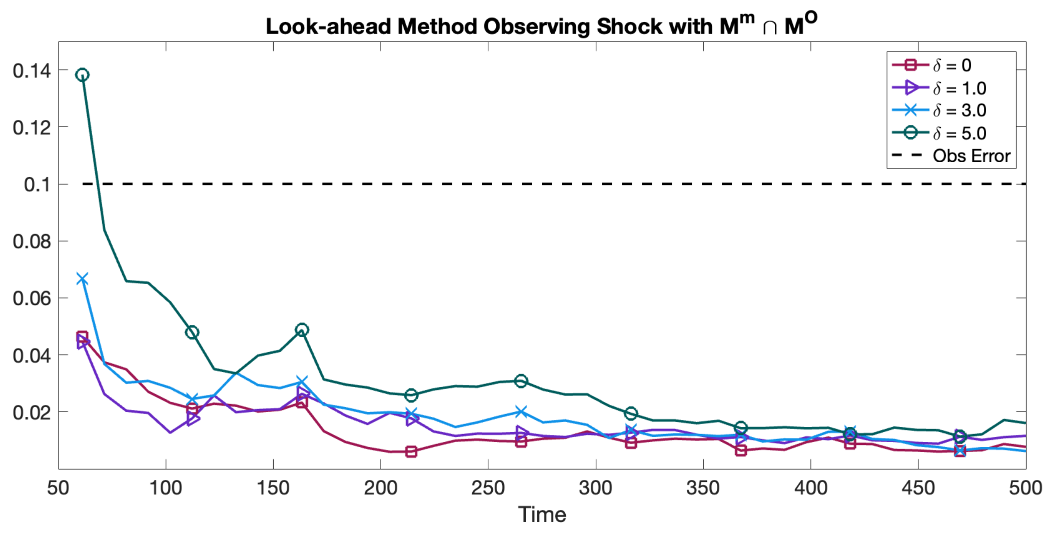

Figure 4.8: Look-ahead method for the 1D inviscid Burgers equation with 25 ensemble members and 5 sample representatives used to create a mesh via $\mathbb{M}^m \cap \mathbb{M}^O$. We consider several tolerances δ for the mesh update, and note that they are larger than when we used the mesh given by $\mathbb{M}^m$.

Note that there in order to see a reduction in the number of mesh updates, and therefore a computational advantage, δ must be significantly larger than it was with either $\mathbb{M}^m$ or any of the meshes in the fixed observation scenario. Asymptotically, the RMSE still decreases to the baseline level, even with the larger δ values. However, there is a considerable difference at the beginning of the time integration.

4.3.4 Observation Scenario 3: Moving Observations Away From Shock

Here we consider a difficult observation scenario: we have two observations that move at the same speed as the shock, but are located far away from the shock. The baseline for this experiment (25 ensemble members using the methods from [17]) is shown in Figure 4.9.

Similar to the other two observation scenarios, we now compare the look-ahead method with various mesh choices and δ tolerances to these baseline results.

4.3.4.1 Experiment 6: Look-ahead Method Not Observing Shock with $\mathbb{M}^m$.

We use 25 ensemble members overall, and again take 5 of them as a representative sample to compute a mesh upon which all 25 members will reside. The only difference between this exper-

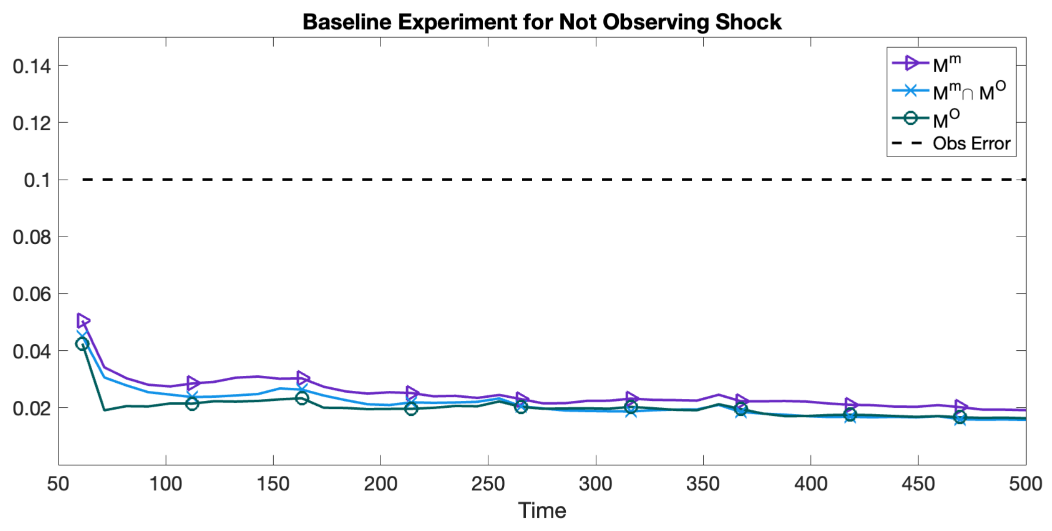

Figure 4.9: Baseline experiment for 1D Burgers with 25 ensemble members where the observations move at the same speed of the shock, far away from the shock.

iment and Experiment 3 is that the observations are taken far away from the shock, whereas in Experiment 3, the shock was observed directly.

We compare RMSE values for different choices of δ. The results are shown in Figure 4.10. In particular, since we are not observing the shock, we see that we get a comparatively bad RMSE for large δ values. Even with the challenging observation scenario, however, we do still have a successful DA procedure.

