Adaptive Piecewise Multilinear Maps for Robust, Generalizable Data-driven Turbulence Models

Niloy Gupta
niloy@umich.edu
University of Michigan
USA

ABSTRACT

This report discusses my class project on implementing an adaptive piecewise multilinear map (PMM) for data-driven RANS turbulence models and comparing them to the existing uniform implementation. Data-driven turbulence models in literature are inadequate for the analysis of complex flows since they lack accuracy, generalizability, and robustness. This can be attributed to the choice of functional form for mapping input features to the desired output. Neural networks are the most common option due to their ability to approximate any function accurately, but they lack robustness when extrapolated. PMMs offer promise because they implement localized learning, but the existing implementations utilize a uniform discretization in the feature-space, resulting in a trade-off between training accuracy and generalizability. For this project, I implement an adaptive PMM that seeks to eliminate this tradeoff to allow for better performing models. The chosen approach is inspired by hyperoctree mesh refinement to enable a computationally efficient implementation. I apply the adaptive PMM to a hypersonic turbulent heat transfer problem and compare its performance to that of a uniform PMM. While the adaptive PMM significantly improves upon the baseline model and uses a fraction of the number of parameters as the uniform PMM, it does not match the uniform PMM's prediction. This can be attributed to a poorly designed adaptive mesh that is unable to capture the level of detail required. Despite these results, the adaptive PMM does show potential for improved modeling of turbulent flows.

KEYWORDS

data-driven turbulence modeling, functional representations

1 INTRODUCTION

Data-driven modeling has emerged as a promising approach for addressing model-form error in Reynolds-averaged Navier-Stokes (RANS) simulations while maintaining computational efficiency. These methods incorporate experimental and high-fidelity data to minimize or eliminate human-introduced assumptions in computational models, thereby reducing error. One of the first successful attempts can be attributed to Duraisamy and coworkers [13, 19]. They demonstrated that, after performing field inversion (FI) to learn a spatially defined discrepancy field β quantifying model-form error, one can identify a set of features to use as inputs to an augmentation model trained to replicate the FI-inferred fields as accurately as possible.

However, FIML also does not guarantee a learnable or robust model. Since the FI problems for each training case are decoupled, the inferred β-fields may not exhibit consistent behavior across cases in the feature space, potentially making it impossible to learn a generalizable model [16, 23]. Moreover, the inferred discrepancy fields may not be learnable using any practical feature set. Since FI is conducted independently of the machine learning step without consideration of the augmentation model's structure, FIML may not yield sufficiently generalizable model augmentations [23].

To address these concerns, Holland et al. [8] and Sirignano et al. [20] proposed Integrated Inference and Machine Learning (IIML). In this approach, the augmentation model is embedded within the solver, replacing the two-step FIML process with direct optimization of the model parameters across all cases simultaneously. By eliminating the intermediate step, IIML inherently accounts for the model's functional form during training, ensuring that the inferred discrepancy fields are realizable by the augmentation model. Additionally, coupling the individual cases during the learning process enforces consistency across them [4, 8, 20].

While model consistency is necessary for a performant model, it is not sufficient. The choice for the model structure greatly affects a data-driven model's training accuracy and extrapolative generalizability and robustness. Many machine learning models use neural networks for their functional forms as they can emulate highly nonlinear functions with great accuracy [9]. However, they lack robustness and generalizability when extrapolated [1, 22, 25]. Neural networks also require large training datasets, typically high-fidelity full-field data from multiple cases. However, for complex flows, available data is typically sparse, noisy, and limited to select flow quantities from a few cases, posing challenges for deep neural networks [11]. They may also learn incorrect physics if the input feature space is not fully populated with data and the training procedure is not properly constrained [2, 23]. These characteristics make neural networks ill-suited for reliable turbulence modeling.

Several alternative structures have been investigated. Matai and Durbin [12] developed a zonal version of FIML using decision trees as a classifier. Instead of continuous outputs, the classifier assigns discrete values to the discrepancy field, better capturing the patchy behavior of inferred augmentation fields. This also enabled the use of exclusively local input features,

allowing for a theoretically more generalizable model. However, decision trees are known for relatively poor extrapolation capabilities.

Srivastava and Duraisamy [23] proposed that model structures should incorporate localized learning to prevent spurious behavior that can arise from nonlinear augmentation functions like simple neural networks. They demonstrated that uniformly discretized piecewise multilinear maps (PMMs) can be used to enforce localized learning and yield robust, generalizable, turbulence models [23]. Gupta and Duraisamy [7] in a later work corroborated these findings. However, both works found that the augmentation performance was hampered by the user's required choice for the discretization level. Finer discretizations improved training accuracy, but prevented the augmentation from yielding significant improvements on unseen cases. Coarser discretizations slightly improved extrapolative performance at the cost of training accuracy.

This report describes the work done to implement an adaptive PMM as the model structure for a data-driven turbulence model to see if it offers improved performance compared to the uniformly discretized version. By being adaptive and choosing an appropriate local discretization level based on the data present in that region of the feature-space, it should increase the capacity of PMMs to approximate much more complex functions without increasing the number of model parameters and provide improved generalizability without sacrificing robustness. Such an implementation should allow data-driven turbulence models to achieve improved predictive capabilities, making them more mature for practical engineering applications.

