Model Reduction for Dynamical Systems: Machine Learning and Memory

Brek Meuris

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2022

Reading Committee:

Alberto Aliseda, Chair

John Kramlich

Panos Stinis

Program Authorized to Offer Degree:
Mechanical Engineering

University of Washington

Abstract

Model Reduction for Dynamical Systems:
Machine Learning and Memory

Brek Meuris

Chair of the Supervisory Committee:
Professor Alberto Aliseda
Department of Mechanical Engineering

Many of the physical phenomena arising in science and engineering can be described by partial differential equations. Typically, partial differential equations of interest cannot be solved analytically and require the use of scientific computing methods, machine learning methods, or a combination of these methods to approximate the solution. Spectral methods are one technique commonly utilized by the scientific computing community to solve partial differential equations due to their ability to achieve a high order of convergence; however, spectral methods require an appropriate choice of basis function—which is not always obvious—and result in a high-dimensional system of equations that must be solved. Reduced order modeling seeks to address the high-dimensionality of these systems of equations and reproduce the dominant features of the solution while utilizing a lower-dimensional system. For systems that are multiscale in nature, reduced order modeling becomes more nuanced, as it is no longer feasible to simply truncate and evolve just a subset of the degrees of freedom.

The focus of this work is the development of two frameworks for the evolution of partial differential equations. The first is a reduced order modeling framework for multiscale dynamical systems that is based on the Mori–Zwanzig formalism. As part of this work, a parameter is introduced to control the time decay of the memory term, which results from the application of the Mori–Zwanzig formalism, and is selected based on data obtained from a

well-resolved full simulation. This framework is applied first to the singular, one-dimensional inviscid Burgers equation as a proof-of-concept and then extended to the three-dimensional Euler equations, where full simulations are only feasible for short times. The second is a scientific machine learning framework that utilizes deep operator neural networks (DeepONets) for the identification of candidate basis functions. The candidate basis functions—which are custom-made for the dynamical system of interest—are orthonormalized through the singular value decomposition and are suitable for usage in a spectral method. Model reduction is possible in this framework through the specification of a singular value threshold, thereby allowing a lower-dimensional system to be utilized for the evolution of a partial differential equation. This framework is applied to six one-dimensional models to provide a proof-of-concept: the advection, advection-diffusion, viscous Burgers, Korteweg–de Vries, Kuramoto–Sivashinsky, and inviscid Burgers equations.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to begin by expressing my appreciation to the University of Washington, where I had the opportunity to have a truly interdisciplinary course of study, work with individuals from multiple colleges and departments, and utilize the many resources available at the university, such as the HYAK supercomputer system.

To Panos Stinis, my advisor for all of this work, I cannot begin to express my gratitude for all his mentoring, for modeling what it means to be a researcher, and for the unwavering support during times of good results as well as during the setbacks. I am deeply appreciative that he always made time for fascinating discussions and continual feedback while allowing me the freedom to truly explore and grow as a researcher.

To Ken Yasuhara, for affording me the opportunity to serve as his graduate assistant for multiple academic years in the Office for the Advancement of Engineering Teaching and Learning. I am ever grateful for the opportunity and the genuine care he showed for my development. The time spent working alongside him was truly transformative and continually pushed my development as an educator and a learner.

To my reading committee, starting with Alberto Aliseda, I want to thank him for all of his guidance and, most notably, his willingness to serve as the chair of my reading committee while allowing me the freedom to explore the topics which interested me the most. To John Kramlich, for serving on my qualifying exam committee all the way through to being one of my reading committee members and for providing continual developmental opportunities through teaching assistant appointments during my early years in the department. To Dana Dabiri, for the continued support, from my qualifying exam committee all the way to the end.

I would like to thank Ann Mescher for initially recruiting me to come to the University of Washington and for her guidance and patience over my initial time in the department. To my fellow labmate during my early years in the department, Sneha Sondur, for her help navigating the department as a new grad student and for all the thoughtful discussions. To Wanwisa Kisalang, for always being willing to listen and provide guidance, regardless of the situation.

To all of my collaborators: Jake Price, Saad Qadeer, Madelyn Shapiro. Without their contributions and discussions, this work would not be at the level that it is at today. To Daiga Koenig, Steven Koenig, Paul Spence, and Thomas Berfield. The education, mentoring, and opportunities provided by the four of them were instrumental in my development and truly facilitated the commencement of this endeavor.

To all of my family and friends who never stopped asking how everything was going and for their continual encouragement. To Alexa Deep, I cannot thank her enough for her endless support and infinite understanding. To Kurt Meuris, for letting me know anything is possible, for always being supportive, and for always listening.