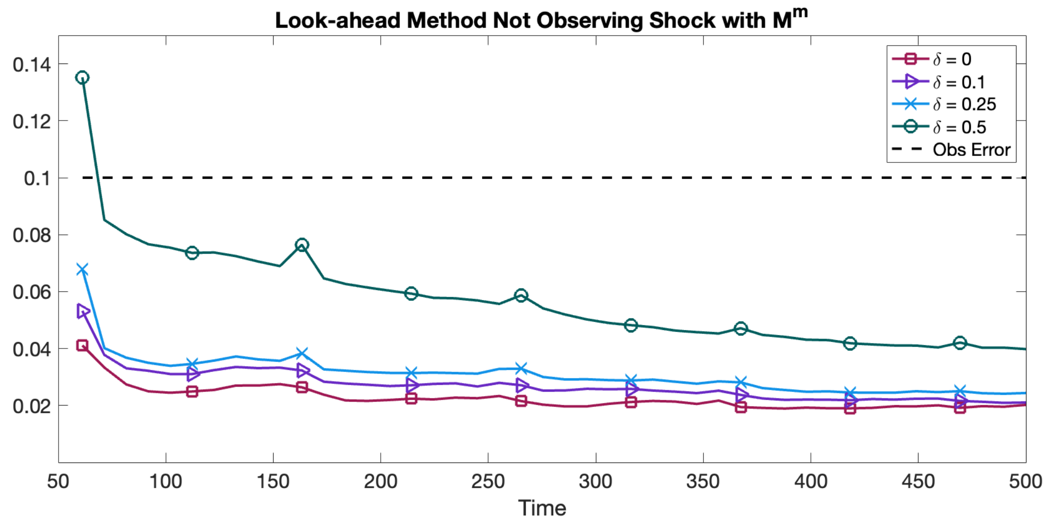

Figure 4.10: Baseline experiment for 1D Burgers with 25 ensemble members where the observations move at the same speed of the shock, far away from the shock.

4.3.4.2 Experiment 7: Look-ahead Method Not Observing Shock with $\mathbb{M}^O$.

When we repeat Experiment 6 with $\mathbb{M}^O$, we see the benefit of concentrating the mesh near the observation locations. That is, our RMSE is lower, even for high values of δ, than in Experiment 6. However, we need to use much higher values of δ in order to lower the number of mesh updates. See Table 4.4. The RMSE results are shown in Figure 4.11.

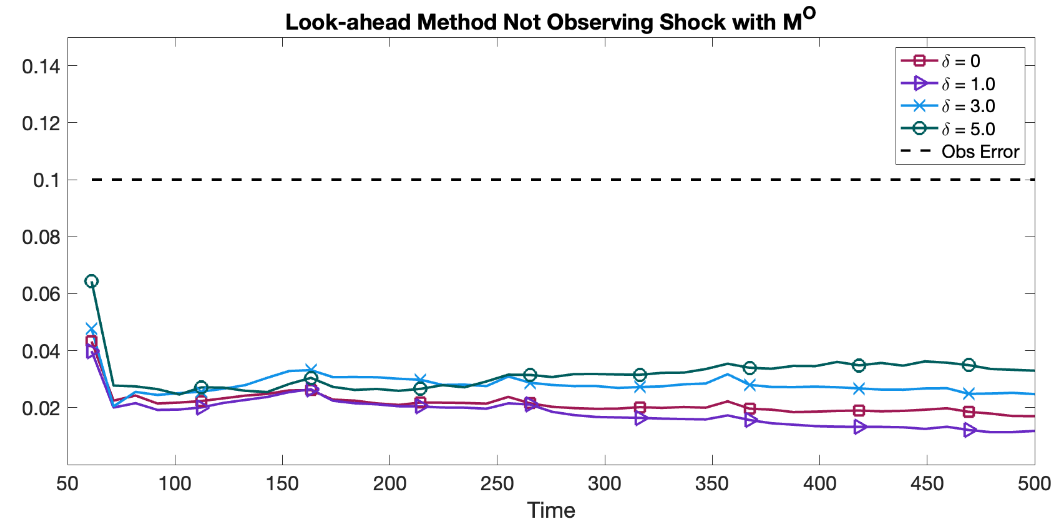

Figure 4.11: Baseline experiment for 1D Burgers with 25 ensemble members where the observations move at the same speed of the shock, far away from the shock.

4.3.4.3 Experiment 8: Look-ahead Method Not Observing Shock with $\mathbb{M}^m \cap \mathbb{M}^O$.

Finally, we repeat the experiment with 25 ensemble members and 5 representative samples using $\mathbb{M}^m \cap \mathbb{M}^O$ as the mesh. As one might predict, we see RMSE values that are in-between those of Experiment 6 and Experiment 7. Again, we have to update our mesh many more times, even with higher values of δ, as compared to the previous experiments. The results are shown in Figure 4.12, and the table of updates for Experiments 6, 7, and 8 is given in Table 4.4.

4.4 Summary

While the approach in [17] yields accurate results, it has each ensemble member evolve on its own independent mesh, and then interpolates to a common mesh at each observational time-step.