This report is organized as follows. Section 2 describes the data-driven methodology and concepts that underly this work. Section 3 describes the details and algorithms of the adaptive PMM implementaion. Section 4 compares the performance of the adaptive PMM with a uniformly discretized PMM in the context of a hypersonic turbulent heat transfer modeling problem. Finally, Section 5 summarizes the main contributions of this work.

2 DATA-DRIVEN MODELING METHODOLOGY

I introduce an augmentation, or discrepancy, model into a RANS solver's governing equations by implicitly defining it as a function of features. Since it is a steady-state system in which the model states $\tilde{\mathbf{u}}_m$ are solved for via the residual operator $\mathcal{R}_m$, the augmented governing equations can be expressed as

$$\mathcal{R}_{m,\text{aug}}(\tilde{\mathbf{u}}_m; \boldsymbol{\beta}(\boldsymbol{\eta}(\tilde{\mathbf{u}}_m), \mathbf{w}), \boldsymbol{\xi}) = 0. \tag{1}$$

The augmentation model $\beta(\boldsymbol{\eta}(\tilde{\mathbf{u}}_m), \mathbf{w})$ is evaluated at every location in the mesh. The location's states are used to calculate a feature vector $\boldsymbol{\eta}$, which are the inputs to the model that calculates $\boldsymbol{\beta}$. $\boldsymbol{\beta}$ can be a scalar or a vector, and it is the discrepancy term for that point in the computational domain. The goal is to determine values for the weights $\mathbf{w}$ that result in the best model by following a data-driven process.

The performance of both the baseline model and the augmented models can be assessed using a scalar cost function $\mathcal{J}$. While it is explicitly just a function of $\tilde{\mathbf{u}}_m$, it is also implicitly a function of the augmentation $\boldsymbol{\beta}(\boldsymbol{\eta}(\tilde{\mathbf{u}}_m), \mathbf{w})$, since the augmentation will affect the obtained solution. Thus, determining the best model can be expressed via the optimization problem

$$\begin{aligned} &\min_{\mathbf{w}} \mathcal{J}(\tilde{\mathbf{u}}_m; \boldsymbol{\beta}(\boldsymbol{\eta}(\tilde{\mathbf{u}}_m), \mathbf{w})) \\ &\text{s.t.} \quad \mathcal{R}_{m,\text{aug}}(\tilde{\mathbf{u}}_m; \boldsymbol{\beta}(\boldsymbol{\eta}(\tilde{\mathbf{u}}_m), \mathbf{w}), \boldsymbol{\xi}) = 0. \end{aligned} \tag{2}$$

2.1 Discrete Adjoint Method

Optimization techniques for problems with the form of Eq. (2) are often most efficiently solved using gradient-based methods as they traverse the search space more effectively. The discrete adjoint method in particular, described below, is effective when the number of inputs exceeds the number of outputs.

Given a scalar cost function $\mathcal{J}(\tilde{\mathbf{u}}_m; \beta(\boldsymbol{\eta}(\tilde{\mathbf{u}}_m), \mathbf{w}))$, and leveraging the fact that the the governing equations must always be satisfied, one can derive the following equation for its total derivative with respect to the model parameters $\mathbf{w}$:

$$\frac{d\mathcal{J}}{d\mathbf{w}} = \left(\frac{\partial \mathcal{J}}{\partial \beta} - \frac{\partial \mathcal{J}}{\partial \tilde{\mathbf{u}}_m} \left[\frac{\partial \mathcal{R}_{m,\text{aug}}}{\partial \tilde{\mathbf{u}}_m} \right]^{-1} \frac{\partial \mathcal{R}_{m,\text{aug}}}{\partial \beta} \right) \frac{\partial \beta}{\partial \mathbf{w}}. \tag{3}$$

Each of the partial derivatives in this equation can be calculated directly using backpropagation or algorithmic differentiation techniques. However, this requires a matrix inversion to calculate a triple-matrix product. To avoid this, one can separately compute $\frac{\partial \mathcal{J}}{\partial \tilde{\mathbf{u}}_m} \left[\frac{\partial \mathcal{R}_{m,\text{aug}}}{\partial \tilde{\mathbf{u}}_m} \right]^{-1}$. This product is referred to as the adjoint vector, and it is the solution $\boldsymbol{\psi}^T$ of the adjoint system that corresponds to the forward problem, given by

$$\boldsymbol{\psi}^T \frac{\partial \mathcal{R}_{m,\text{aug}}}{\partial \tilde{\mathbf{u}}_m} = -\frac{\partial \mathcal{J}}{\partial \tilde{\mathbf{u}}_m}. \tag{4}$$

Therefore, the desired sensitivities are obtained by calculating

$$\frac{d\mathcal{J}}{d\mathbf{w}} = \left(\frac{\partial \mathcal{J}}{\partial \beta} + \boldsymbol{\psi}^T \frac{\partial \mathcal{R}_{m,\text{aug}}}{\partial \beta} \right) \frac{\partial \beta}{\partial \mathbf{w}} \tag{5}$$

where $\boldsymbol{\psi}^T$ is obtained by solving Eqn. (4) [5, 6].