This is only a small listing of the many individuals, past and present, to whom I owe a debt of gratitude, so to all of you who were not noted above, thank you.

Chapter 1
INTRODUCTION

Many of the physical phenomena arising in science and engineering can be described by partial differential equations (PDEs). Typically, PDEs of interest for modern applications cannot be solved analytically and require alternative approaches. To address this need, the methods devoted to solving PDEs have continued to evolve over the last 70+ years with the continued increases in computing power. Scientific computing and, more recently, machine learning, are two methods commonly employed to approximate the solution to PDEs. Scientific computing has a long history and comprises many methods for the numerical simulation of the underlying PDE, while machine learning methods are a more recent development and are being increasingly applied to PDEs with the goal of identifying the dynamical systems described by a PDE from data [64]. The application of scientific computing techniques requires the solution of a high-dimensional system and has led to the development of reduced order modeling techniques that focus on modeling the dominant features of a system. The application of machine learning techniques can result in a lack of "explainability" [77] of the model and has led to the development of scientific machine learning [3] techniques that attempt to leverage the strengths of both scientific computing and machine learning.

1.1 Scientific computing

Starting with finite-difference methods, which predate the explosion of computing power in the second half of the 20th century, scientific computing methods for solving PDEs now include methods such as the finite-element method, the finite-volume method, and the spectral method [73]. Within each of these methods, there are a wealth of techniques for dealing with the nuanced spatiotemporal nature of PDEs and the complex dynamics they exhibit.

These four methods differ in their representation of approximate solutions, but all result in a high-dimensional system of equations that must be solved. Furthermore, the dimensionality of this system of equations is linked to the level of desired resolution. Spectral methods have constituted a significant part of the field of scientific computing due to their connection to approximation theory and ability to achieve a high order of convergence [11]. Spectral methods rely on a set of basis functions (e.g., Fourier series) to develop a truncated series expansion of the solution to the PDE. The series expansion is then substituted into the PDE of interest, a residual function is defined, and the expansion coefficients are chosen to minimize the residual function in an appropriate sense [9]. One of the main considerations when utilizing spectral methods is the choice of basis function, which may not be obvious in advance and requires considerations of speed, accuracy, and/or boundary condition constraints [10].

For many real-world applications, the solutions to the PDEs of interest are too complicated to properly resolve all length scales and/or the high-dimensional system of equations resulting from a spectral method (or any of the other three methods noted above) is too expensive for direct numerical simulation. Reduced order modeling attempts to reproduce the dominant features of the solution while utilizing a lower-dimensional system. One of the more popular frameworks for constructing reduced order models is proper orthogonal decomposition (POD) [12, 34]. Proper orthogonal decomposition for PDEs relies on the singular value decomposition (SVD) to generate basis functions that are tailored to the problem of interest and are optimal in a two-norm sense [10]. A common implementation approach of the POD algorithm relies on snapshots of the solution of the PDE of interest to generate a matrix where each column represents a distinct instant of time. The left singular vectors resulting from performing the SVD on this snapshot matrix provide the POD modes, and using the singular value spectrum, a threshold for the truncation of the POD modes can be established. Spectral methods are then often utilized in conjunction with the truncated POD modes to construct a reduced order model (ROM) of the PDE.

One critical aspect of reduced order modeling is the specification of the truncation threshold for generating the lower-dimensional system. For the POD approach, truncation is often

selected based on the singular value spectrum and effectively treats the discarded POD modes as unimportant for evolving the system at the desired accuracy. In the case of a system that displays behavior at multiple scales, the ability to disregard the additional degrees of freedom is no longer a safe assumption, and the construction of reduced order models becomes more nuanced. For problems that are multiscale in nature, such as turbulence modeling, which contains spatiotemporal motion involving a wide range of scales [15, 75] and is of particular interest to the mathematical fluid dynamics community, a modeler must use knowledge of each scale and the relationships between scales to construct an appropriate ROM. If there is a clear demarcation between the scales present, and they can be sorted into a few discrete groupings, there are a variety of techniques that have been developed (e.g., see [30, 78] and references therein). In many cases, there is no clear demarcation between the scales, or solutions have characteristics that would require a resolution down to the zero-length scale (e.g., shocks). When solutions develop activity at length scales below the smallest scale available to the simulation, the computed solution becomes under-resolved and degrades the accuracy of the solution. One mathematical framework for constructing ROMs for systems that are truly multiscale in nature is the Mori–Zwanzig (MZ) formalism. The MZ formalism, which originated from irreversible statistical mechanics [85], has been modernized and used successfully to construct reduced order models (e.g., [5, 18, 56]). The MZ formalism provides an exact and general framework for the reduction of the dynamics of a full system to the dynamics of a set of reduced (resolved) variables without requiring any explicit scale separation [78]. In multiscale models, the unresolved variables impact the remaining resolved variables (the modes retained after truncation in the non-multiscale POD case), and this interaction is accounted for by a memory term that results from the application of the formalism.