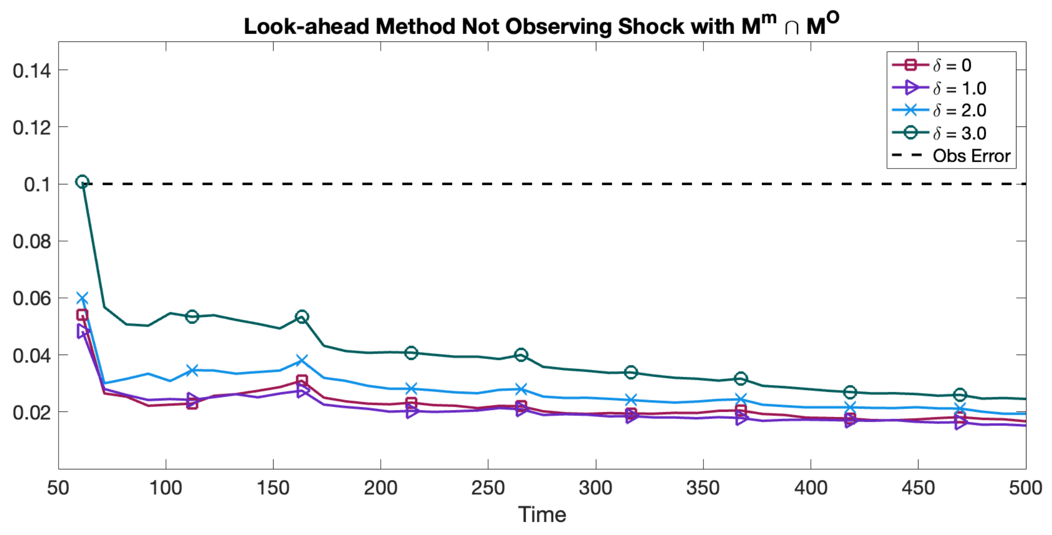

Figure 4.12: Baseline experiment for 1D Burgers with 25 ensemble members where the observations move at the same speed of the shock, far away from the shock.

Common Mesh	Mesh Tolerance	# Updates	Average RMSE
$\mathbb{M}^m$	0.0	49	0.0234
$\mathbb{M}^m$	0.1	36	0.0278
$\mathbb{M}^m$	0.25	22	0.0320
$\mathbb{M}^m$	0.5	13	0.0617
$\mathbb{M}^m \cap \mathbb{M}^O$	0.0	49	0.0229
$\mathbb{M}^m \cap \mathbb{M}^O$	1.0	49	0.0221
$\mathbb{M}^m \cap \mathbb{M}^O$	2.0	44	0.0279
$\mathbb{M}^m \cap \mathbb{M}^O$	3.0	29	0.0397
$\mathbb{M}^O$	0.0	49	0.0226
$\mathbb{M}^O$	1.0	49	0.0195
$\mathbb{M}^O$	3.0	45	0.0283
$\mathbb{M}^O$	5.0	30	0.0325

Table 4.4: Number of updates for mesh based on mesh tolerance and mesh type.

This can be computationally expensive. For certain problems, such as 1D inviscid Burgers, there actually is very little variation in the individual ensemble meshes. This suggests that a single adaptive mesh might be used for all ensemble meshes, reducing the amount of computation and also eliminating the need for interpolating to a common mesh for each DA step.

Additionally, if the mesh is sufficient over a long period of time, there is no need to update the mesh at each time-step. To monitor this, a user-determined threshold δ is chosen. A representative selection of ensemble members is used to compute the mesh at each observational time-step. If norm of the difference between the new mesh and the old mesh is greater than δ, the new mesh becomes the mesh for all ensemble members. Otherwise, we leave the current mesh as it is and avoid unnecessary interpolation.

Choosing δ larger results in fewer mesh updates, thereby reducing the computational cost. This is especially impactful in Experiments 1 and 2, where even a small increase in δ results in a many fewer mesh updates. However, with fewer mesh updates, we run the risk of having a mesh that does not sufficiently resolve the mesh near sharp interfaces. There is a corresponding increase in RMSE for larger δ, especially in certain observation scenarios, and when the shock is relatively steep. In Experiments 3, 4, and 5, for example, we need to use a much larger δ for the mesh formed from $\mathbb{M}^O$ and $\mathbb{M}^m \cap \mathbb{M}^O$ in order to see a decrease in the number of mesh updates, as compared to using $\mathbb{M}^m$ to form the mesh, and we also see an increase in RMSE. This results in a trade off: depending on the problem and the observation scenario, a smaller δ may result in more frequent mesh updates and more accurate solutions, while a larger δ may lower the computational cost at the expense of a higher RMSE.

The main advantage of this method is the computational efficiency achieved through the look-ahead method and reducing the number of mesh updates. We are able to achieve similar RMSE with decreased computational cost. Even if we update the mesh at every time-step, using a sample of ensemble members to do so instead of the full ensemble cuts that computation time by whatever fraction we are choosing for the representative sample. In the experiments presented above, we used 5 representative ensemble members instead of all 25, so the cost of computing the mesh is $\frac{1}{5}$ of what it would have been using all ensemble members. In addition, having a scheme where the mesh does not update every time also reduces the computation time. Furthermore, the choice of integrating the ensemble members forward in time on a fixed mesh instead of on a moving mesh also decreases the computation time for the time integration.

There are a few considerations one must take into account for this approach. One question is how many ensemble members to use as the representative sample for forming the mesh. Through experimental tuning, we saw that 5 ensemble members were sufficient for forming the mesh for the ensemble of 25 solutions to (4.3). The choice of δ must also be tuned for each application and observation scenario. As shown in Experiments 1 and 2 versus Experiments 3, 4, and 5, the values of δ that work in one situation may have little to no impact on the number of mesh updates in a different scenario. With careful tuning, however, these issues can be resolved to produce a stable DA scheme that is both accurate and computationally efficient.

In the above results, we explored updating the mesh so that it would be accurate for the next observation timestep. We might also want a mesh that is accurate throughout the time interval until the next observation is taken. One way we might achieve this goal is by taking the metric tensor intersection of the old mesh and the new mesh. Doing so will concentrate mesh points near the current shock, and also near the location of the shock at the next time step. This idea is demonstrated in Figure 4.13.