2.2 Learning and Inference Assisted by Feature-space Engineering

The general modeling framework I use for this effort is learning and inference assisted by feature-space engineering (LIFE). First proposed by Srivastava and Duriasamy [23], LIFE extends IIML by providing guiding principles for developing model augmentations that: 1) generalize to unseen configurations and cases, 2) minimize spurious predictions, and 3) train using limited data. The framework's key contribution is the concept of localized learning. Models that utilize localized learning restrict modifications to regions of the feature-space

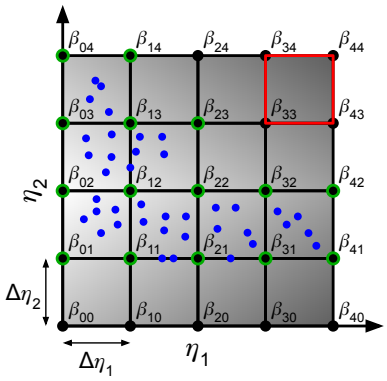
with training data, ensuring that the model behaves according to the baseline model elsewhere. Since the baseline model is a physically valid solution, this prevents spurious behavior from occurring. Several approaches can enforce localized learning, including using piecewise representations of a discretized feature-space, superposing finite support functions centered at available data points, and introducing artificial data points to enforce desired behavior in regions lacking training data. The LIFE framework leaves choosing hyperparameters, functional forms, and specific implementation strategies to the user to allow them to strike an optimal balance between localizing the influence of data for robustness and having larger regions of influence for generalizability [23].

2.3 Piecewise Multilinear Maps

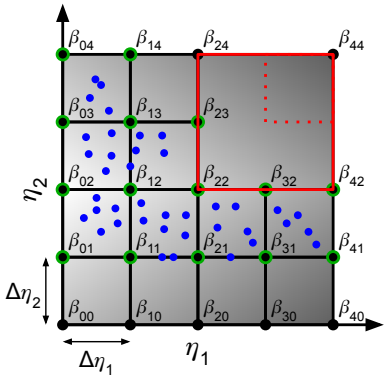
A piecewise multilinear map (PMM) is a specific discretized representation of a function defined over a bounded hyperrectangle in $\mathbb{R}^n$. Rather than store an explicit function of the inputs, it stores a piecewise continuous mapping from a set of coordinates of the inputs to their outputs and linearly interpolates to calculate the function given an arbitrary input. The parameters defining a model using this feature-space representation are the output values stored at the discretized nodes. PMMs inherently employ localized learning. The output values corresponding to an input coordinate are solely determined by the values stored at the vertices of the lowest dimensional hyperrectangle that encompasses it. Therefore, when derivatives are calculated using backpropagation, there are only nonzero values for nodes that are adjacent to hyperrectangles containing data. The degree to which one wants to enforce localized learning can be controlled by how finely one discretizes the feature-space.

However, PMMs have drawbacks. Their piecewise linear assumption limits them to approximating functions with low total variation, necessitating more careful feature design compared to other representations. It also suffers from the curse of dimensionality when a large number of features with fine discretizations is desired. Consequently, PMMs require a sparse feature set and coarse discretization that may limit augmentation accuracy.

2.3.1 Existing Uniform Implementation. For turbulence modeling, the only examined PMM structure has been a uniformly discretized PMM. A two-dimensional example of this is illustrated in Figure 1a. For this example, to evaluate the function stored by the PMM at an arbitrary point, one would first determine which cell it lies in before using the values β stored at the four neighboring nodes to perform bilinear interpolation at the desired point. If the desired point lies on a cell edge, a simple one-dimensional interpolation using the endpoints is performed instead. Since only the four nodes adjacent to the cell are used to calculate the function's value, only these nodes have non-zero derivatives during backpropagation and are modified during learning for the desired point.



(a) Uniformly discretized PMM



(b) Adaptive PMM

Figure 1: Two-dimensional examples comparing PMM implementations. The model's parameters are the values β_{ij} that are the output of the model at that node in the feature-space, which are represented by black dots. The blue dots represent data points in the training set. Green circles identify nodes that are modified from the baseline value during training.

However, if one examines the red square, one can see that any evaluation within this square will only predict the baseline model value. Since no training data is present in that cell or an adjacent one, its nodes' values are never modified. While this ensures a robust model since we do not have training data for that cell, it hampers generalizability as the training data present in more distant cells can likely provide information about what should happen in the red cell.

3 PROPOSED ADAPTIVE PMM IMPLEMENTATION

Figure 1b shows how the some of the issues with a uniformly discretized PMM can be alleviated. If the grid is coarsened in the upper-right quadrant to reflect that there is no data in

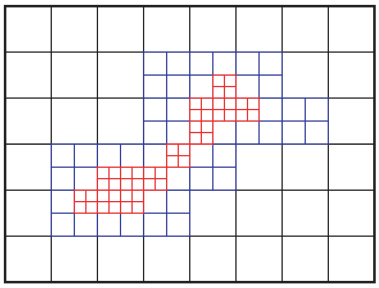


Figure 2: Example of a two-dimensional hierarchical rectangular mesh.

any of the cells, then the coarser grid can be used to extend the range of influence of the training data to the dotted red square. In theory, this should enhance generalizability, and also has the added benefit of reducing the number of model parameters.