1.2 Machine learning

Machine learning—a subfield of artificial intelligence—is focused on how computer systems can be constructed so that they automatically improve through experience [37] and are typically cast as a problem of improving some performance metric when executing a given task,

and where this improvement is obtained through experience. Recent advances in a particular subfield of machine learning, referred to as deep learning, have changed the trajectory of artificial intelligence. Through the use of deep neural networks, deep learning utilizes a representation mechanism of learning where the composition of simple, nonlinear models transform representations at one level into representations at a higher but more abstract level [42]. Algorithmic advances, coupled with the improvement in computing resources in the form of graphics processing units (GPUs), along with the explosion of data available for training deep neural networks, helped lead to the resurgence of neural networks in the 21st century. These recent advances have allowed deep neural networks to become very adept at discovering structure in high-dimensional data and have led to state-of-the-art performance in applications, such as image classification, machine translation, and natural language processing [31, 42].

Deep neural networks have also been utilized to approximate the solution to PDEs. Applications of deep neural networks to PDEs have utilized a variety of network structures, such as feedforward neural networks (FFNNs) [64], recurrent neural networks (RNNs) [52], residual networks (ResNets) [62], and convolutional neural networks (CNNs) [79]. More recently, there has been renewed interest in using neural networks not just to approximate functions (i.e., maps between finite-dimensional Euclidean spaces) but also to approximate operators (i.e., maps between infinite-dimensional function spaces) and, more specifically, to map input data to PDEs (both linear and nonlinear) to their solutions [44–46]. The deep operator neural networks (DeepONets) for approximating operators presented in [46] are capable of accurately approximating operators that map data taken from a prescribed input space (e.g., initial conditions, forcing terms), sampled at discrete "sensor" locations, to the solutions. Neural networks designed to approximate operators also have the potential to circumvent the high computational cost associated with performing repeated simulations using traditional scientific computing methods (e.g., finite-difference, finite-element) when solving parametric PDEs. One major area of disadvantage for neural networks compared to traditional scientific computing techniques is extrapolation. Since scientific computing aims

to directly solve the underlying PDE, it is straightforward, for example, to obtain solutions for two different temporal domains. In contrast, when working with deep neural networks, extrapolation beyond the temporal training interval can lead to significant degradation of the solution accuracy.

Historically, deep neural network applications to PDEs have relied primarily on the large availability of data to learn approximations without regard for any additional problem-specific constraints obeyed by the data (e.g., physical laws, domain knowledge). To address this lack of enforcement of constraints during training, domain-aware concepts, such as physics-informed machine learning (e.g., see [38] and references therein) and, more broadly, scientific machine learning (e.g., see [3] and references therein) have been receiving increasing attention. These concepts aim to leverage physical laws, domain knowledge, etc., to improve the performance of deep learning algorithms. By embedding this knowledge into deep learning models, the amount of data required for training and the training times are reduced in many cases. Furthermore, requiring the output to satisfy the underlying physical laws leads to improvements in the approximation accuracy [76] over purely data-driven neural networks while also improving the interpretability of the model [3].

1.3 Research objectives

The focus of this work is the development of two frameworks for the evolution of PDEs and seeks to satisfy two research objectives. The first, outlined in research objective one, seeks to develop a reduced order modeling framework for multiscale PDEs through the MZ formalism, renormalization, and the introduction of a parameter for the control of the time decay of the memory term that encodes knowledge about the relationships between the resolved and unresolved scales. The second, outlined in research objective two, seeks to develop a scientific machine learning framework that utilizes DeepONets for the construction of basis functions that are custom-made for the PDE of interest, suitable for usage in a spectral method, and allows for the potential of evolving a lower-dimensional system through the specification of a truncation threshold. The two research objectives are further outlined below:

1. The MZ formalism, in conjunction with various memory approximations, has been used to successfully construct reduced order models of various PDEs (see discussion in Section 1.1). In this work, the concept of a time-dependent perturbative renormalization framework was developed for ROMs constructed using the MZ formalism and investigated as a means to construct stable ROMs, which also accurately reproduce the key solution characteristics for long times. The time dependence is controlled by introducing a parameter that allows for the control of the time decay of the memory and can be selected based on limited data from a well-resolved full simulation. This approach was investigated for two multiscale PDEs; the one-dimensional inviscid Burgers equation and the three-dimensional Euler equations. Chapter 2 is devoted to this research objective.