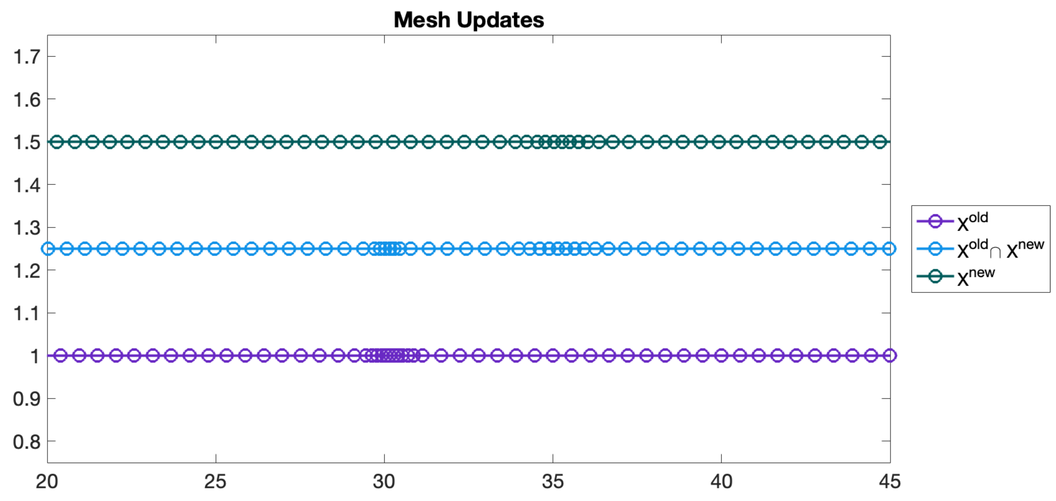

Figure 4.13: The old mesh has mesh points concentrated near the location of the shock, currently, while the new mesh concentrates the mesh points near the location of the shock at the next observation timestep. Intersecting the monitor functions that determine X^{old} and X^{new} gives us a mesh that is concentrated in both places.

Another topic that merits further investigation is how the representatives are chosen. We choose a portion of the ensemble members at random, but perhaps the choice of the ensemble members could be optimized to retain information that best reflects all of the ensemble members.

Finally, while the lookahead method works well when the ensemble meshes exhibit little variance, it remains to be seen how well this works when there are outliers. Perhaps some members should be allowed to reside on their own mesh if they are very different from the rest of the ensemble members.

Chapter 5

Goal-Oriented Metric Tensors

Abstract

We develop goal-oriented metric tensors for adaptive meshing techniques to be used in a data assimilation (DA) context. The adaptive meshes move not only in relation to the solution of the PDE, concentrating mesh points where the solution requires a finer resolution, but also with respect to the observation locations. Previously, these two goals had been addressed in an ad hoc way; here we present a method to optimally find the best meshing functional for the purpose of a successful DA procedure. We present a new method of finding goal-oriented monitor functions and discuss methods of optimizing the monitor function in order to minimize the norm of the innovation or the root mean squared error (RMSE), or some other metric that measures the skill of the DA scheme. We implement this new metric tensor with the one dimensional reaction-diffusion equation.

5.1 Introduction

As we have seen, the process of combining ensemble-based data assimilation (DA) with adaptive moving meshes is nontrivial. The computations in ensemble-based DA techniques implicitly assume that the ensemble members all reside on the same mesh. However, if the ensemble members are allowed to evolve on their own independent meshes, we must take care when combining the ensemble values. Consider an ensemble-based DA scheme where an ensemble of solutions is used to approximate the true solution and its uncertainty. A common method of using adaptive moving meshes with an ensemble-based DA procedure is to have a common mesh upon which the DA update is performed [6, 2, 26, 17]. There are various choices for this common mesh, the simplest being a fixed, uniform mesh. However, one may wish to incorporate the features of the individual meshes so that the common mesh adequately supports each ensemble member.

Another concern when implementing DA with an adaptive mesh is how to concentrate the common mesh near observation locations so as to reduce the need for interpolating observations. However, addressing either of these concerns at a fixed time may result in a common mesh that may not be suitable at a different time. For example, the ensemble solutions may need finer resolution in a different part of the spatial domain as time progresses, or, in the case of Lagrangian observations, the observation locations might also change in time. For these situations, we consider an adaptive common mesh.

There are many ways to define mesh adaptation. We follow the approach of Huang and Russell [13] and define an adaptive mesh through the use of a user-prescribed metric tensor, where the mesh can be viewed as uniform in a given metric. That is, the elements of the mesh have unit volume and are similar to a regular element under the given metric. This metric is defined by a positive-definite matrix valued monitor function $\mathbb{M} = \mathbb{M}(x)$. This user- prescribed metric tensor defines the independent mesh movement for each of the ensemble members, and the adaptive common mesh is then formed through the metric tensor intersection of each of the ensemble members.

Two common choices for the metric tensor are based on the arc length and the Hessian of the numerical solution. Both of these choices will yield a mesh that will sufficiently support the numerical solutions of the ensemble members, but it does not take into account the location of the observations. Here we suggest a new metric tensor that monitors mesh movement to improve the DA procedure. In this goal-oriented metric tensor framework, the mesh moves not only based on the solution, but also based on some quantity important to the DA procedure, such as proximity to observation locations. While in [17] we balanced these two objectives in an ad hoc way, here we present a variational approach to updating the adaptive moving mesh. More specifically, we wish to determine an optimal mesh for the assimilation, thereby enhancing the skill of the DA scheme.

5.2 A Variational Approach to Determining the Metric Tensor

The success of a DA procedure can be determined by a identical twin experiment, where the prediction is compared to a "truth" run. The root mean squared error (RMSE) is computed as

$$RMSE = \frac{1}{\sqrt{M}} \|u^{truth} - \bar{u}\|_2, \tag{5.1}$$

where $\bar{u}$ is the analysis mean. Ideally, we would like to choose the metric tensor $\mathbb{M}$ so that the mesh moves in a way that minimizes the RMSE. In practice, however, the truth is not known, so instead we use a proxy. We can minimize either the norm of the innovation, $\|I\|_2 = \|y - H(\bar{u})\|_2$, or the norm of the Kalman update, $\|K \cdot I\|_2$, where $K = P^f H^T \left(HP^f H^T + R\right)^{-1}$ is the Kalman gain. In the previous equation, H is the linear (or linearized) observation operator, R is the observation covariance matrix, and P^f is the forecast covariance.