Adaptive PMMs for turbulence modeling do not currently exist in literature. This is likely due to simplicity and computational cost. By having n_k regularly spaced nodes in each dimension $k = 1, ..., K$, the model can be simply stored using the minimum value, maximum value, and n_k for each dimension and the function's value at each node, resulting in $(3K + n_1 \cdots n_K)$ total parameters. No other grid information is required. Furthermore, given a feature-space coordinate, evaluating the PMM has constant time complexity and cell-indexing can be done explicitly. The formulas for evaluating the PMM are also simply the K-linear interpolation formulas using 2^K points. In contrast, if an unstructured mesh is used to represent the PMM, the required memory would significantly increase and cell-indexing would no longer be constant in time, making the model much more expensive.

3.1 Mesh structure and representation

To resolve this, my adaptive PMM takes inspiration from pointwise adaptive mesh refinement strategies. Specifically, I represent the feature-space using a hierarchical rectangular mesh based on hyperoctrees (the generalization of two-dimensional quadtrees and three-dimensional octrees). First, a baseline discretization level is chosen. Then, based on the availability and significance of training data, the domain is progressively discretized up to a maximum level. A two-dimensional example of this with two levels is shown in Figure 2. In the context of this project, the smaller red cells would contain many training data points and allow for a more detailed representation of the augmentation to improve training accuracy, while the larger black cells would have few, if any, training data and be used to improve generalizability.

For this project, I developed a simple n-dimensional hyperoctree mesh file format. It contains a header line and two sections. The header provides the number of dimensions n_{dim}, the number of cells at the coarsest level for each dimension $m_1, ..., m_{n_{dim}}$, and the number of refinement levels n_r. The first section contains a subsection for each level j in the mesh. Each subsection contains $2^{j \cdot n_{dim}} \prod_i^{n_{dim}} m_i$ integers. These integers can either have a value of -2 if the cell at that level does not exist in the hyperoctree mesh, -1 if it exists but is refined, or a non-negative value prescribing a cell index if it does exist in the mesh at that level. Therefore, the first subsection cannot have a -2 value, and the last cannot have a -1 value. The largest integer value present in this section is equal to the total number of cells in the mesh n_c minus one. The information contained in this section is referred to as the "mesh tree." The second section is a two-dimensional array of size $n_c \times 3^{n-1}$. Each row provides the node indices that define or lie on the cell. The first $2^{n_{dim}}$ values in the row describe which nodes are the vertices of the cell. The remaining values are either -1 if that corresponding hanging node does not exist in the mesh or a non-negative node index if it does exist and lies on a lower dimensional face of that cell. This two-dimensional array is referred to as the "cell-to-node" array. The functional values at these nodes are stored in a separate file.

3.2 PMM output calculation

To calculate the output of the adaptive PMM, I follow the procedure outlined by Algorithm 1. For this project, only a two-dimensional version is developed and implemented.

With appropriate preprocessing of the mesh, cell properties and node indices can be obtained by simplying accessing elements of arrays. The only search required is finding the appropriate level in the hyperoctree. Therefore, this approach has linear time-complexity with respect to the number of levels in the mesh, constant-time complexity with respect to the number of features or dimensions, and constant-time complexity with respect to the number of model parameters. This makes it suitable to be integrated within a RANS solver as long as the number of refinements is kept to a small number.

4 APPLICATION TO HYPERSONIC TURBULENT HEAT TRANSFER MODELING

I investigate the efficacy of the adaptive PMM by comparing it to data presented by Gupta and Duraisamy [7] for their hypersonic turbulent heat transfer modeling effort.

4.1 Modeling Problem and Augmentation Choice

Since the most commonly used turbulence models were designed and calibrated for subsonic incompressible flows without significant pressure gradients or heat transfer, they struggle to accurately predict hypersonic flows. In particular, when applied to hypersonic flows exhibiting shock-boundary layer interactions (SBLIs), they grossly overestimate the predicted heat transfer [15]. One of the sources for this error is the turbulent heat transfer model, typically given by

Algorithm 1 Algorithm for calculating the adaptive PMM output in n_{dim}-dimensions

Input: Feature values $\eta_1, ...\eta_{n_{dim}}$ in each dimension
Output: Adaptive PMM functional value β

procedure DETERMINE CELL INDEX($\eta_1, ...\eta_{n_{dim}}$)
  for each level **do**
    for each dimension **do**
      Calculate the cell index in that dimension
      Calculate the minimum and maximum η_i values in that cell
    end for
    Calculate the unrolled cell index for that level
    Check the mesh tree to see if the cell is a leaf or is refined
    if cell is a leaf **then**
      Obtain overall cell index from the mesh tree
      break
    end if
  end for
end procedure
Extract the cell's row in the cell-to-node array
if cell does not have a hanging node **then**
  Use bilinear interpolation of the parameter values at the cell's vertices to calculate β
else
  Use inverse distance weighting ($p = 2$) with all nodes on the cell to calculate β
end if

$$c_p\overline{\rho u_j'' T''} \approx -\frac{c_p\overline{\mu}_t}{\mathrm{Pr_t}}\frac{\partial\hat{T}}{\partial x_j}. \tag{6}$$