2. The DeepONet architecture has been used to successfully map data taken from a given input function space to the solution of a PDE (see discussion in Section 1.2). In this work, a scientific machine learning framework was developed and investigated, which couples the strengths of deep neural networks for the approximation of operators, the SVD for determining dominant features in data, and the high order of convergence of spectral methods as a means to accurately evolve PDEs for temporal intervals and input function spaces, both within and outside the deep neural network training domain. More specifically, DeepONets are utilized to identify candidate custom-made basis functions, which are then ordered and orthonormalized through the SVD, and subsequently used as the basis for which to expand the solution of a PDE. Furthermore, the usage of the SVD allows for a truncation threshold to be established so that a reduction in model size may be achievable. This approach was investigated for six one-dimensional PDEs, two of which are linear and four that are nonlinear. Chapter 3 is devoted to this research objective.

A summary of the contributions of this work and potential avenues for additional future work are presented in Chapter 4. Further details and supplemental material for the first

research objective can be found in Appendices A, B, C, and D, while further details and supplemental material for the second research objective can be found in Appendices E, F, G, H, I, J, K, L, and M.

Chapter 2

OPTIMAL RENORMALIZATION OF MULTISCALE SYSTEMS

In this chapter, the work completed toward the first research objective pertaining to time-dependent perturbative renormalization is presented. In Section 2.1, a derivation of the MZ formalism is provided. Section 2.2 outlines the particular approximation utilized for the memory term, while Section 2.3 presents the framework for time-dependent perturbative renormalization using data from a well-resolved but limited full simulation. This framework is applied to two PDEs in Section 2.4, the one-dimensional inviscid Burgers equation (Section 2.4.1) and the three-dimensional Euler equations (Section 2.4.2).

An earlier version of this work was published in the Proceedings of the National Academy of Sciences [59], and the library of MATLAB [49] codes expanded as part of this work are publicly available at: `https://github.com/brekmeuris/Renormalized_Mori_Zwanzig`

2.1 Derivation of the Mori–Zwanzig equation

Consider a system of ordinary differential equations (ODEs),

$$\frac{d\mathbf{u}(t)}{dt} = \mathbf{R}(\mathbf{u}(t)), \quad \mathbf{u}(0) = \mathbf{u}_0, \tag{2.1}$$

where $\mathbf{u}(t) = \{u_k(t)\}$, $k \in F \cup G$. The variables in F will be considered the resolved variables while the set of variables in G will be considered the unresolved variables such that $\hat{\mathbf{u}} = \{u_k(t)\}$, $k \in F$, and $\tilde{\mathbf{u}} = \{u_k(t)\}$, $k \in G$. A system of ODEs, such as that shown in Equation (2.1), may result when utilizing an appropriate set of basis functions and the Galerkin condition to convert a PDE into a system of ODEs—as commonly done when using a spectral method [9, 11, 34]. It can be shown that this system of ODEs can be transformed

into a system of linear PDEs [14] given by

$$\frac{du_k(t)}{dt} = \frac{\partial}{\partial t} e^{t\mathcal{L}} u_{0k} = e^{t\mathcal{L}} \mathcal{L} u_{0k} \tag{2.2}$$

with $k \in F \cup G$ and $\mathcal{L}$, the Liouville operator, given by

$$\mathcal{L} = \sum_{k \in F \cup G} R_k(\mathbf{u}_0) \frac{\partial}{\partial u_{0k}}. \tag{2.3}$$

Refer to Appendix A for additional details about this claim.