This leads us to a constrained optimization problem

$$\min_x G(u,x) \quad \text{subject to} \quad \nabla_x \mathscr{L}(u,x) = 0, \tag{5.2}$$

where, e.g., $\mathscr{L}(u,x) = \sum_j \alpha_j \mathscr{L}_j(u,x)$ is a meshing functional depending on the (possibly time dependent) parameters $\{\alpha_j\}$, and $G(u,x)$ is a measure of the skill of the DA scheme.

As discussed above, possible options for $G(u,x)$ include the innovation, RMSE, or norm of the Kalman gain. In a variational DA scheme, one could also optimize over the cost function used in 3DVar or 4DVar, or the variance of a particle filter. Furthermore, the choice of functions $G(u,x)$ can be adapted in space and time to target locations of high interest.

For the meshing functional $\mathscr{L}(u,x)$, we use a sum of terms $\mathscr{L}_j(u,x)$ that take into account different factors influencing the mesh and skill of the DA. These include information about how well the ensemble members are approximated on the common mesh, error in interpolation of ensemble members, and error of interpolation of observations. Note that the condition $\nabla_x \mathscr{L}(u,x) = 0$ can either be solved as an algebraic equation or by using a numerical approximation of the gradient descent differential equation.

There are many ways to solve this optimization problem. One can simply tune the parameters $\{\alpha_j\}$ by comparing the resulting $G(u,a)$. Another option would be to solve for α using a classical method, like Lagrange multipliers. We could also solve for $\{\alpha_j\}$ in a Bayesian context, as DA is frequently used for parameter estimation. A third choice would be to use a machine learning approach to estimate the coefficients.

5.2.1 Example: Reaction Diffusion Equation

Consider the reaction diffusion equation

$$u_t = u_{xx} + g(u), \quad 0 < x < L \tag{5.3}$$

$$u_x(0,t) = u_x(L,t) = 0, \quad t > 0 \tag{5.4}$$

where $g(u)$ is some smooth bistable nonlinearity with three zeros. In the numerical experiments that follow, we take

$$g(u,a) = u(u-a)(u-1),$$

where a is a parameter that controls the location of the unstable root. We implement (5.3) with $L = 50$, Neumann boundary conditions and initial condition given by

$$u(x,0) = \frac{1}{2}\left(\tanh\left(\frac{x-25}{2\sqrt{2}}\right)+1\right). \tag{5.5}$$

Assuming a moving mesh, its discretization is given by

$$\dot{U}_j = \left[\frac{U_{j+1}-U_{j-1}}{x_{j+1}-x_{j-1}}\right]\dot{x}_j + \frac{2}{x_{j+1}-x_{j-1}}\left[\frac{U_{j-1}-U_j}{x_j-x_{j-1}} + \frac{U_{j+1}-U_j}{x_{j+1}-x_j}\right] + g(U_j). \tag{5.6}$$

For the mesh movement, we let $\mathscr{L}_1(u,x)$ be the functional that satisfies the equidistribution of arc length. For a fixed time, the arc length can be calculated as $s = \int_a^b \sqrt{1+\left(\frac{du}{dx}\right)^2}dx$. Taking the gradient and discretizing in space, the mesh movement for equidistribution of arc length is determined by (5.7):

$$\tau \dot{x}_j = \sqrt{\left(x_{j+1}-x_j\right)^2+\left(U_{j+1}-U_j\right)^2}-\sqrt{\left(x_{j-1}-x_j\right)^2+\left(U_{j-1}-U_j\right)^2}. \tag{5.7}$$

However, we also want to include a meshing functional that forces the common mesh points close to the observation locations. Let L denote the number of ensemble members, D the dimension of the observation space, and M the number of mesh points in the common mesh. Fixing time, we let $\mathbf{u_c} = \{u(x_j^c)\}_{j=1}^M$, and let I denote an interpolation operator that maps $\mathbf{u_c}$ to $\mathbf{u_o}$ where $\mathbf{u_o}$ is supported at all the observation locations $\{x_j^o\}_{j=1}^D$ (plus potentially other locations). Then we set

$$\mathscr{L}_2(x^c) = \alpha_2 \sum_{k=1}^{D} ||x_k^o - x_l^c||^2$$

where x_l^c is the mesh point in x^c closest to x_k^o or some other measure of the interpolation error due to mismatch between the common mesh locations and the observation locations.

Normalizing the equation so that $\alpha_1 = 1$, this can be written as a general minimization problem:

$$\min_x G(u,x), \qquad \text{subject to} \qquad \mathscr{L}(u,x) := \mathscr{L}_1(u,x(\alpha)) + \alpha\mathscr{L}_2(u,x(\alpha)) = 0, \tag{5.8}$$

where α is a (possibly time-dependent) parameter that controls the weight of the second constraint.

Then the new (discretized) constraint is given by

$$\mathscr{L}(u,x) = \sum_{j=1}^{M} \sqrt{1+\left(\frac{U_{j+1}-U_j}{x_{j+1}-x_j}\right)^2}\Delta x_j + \alpha \sum_{i=1}^{D} \|x_i^o - x_l^c\|^2. \tag{5.9}$$

5.2.2 Numerical Results

We employ the Local Ensemble Transform Kalman Filter (LETKF) [15] with the adaptive moving meshing scheme described above to implement (5.3) with $a = 0.6$ in the $g(u,x)$ equation. For our constraint equation, we use a combination of the equidistribution of arc length, as well as a term to monitor the closeness to the observation locations. The discretized constraint that we use in the following example is given by (5.9).

We use 5 ensemble members, observation error with covariance matrix $0.1I$, background covariance of $0.1I$, and initial perturbation covariance of $0.5I$. We do not employ any localization scheme. There are 5 observations equally spaced through the domain, and 20 mesh points. We use a integration timestep of 10^{-4}, an observation timestep of 10^{-2}, and we integrate until $t = 1$. We tune various values of α to see the effect on the innovation and the RMSE, as shown in Table 5.1.