It is analogous to the model for the mean laminar heat transfer and makes use of Morkovin's hypothesis to provide a constant value for the turbulent Prandtl number $\mathrm{Pr_t}$ close to unity [3, 21]. However, in hypersonic flows and flows with shockwaves, compression and dilatation cannot be ignored, invalidating this assumption and requiring a variable-$\mathrm{Pr_t}$ model [10, 14, 17]. Consequently, I augment the baseline turbulence model, the Wilcox 2006 $k-\omega$ model [24], through the modification

$$\mathrm{Pr_t} = \exp\left(\beta\left(\boldsymbol{\eta}(\tilde{\mathbf{u}}_m)\right)\right)\mathrm{Pr_{t,0}}, \tag{7}$$

where $\mathrm{Pr_{t,0}} = 0.9$ is the baseline value.

4.2 Objective Function and Inference Problem

I assess the turbulence model and RANS performance using the objective function

$$\mathcal{J} = \frac{1}{n_x}\sum_{i=1}^{n_x}\left(q_{w,m}(x_i) - q_{w,ref}(x_i)\right)^2, \tag{8}$$

Table 1: Features used and PMM configuration for the preliminary model

Feature	Lower Bound	Upper Bound	# Nodes (Level 0)
$4\tanh\left(\frac{\log_{10}(\nu_t/\nu+\varepsilon)}{0.75\cdot 4}\right)$	-4.0	4.0	6
$1.5\tanh\left(\frac{1}{1.5}\frac{-\nabla\overline{T}\cdot\hat{n}}{\overline{T}/d_{\mathrm{wall}}}\right)$	-1.5	1.5	6

which is the mean-squared error over n_x faces of the wall heat transfer predictions by the model with respect to reference data constructed from the experimental data provided by Schulein [18] for oblique shocks impinging the Mach 5 boundary layer over a flat plate. The data contains wall heat transfer distributions for three different shock strengths of 6°, 10°, and 14° deflection angles, which will be referred to as Cases S6, S10, and S14 respectively.

The inference problem can be stated as Eqn. 2 with the objective function $\mathcal{J}$ provided by Eqn. 8. Since the goal of this effort is to focus on model structures, the input features for the adaptive PMM are fixed to match the work conducted by Gupta and Duraisamy [7].

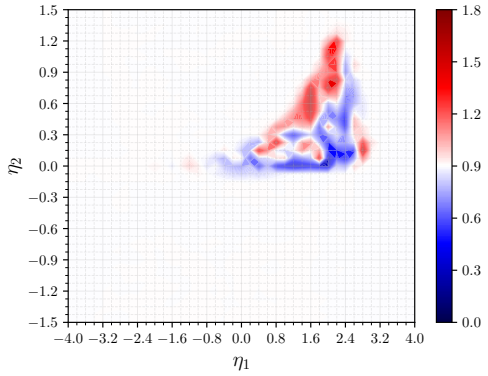
4.3 Model Design

Table 1 describes the features and coarsest discretization level of the augmentation model. For this effort, I use 3 refinement levels to achieve a maximum feature-space fidelity equal to the global feature-space fidelity of the preliminary model of Gupta and Duraisamy [7].

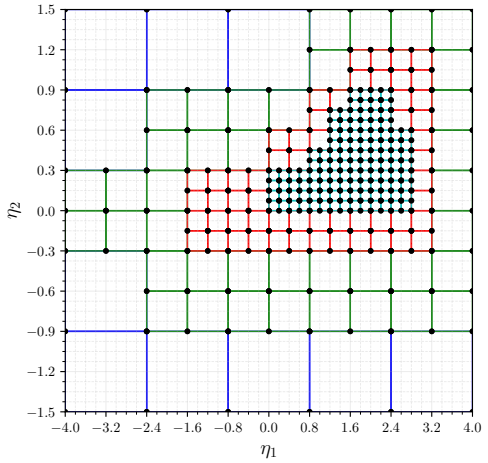
The adaptive mesh is designed by hand based on the inferred augmentation mapping and feature-space distribution provided by Gupta and Duraisamy [7]. Figure 3 compares the two. The finest discretization level is used in the upper right quadrant of the feature-space where most of the data is present and the inferred augmentation exhibits the complex behavior. Beyond this, the discretization level of the remaining feature-space is chosen to be as coarse as possible while maintaining a balanced mesh that has neighboring cells differing by at most a factor of two in size. The uniform mesh required using 1681 parameters to fully specify the model, but only approximately 14% of them were significant [7]. With the adaptive mesh, only 261 parameter are now required.