Value of α	Norm of Innovation	RMSE
1	0.0522	0.0081
5	0.0553	0.0080
10	0.0512	0.0080
15	0.0559	0.0089
20	0.0504	0.0087
25	0.0526	0.0073

Table 5.1: Time-averaged norm of the innovation and RMSE for the reaction- diffusion equation with different values of α.

The innovation and RMSE both stay relatively small for all choices of α. The time-series for the innovation and the RMSE with $\alpha = 10$ are shown in Figures 5.1 and 5.2 respectively.

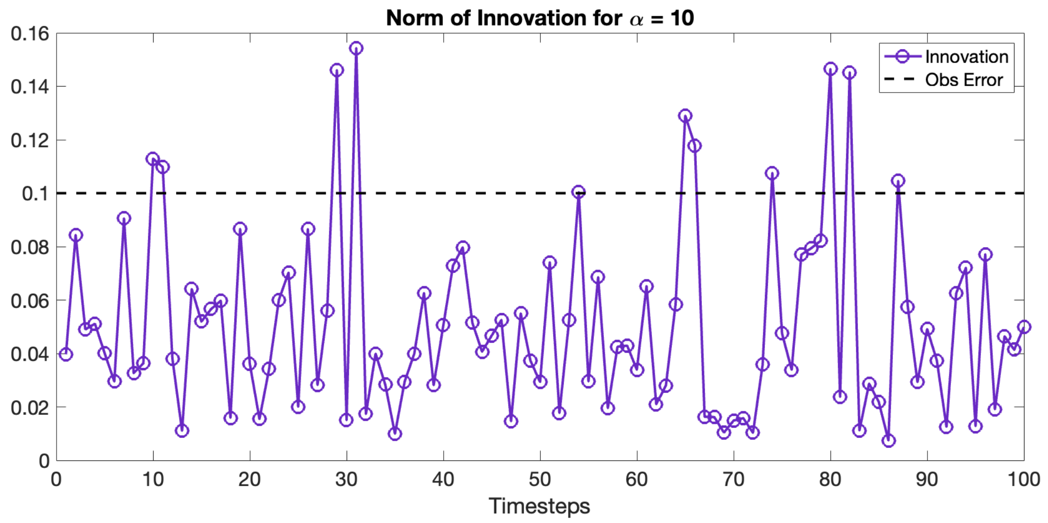

Figure 5.1: Time series for the norm of the innovation for the reaction diffusion equation with $\alpha = 10$.

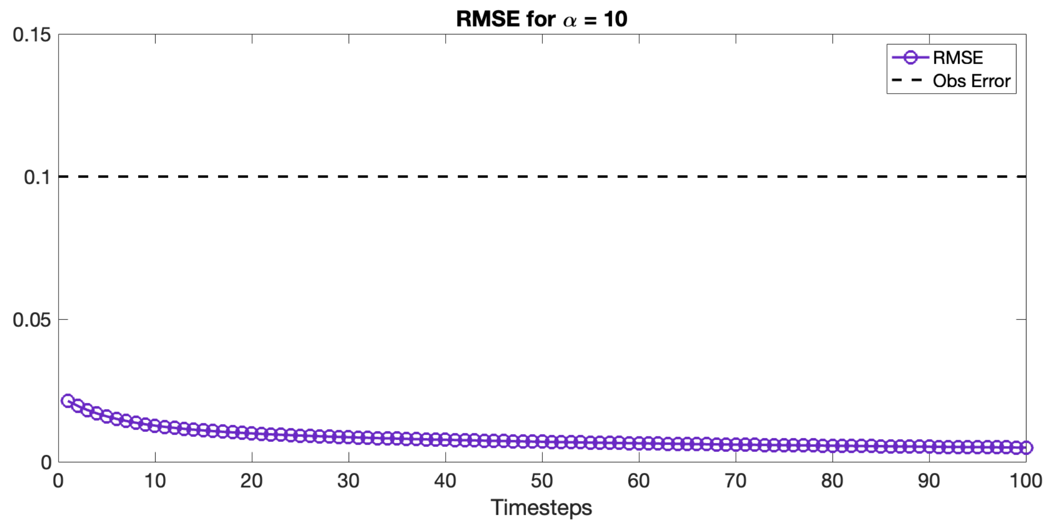

Figure 5.2: Time series for the RMSE for the reaction diffusion equation with $\alpha = 10$.

5.3 Summary

While the methods used in [17] worked well for an ensemble-based DA scheme with an adaptive moving mesh, the choice of metric tensor for the common mesh was an arbitrary one. Instead of using the metric tensor intersection and concentrating the mesh points in an ad hoc way, here we present a means to systematically choose the metric tensor that defines the common mesh in a variational setting.

Through the use of a constraint equation that balances the equidistribution of arclength, along with a constraint that penalizes for not having mesh points close to the observation location, we are able to adjust the constraint equation in order to optimize a desired value, such as the RMSE or the norm of the innovation. That is, we determine a common mesh that takes into account different facts that impact mesh location and hence the DA skill in order to optimize the efficacy of the DA scheme. In this experiment, we tune α explicitly, by looking both at the RMSE and the norm of the innovation. We do note, however, that we could also have solved the generalized optimization equation to solve for α.

Chapter 6

Conclusion

Through the use of an adaptive common mesh developed from the moving mesh PDE (MMPDE) method, we develop an ensemble based DA scheme where each of the ensemble members evolve independently on their own adaptive meshes. At each observational time-step, the ensemble members are interpolated to the adaptive common mesh, updated according to the DA scheme, and then interpolated back to their individual meshes.