4.4 Training Results

Figure 4 compares the training history of the augmentations using the two PMM structures. Both were trained using a form of the BFGS algorithm with an explicitly defined step size. The uniform PMM was able to obtain two orders of magnitude improvement across all cases in roughly 20 steps. The adaptive PMM was at best able to obtain one order of magnitude improvement in 8. In order to obtain this improvement, the adaptive PMM augmentation also required a carefully chosen, significantly smaller, step size evolution.

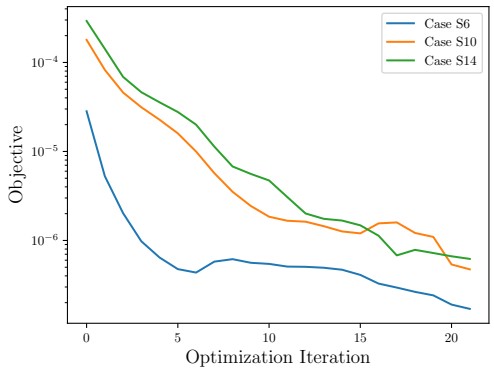


(a) Inferred augmentation mapping of the feature-space coordinates to the predicted $\mathrm{Pr_t}$-values for the uniform PMM. Intersections of the gray lines correspond to stored nodal/parameter values of the model. [7]

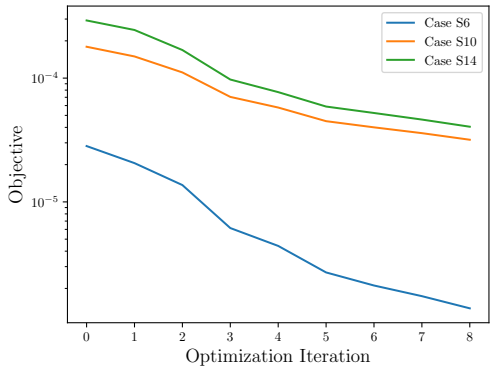


(b) Adaptive PMM mesh. Each color corresponds to a cell of a different level. Black dots at the intersection of cell edges correspond to stored nodal/parameter values of the model.

Figure 3: Comparison of the uniform and adaptive PMM meshes used as the augmentation model structures.



(a) Uniform



(b) Adaptive

Figure 4: Comparisons of the training history for both PMM implementations.

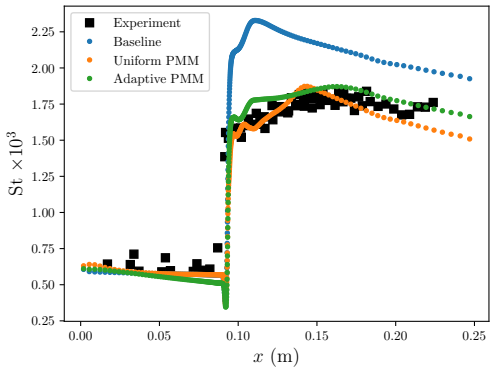


Figure 5: Comparison of the inferred heat transfer distributions of the inferred augmentations with the baseline model and reference data for Case S6.

Figure 5 compares the inferred heat transfer distributions in terms of the Stanton number $\mathrm{St} = \frac{q_w}{\rho_\infty u_\infty c_p (T_r - T_w)}$ for Case S6. The augmentation using the adaptive PMM results in a slightly worse heat transfer prediction compared to that using the uniform PMM. This is the case for which the adaptive PMM performs the best on; the inferred distributions for Cases S10 and S14 are significantly worse than that of the uniform PMM.

Figure 6, which shows the inferred $\mathrm{Pr_t}$ fields for both augmentations for Case S6, gives insight on why the adaptive PMM results in a less accurate prediction. The augmentation using the adaptive PMM is unable to learn some of the features that the one using the uniform PMM can. Namely, the bands of increased $\mathrm{Pr_t}$ in the top of the boundary layer upstream and just downstream of the shock and in the near wall region

downstream of the shock are missing from the adaptive PMM's prediction.

The reason for the difference in the predicted fields can be seen in Figure 7, which shows the contours of the inferred augmentation in the feature-space for the adaptive PMM. The inferred mapping only contains two small regions of slightly increased $\mathrm{Pr_t}$. The large region of significantly increased $\mathrm{Pr_t}$ present in the augmentation inferred using the uniform PMM (see Figure 3a) is missing. The reason for this is likely due to the quality of the adaptive mesh. More cells at the highest mesh refinement level may be required to infer the desired behavior.

5 CONCLUSION AND DISCUSSION

This report describes the preliminary work done to develop an adaptive PMM to use as a model structure for data-driven turbulence models. I propose an adaptive PMM to improve the predictive performance of data-driven models using uniform PMMs without sacrificing training accuracy by increasing the size of cells in the feature-space where data is sparse. This will increase the range of influence of the training data, theoretically allowing it to generalize better when extrapolated outside the training regime. To store the adaptive mesh, I use a hyperoctree mesh structure inspired by adaptive mesh refinement strategies. This allows for relatively efficient storage of the mesh and interpolation calculations to have constant time-complexity with respect to the number of features and parameters in the model.

I compare the adaptive PMM's performance to that of the uniformly discretized PMM in the context of turbulent heat transfer modeling for hypersonic flows. While the adaptive PMM results in a model that is able to improve the accuracy of wall heat transfer predictions from the baseline using a fraction of the number of parameters, it is unable to match the performance of the uniform PMM. This can be attributed to its inability to resolve all the detail that the uniform PMM was able to infer, which is likely the result of using a poorly designed adaptive mesh.

Future work will be focused on developing automatic adaptive mesh generating techniques to avoid the suspected issues with manually-designed meshes. In addition, generalizing the algorithms and code developed for this project to an arbitrary number of dimensions and improving the hyperoctree mesh storage and search algorithms to be more efficient are critical concerns. Once these tasks are completed and the adaptive PMM is refined, the improved scaling behavior should enable significantly more robust, generalizable, and efficient data-driven turbulence models.

More information about this work can also be found at its website: https://niloygupta-um.github.io/CSE598-Adaptive-PMM.

REFERENCES

[1] E. Barnard and L.F.A. Wessels. 1992. Extrapolation and interpolation in neural network classifiers. *IEEE Control Systems Magazine* 12, 5 (1992), 50–53. https://doi.org/10.1109/37.158898

[2] Andrea Beck and Marius Kurz. 2021. A perspective on machine learning methods in turbulence modeling. *GAMM-Mitteilungen* 44, 1 (2021), e202100002. https://doi.org/10.1002/gamm.202100002 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/gamm.202100002

[3] L. Duan, I. Beekman, and M. P. Martín. 2011. Direct numerical simulation of hypersonic turbulent boundary layers. Part 3. Effect of Mach number. *Journal of Fluid Mechanics* 672 (2011), 245–267. https://doi.org/10.1017/S0022112010005902

[4] Yuan Fang, Yaomin Zhao, Fabian Waschkowski, Andrew S. H. Ooi, and Richard D. Sandberg. 2023. Toward More General Turbulence Models via Multicase Computational-Fluid-Dynamics-Driven Training. *AIAA Journal* 61, 5 (2023), 2100–2115. https://doi.org/10.2514/1.J062572 arXiv:https://doi.org/10.2514/1.J062572

[5] Michael B. Giles, Mihai C. Duta, Jens-Dominik Muller, and Niles A. Pierce. 2003. Algorithm Developments for Discrete Adjoint Methods. *AIAA Journal* 41, 2 (2003), 198–205. https://doi.org/10.2514/2.1961 arXiv:https://doi.org/10.2514/2.1961

[6] Michael B. Giles and Niles A. Pierce. 2000. An Introduction to the Adjoint Approach to Design. *Flow, Turbulence and Combustion* 65, 3 (01 Dec 2000), 393–415. https://doi.org/10.1023/A:1011430410075

[7] Niloy Gupta and Karthik Duraisamy. 2026. Computational and physical considerations for the development of machine learning augmented turbulence models. *International Journal of Heat and Fluid Flow* 117 (2026), 110089. https://doi.org/10.1016/j.ijheatfluidflow.2025.110089

[8] Jonathan R. Holland, James D. Baeder, and Karthikeyan Duraisamy. 2019. *Field Inversion and Machine Learning With Embedded Neural Networks: Physics-Consistent Neural Network Training.* https://doi.org/10.2514/6.2019-3200 arXiv:https://arc.aiaa.org/doi/pdf/10.2514/6.2019-3200

[9] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* 2, 5 (1989), 359–366. https://doi.org/10.1016/0893-6080(89)90020-8

[10] C. C. Horstman and F. K. Owen. 1972. Turbulent properties of a compressible boundary layer. *AIAA Journal* 10, 11 (1972), 1418–1424. https://doi.org/10.2514/3.6640 arXiv:https://doi.org/10.2514/3.6640

[11] J. Nathan Kutz. 2017. Deep learning in fluid dynamics. *Journal of Fluid Mechanics* 814 (2017), 1–4. https://doi.org/10.1017/jfm.2016.803

[12] R. Matai and P. A. Durbin. 2019. Zonal Eddy Viscosity Models Based on Machine Learning. *Flow, Turbulence and Combustion* 103, 1 (01 Jun 2019), 93–109. https://doi.org/10.1007/s10494-019-00011-5

[13] Eric J. Parish and Karthik Duraisamy. 2016. A paradigm for data-driven predictive modeling using field inversion and machine learning. *J. Comput. Phys.* 305 (2016), 758–774. https://doi.org/10.1016/j.jcp.2015.11.012

[14] Stephan Priebe and M. Pino Martín. 2021. Turbulence in a hypersonic compression ramp flow. *Phys. Rev. Fluids* 6 (Mar 2021), 034601. Issue 3. https://doi.org/10.1103/PhysRevFluids.6.034601

[15] Pratikkumar Raje, Eric Parish, Jean-Pierre Hickey, Paola Cinnella, and Karthik Duraisamy. 2025. Recent developments and research needs in turbulence modeling of hypersonic flows. *Physics of Fluids* 37, 3 (03 2025), 031304. https://doi.org/10.1063/5.0253703 arXiv:https://pubs.aip.org/aip/pof/article-pdf/doi/10.1063/5.0253703/20440570/031304_1_5.0253703.pdf

[16] Mario J. Rincón, Martino Reclari, Xiang I. A. Yang, and Mahdi Abkar. 2025. A generalisable data-augmented turbulence model with progressive and interpretable corrections. arXiv:2503.18568 [physics.flu-dyn] https://arxiv.org/abs/2503.18568

[17] JC Rotta. 1965. Heat transfer and temperature distribution in turbulent boundary layers at supersonic and hypersonic flow. *AGARDograph* 97 (1965), 41–63.