We follow the MMPDE adaptive meshing strategy where the mesh of each ensemble member is determined by a matrix-valued monitor function, also called a metric tensor, so that the mesh is viewed as uniform in that metric. At each observational time-step, an adaptive common mesh is calculated. There are several choices for this common mesh. One choice, M^m, is obtained by taking the intersection of the ensemble members' metric tensors. This results in a common mesh that in some sense satisfies all of the ensemble members. Another option, $\mathbb{M}^O$, is to concentrate the common mesh near observation locations or observation trajectories. Concentrating the mesh near the observation locations reduces the amount of interpolation error at each observational time-step. A third choice is to intersect $\mathbb{M}^m$ with $\mathbb{M}^O$. Using the observational mesh $\mathbb{M}^O$ in a DA scheme reduces the transient time in converging to the asymptotic behavior, but regardless of which is employed as the common mesh, all choices $\mathbb{M}^m$, $\mathbb{M}^O$, and $\mathbb{M}^m \cap \mathbb{M}^O$ produce stable results. The efficacy of several techniques developed in this work is illustrated using sharp interface problems, in particular 1D and 2D inviscid Burgers equations, under a discontinuous Galerkin discretization.

We develop a new adaptive localization algorithm based on the metric tensor of the common mesh $\mathbb{M}^m$. The MT adaptive localization scheme uses the metric tensor to define a domain localization strategy that is dynamically updated in time and space. For the 1D and 2D inviscid Burgers equations, the MT localization scheme compared favorably with the Gaspari-Cohn (GC) localization schemes. One of the benefits of the MT localization scheme is that it is robust with respect to the tuning parameters and requires less precise tuning than GC localization in either the model space or in the observation space.

The interpolation that is used at each observational time-step can have a significant impact on the performance of the DA scheme. Using a DG discretization for the PDE together with a DG-based interpolant allows the ensemble members to maintain the advantages of DG discretization independent of their supporting mesh. For the 1D inviscid Burgers problem, there was no significant difference in RMSE between linear and DG interpolation. For the 2D inviscid Burgers equation, however, using a DG-based interpolation scheme improved the overall performance of the DA scheme as compared to linear interpolation.

This metric tensor approach to data assimilation on adaptive moving meshes, as well as the MT localization scheme, is applicable in higher spatial dimensions. There are several interesting avenues for further investigation. These include the development of adaptive meshes in which a single mesh supports all ensemble members over an observation cycle. In this approach, we use a representative sampling of the ensemble members to create a mesh (either $\mathbb{M}^m$, $\mathbb{M}^O$, or $\mathbb{M}^m \cap \mathbb{M}^O$) and the associated MT localization scheme. We use that single mesh for all of the ensemble members. At each observational time-step, we compute the representative mesh again, and if it differs sufficiently from the previous mesh (as determined by a user-defined threshold), we interpolate all ensemble members to that new mesh.

This single adaptive mesh approach has the potential to greatly reduce the computational time, achieving similar RMSE results as compared to using an independent adaptive mesh for each ensemble member. However, careful tuning is required. Choosing the threshold too small will not reduce the computational cost, as the mesh will still have to be updated frequently. On the other hand, if we choose the threshold too large, the mesh may not sufficiently resolve the ensemble members near the sharp gradients.

Finally, while the methods presented in [17] yield a stable DA algorithm for incorporating adaptive moving meshes with an ensemble based DA scheme, the way in which we define the common mesh was done arbitrarily via the metric tensor intersection of the ensemble methods and then potentially concentrated near the observation locations. As an alternative, we can create the common mesh through a metric tensor that is defined in a way to minimize the RMSE, the norm of the innovation, or some other measure of DA success. In this variational setup, we minimize the feature of interest subject to a constraint equation that consists of a linear combination of various cost equations, such as the equidistribution of arclength or the proximity of mesh points to the observation locations. Solving for the (possibly time-dependent) coefficients in the constraint equation gives us a way to systematically modify the common mesh so that we achieve the best DA results.

References

[1] M. Asch, M. Bocquet, and M. Nodet. *Data assimilation: methods, algorithms, and applications*. SIAM, 2016.

[2] A. Aydoğdu, A. Carrassi, C. Guider, C. Jones, and P. Rampal. Data assimilation using adaptive, non-conservative, moving mesh models. *Nonlin. Processes Geophys.*, 26:175–193, 2019.

[3] R. Bannister. A review of operational methods of variational and ensemble-variational data assimilation. *Quart. J. Royal Meteorological Soc.*, 143:607–633, 2017.

[4] B. Bonan, N. K. Nichols, M. J. Baines, and D. Partridge. Data assimilation for moving mesh methods with an application to ice sheet modelling. *Nonlin. Processes Geophys.*, 24:515–534, 2017.

[5] A. Carrassi, M. Bocquet, L. Bertino, and G. Evensen. Data assimilation in the geosciences: An overview on methods, issues and perspectives. *Wires Climate Change*, 9, 2018.

[6] J. Du, J. Zhu, F. Fang, C. Pain, and I. Navon. Ensemble data assimilation applied to an adaptive mesh ocean model. *Int. J. Numer. Methods Fluids*, 82:997–1009, 2016.

[7] W. Huang. Variational mesh adaptation: isotropy and equidistribution. *J. Comput. Phys.*, 174:903–924, 2001.

[8] W. Huang. Mathematical principles of anisotropic mesh adaptation. *Comm. Comput. Phys.*, 1:276–310, 2006.

[9] W. Huang and L. Kamenski. A geometric discretization and a simple implementation for variational mesh generation and adaptation. *J. Comput. Phys.*, 301:322–337, 2015.

[10] W. Huang and L. Kamenski. On the mesh nonsingularity of the moving mesh PDE method. *Math. Comp.*, 87:1887–1911, 2018.

[11] W. Huang, Y. Ren, and R. D. Russell. Moving mesh methods based on moving mesh partial differential equations. *J. Comput. Phys.*, 113:279–290, 1994.

[12] W. Huang, Y. Ren, and R. D. Russell. Moving mesh partial differential equations (MMPDES) based on the equidistribution principle. *SIAM J. Numer. Anal.*, 31:709–730, 1994.

[13] W. Huang and R. D. Russell. *Adaptive moving mesh methods*, volume 174. Springer Science & Business Media, 2010.

[14] W. Huang and W. Sun. Variational mesh adaptation II: error estimates and monitor functions. *J. Comput. Phys.*, 184:619–648, 2003.

[15] B. R. Hunt, E. J. Kostelich, and I. Szunyogh. Efficient data assimilation for spatiotemporal chaos: a local ensemble transform Kalman filter. *Physica D*, 230:112–126, 2007.

[16] E. Kalnay, H. Li, T. Miyoshi, S.-C. Yang, and J. Ballabrera-Poy. 4-D-Var or ensemble Kalman filter? *Tellus A: Dynamic Meteorology and Oceanography*, 59:758–773, 2007.

[17] C. Krause, W. Huang, D. Mechem, E. S. Van Vleck, and M. Zhang. A metric tensor approach to data assimilation with adaptive moving meshes. *J. Comput. Appl. Math., submitted*, 2021.

[18] K. J. H. Law, A. M. Stuart, and K. C. Zygalakis. *Data Assimilation: A Mathematical Introduction*, volume 62 of *Texts in Applied Mathematics*. Springer International Publishing, 2015.

[19] A. C. Lorenc. The potential of the ensemble Kalman filter for NWP—A comparison with 4D-Var. *Quart. J. Royal Meteorological Soc.*, 129:3183–3203, 2003.

[20] A. C. Lorenc. Recommended nomenclature for EnVar data assimilation methods. *Research Activities in Atmospheric and Oceanic Modeling*, 5, 2013.

[21] J. Mandel. Efficient implementation of the ensemble Kalman filter. Technical report, University of Colorado at Denver, Center for Computational Mathematics, 2006.

[22] A. Moosavi, A. Attia, and A. Sandu. A machine learning approach to adaptive covariance localization. Technical Report arXiv:1801.00548, 2018.

[23] A. A. Popov and A. Sandu. A Bayesian approach to multivariate adaptive localization in ensemble-based data assimilation with time-dependent extensions. *Nonlin. Processes Geophys.*, 26:109–122, 2019.

[24] S. Ravela, K. Emanuel, and D. McLaughlin. Data assimilation by field alignment. *Physica D: Nonlinear Phenomena*, 230:127–145, 2007.

[25] S. Reich and C. Cotter. *Probabilistic forecasting and Bayesian data assimilation*. Cambridge University Press, 2015.

[26] C. Sampson, A. Carrassi, A. Aydoğdu, and C. K. R. T. Jones. Ensemble kalman filter for nonconservative moving mesh solvers with a joint physics and mesh location update. *Quarterly Journal of the Royal Meteorological Society*, 147(736):1539–1561, 2021.

[27] P. J. van Leeuwen, H. R. Künsch, L. Nerger, R. Potthast, and S. Reich. Particle filters for high-dimensional geoscience applications: A review. *Quart. J. Royal Meteorological Soc.*, 145:2335–2365, 2019.

[28] B. Wang, J. Liu, L. Liu, S. Xu, and W. Huang. An approach to localization for ensemble-based data assimilation. *PLOS ONE*, 13:1–23, 2018.

[29] M. Zhang, J. Cheng, W. Huang, and J. Qiu. An adaptive moving mesh discontinuous Galerkin method for the radiative transfer equation. *Comm. Comput. Phys.*, 27:1140–1173, 2020.

[30] M. Zhang, W. Huang, and J. Qiu. High-order conservative positivity-preserving DG-interpolation for deforming meshes and application to moving mesh DG simulation of radiative transfer. *SIAM J. Sci. Comput.*, 42:A3109–A3135, 2020.

Appendix A

Notation Glossary

Variable	Description
$\mathscr{C}$	Gaspari-Cohn localization function
D	dimension of observation space
d	spatial dimension of PDE
e_i	i^{th} ensemble member
F_K	affine mapping between elements in mesh and reference element
G	Mesh function
$\mathscr{H}$	(nonlinear) observation operator
H	linearization of $\mathscr{H}$
h	refers to spatial discretization, as a subscript
I	identity
j	spatial discretization index
$\mathbf{K}$	Kalman gain matrix
K	element in mesh
$\hat{K}$	reference element
$\mathscr{L}$	function whose gradient gives mesh equation
L	localization parameter
M	dimension of state space
$\hat{m}$	predicted ensemble mean
$\mathbb{M}$	metric tensor

Variable	Description (continued)
$\mathbb{M}_K$	metric tensor at element K
$\mathbb{M}_K^{e_i}$	metric tensor of i^{th} ensemble member at element K
$\mathbb{M}_K^{m}$	metric tensor of mesh obtained from metric tensor intersection of ensemble meshes at element K
$\mathbb{M}_K^{O}$	metric tensor of observation mesh at element K
n	time index
$\mathscr{N}$	normal distribution
N_e	number of ensemble members
N_O	number of observations
N	number of simplicial complexes in mesh
P^b	background error covariance matrix
R	observation error covariance matrix
S	length of spatial domain for 1D Burgers
t	time variable
Δt	observation time scale
T	matrix transpose, as a superscript
T	time index at final time
$\mathscr{T}_h^{e_i}$	i^{th} ensemble mesh
$\mathscr{T}_h$	common mesh
u	state variable
$\hat{u}$	state forecast
u^b	background state
U	numerical solution
V	state vector augmented with mesh locations
x	node location in 1D
x^O	observation location
y	observation variable

Variable	Description (continued)
z	continuous time DA observation variable
Ψ	physical model for discrete time dynamical system
ξ	model noise
η	observation noise
Σ	model error covariance matrix
τ	scalar to determine mesh velocity
Ω	polyhedral domain