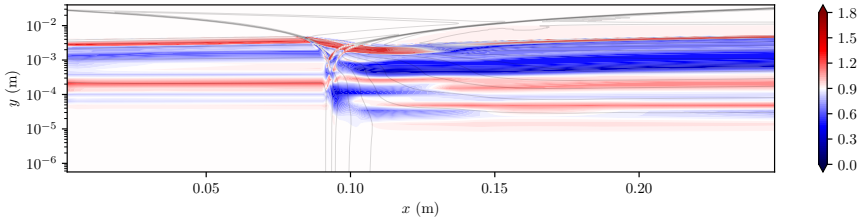
[18] Erich Schülein. 2006. Skin Friction and Heat Flux Measurements in Shock-/Boundary Layer Interaction Flows. *AIAA Journal* 44, 8 (2006), 1732–1741. https://doi.org/10.2514/1.15110 arXiv:https://doi.org/10.2514/1.15110

[19] Anand Pratap Singh and Karthik Duraisamy. 2016. Using field inversion to quantify functional errors in turbulence closures. *Physics of Fluids* 28, 4 (04 2016), 045110. https://doi.org/10.1063/1.4947045 arXiv:https://pubs.aip.org/aip/pof/article-pdf/doi/10.1063/1.4947045/14093553/045110_1_online.pdf

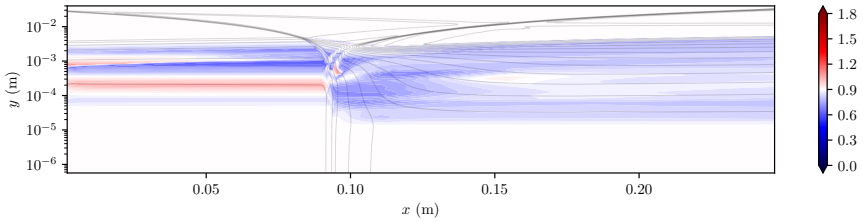
[20] Justin Sirignano, Jonathan F. MacArt, and Jonathan B. Freund. 2020. DPM: A deep learning PDE augmentation method with application to large-eddy simulation. *J. Comput. Phys.* 423 (2020), 109811. https://doi.org/10.1016/j.jcp.2020.109811

[21] Alexander J Smits and Jean-Paul Dussauge. 2006. *Turbulent shear layers in supersonic flow.* Springer Science & Business Media.

[22] Vishal Srivastava. 2022. *Generalizable Data-driven Model Augmentations Using Learning and Inference assisted by Feature-space Engineering.* PhD



(a) Uniform



(b) Adaptive

Figure 6: Comparisons of the training history for both PMM implementations.

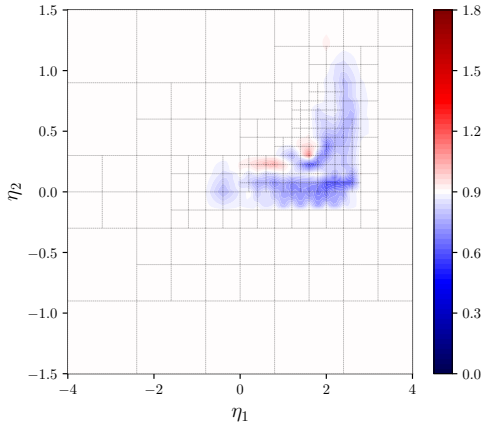


Figure 7: Contour plot of the inferred augmentation in the feature-space using the adaptive PMM.

thesis. University of Michigan, Ann Arbor, Ann Arbor, MI. https://doi.org/10.7302/6032

[23] Vishal Srivastava and Karthik Duraisamy. 2021. Generalizable physics-constrained modeling using learning and inference assisted by feature-space engineering. *Phys. Rev. Fluids* 6 (Dec 2021), 124602. Issue 12. https://doi.org/10.1103/PhysRevFluids.6.124602

[24] David C. Wilcox. 2008. Formulation of the k-w Turbulence Model Revisited. *AIAA Journal* 46, 11 (2008), 2823–2838. https://doi.org/10.2514/1.36541 arXiv:https://doi.org/10.2514/1.36541

[25] Min Zhu, Handi Zhang, Anran Jiao, George Em Karniadakis, and Lu Lu. 2023. Reliable extrapolation of deep neural operators informed by physics or sparse observations. *Computer Methods in Applied Mechanics and Engineering* 412 (2023), 116064. https://doi.org/10.1016/j.cma.2023.116064