Define P to be an orthogonal projection onto the space of functions that only depend on the resolved variables $\hat{\mathbf{u}}_0$, with complementary projector, $Q = I - P$. Equation (2.2) can then be rewritten as

$$\frac{du_k(t)}{dt} = e^{t\mathcal{L}} P \mathcal{L} u_{0k} + e^{t\mathcal{L}} Q \mathcal{L} u_{0k}. \tag{2.4}$$

The evolution operator $e^{t\mathcal{L}}$ can be decomposed through the use of Dyson's formula [16],

$$e^{t\mathcal{L}} = e^{tQ\mathcal{L}} + \int_0^t e^{(t-s)\mathcal{L}} P \mathcal{L} e^{sQ\mathcal{L}} ds. \tag{2.5}$$

Using Equation (2.5) to decompose the second term in Equation (2.4) yields

$$\frac{du_k(t)}{dt} = \underbrace{e^{t\mathcal{L}} P \mathcal{L} u_{0k}}_{Markov} + \underbrace{e^{tQ\mathcal{L}} Q \mathcal{L} u_{0k}}_{Noise} + \underbrace{\int_0^t e^{(t-s)\mathcal{L}} P \mathcal{L} e^{sQ\mathcal{L}} Q \mathcal{L} u_{0k} ds}_{Memory}, \tag{2.6}$$

which is the Mori–Zwanzig (MZ) equation. The Mori–Zwanzig equation is an exact representation of the original PDE and serves as a starting point for constructing reduced order models. The first term on the right-hand side is called the Markov term, as it depends only on the instantaneous values of the resolved variables at the current time. The second term is called the noise term, and the third term is the memory term.

The projection P, was specified as $Pf(\mathbf{u}_0) = Pf(\hat{\mathbf{u}}_0, \tilde{\mathbf{u}}_0) = f(\hat{\mathbf{u}}_0, 0)$, and has three important features [61]. First, it commutes with nonlinear functions. Second, the Markov

term that results from the MZ formalism matches the term that results from a Galerkin spectral approach (if the unresolved variables were set to zero for all time) for the right-hand side of the ODE. Third, when the projector is applied to Equation (2.6), the noise term is eliminated. Utilizing the specified projection P, and projecting a second time,

$$\frac{dPu_k(t)}{dt} = Pe^{t\mathcal{L}}P\mathcal{L}u_{0k} + P\int_0^t e^{(t-s)\mathcal{L}}P\mathcal{L}e^{sQ\mathcal{L}}Q\mathcal{L}u_{0k}ds, \tag{2.7}$$

where

$$Pe^{tQ\mathcal{L}}Q\mathcal{L}u_{0k} = 0, \tag{2.8}$$

i.e., the noise term vanishes. Equation (2.7) now represents the projected dynamics of the resolved variables and gives the average behavior of u_k. Furthermore, Equation (2.7) is suitable for the prediction of each trajectory that originates from the chosen initial condition for the resolved variables (again, unresolved variables set to zero).

To solve a system of equations using Equation (2.7) requires the computation of the Markov term and the memory term. However, due to the presence of the orthogonal dynamics operator $e^{sQ\mathcal{L}}$, the system is not closed, and simulating the memory term requires knowledge of the dynamics of the unresolved variables. It is the presence of the orthogonal dynamics operator that makes reduced order models based directly on the MZ formalism prohibitively expensive. Despite the complexity of the memory term, dropping it entirely and simulating only the Markov term may not accurately reproduce the true dynamics of the resolved variables in a full simulation. Therefore, any multiscale, reduced order model must address the memory term or provide justification for the ability to safely neglect it.

2.2 Approximating the memory term

Due to the complexity of the resulting memory term and its dependence on the unresolved variables, various approximation techniques have been developed, see [17, 18, 33, 69–71] for additional details. A more recent approximation, referred to as the complete memory approximation (CMA), originally outlined in [60, 61], was utilized to approximate the memory

term, and the details of this approximation follow. To facilitate the construction of approximate models of varying order for the memory term, first, define the Markov term as the zeroth-order term,

$$R_k^0(\hat{\mathbf{u}}) = Pe^{t\mathcal{L}}P\mathcal{L}u_{0k}, \tag{2.9}$$

and the memory term as

$$\mathcal{M}_k = P\int_0^t e^{(t-s)\mathcal{L}}P\mathcal{L}e^{sQ\mathcal{L}}Q\mathcal{L}u_{0k}ds. \tag{2.10}$$

There are two operators present in Equation (2.10), the aforementioned orthogonal dynamics operator $e^{sQ\mathcal{L}}$, and the full dynamics operator $e^{(t-s)\mathcal{L}}$, and each of these operators may evolve on their own timescales. For the approximation presented, an absence of timescale separation was assumed.

The simplest memory approximation model, commonly referred to as the "t-model"—see [5,17,33,33,69] for the application of the t-model to a variety of problems—is constructed by assuming that the integrand in Equation (2.10) is constant, which yields

$$\mathcal{M}_k \approx tPe^{t\mathcal{L}}P\mathcal{L}Q\mathcal{L}u_{0k}. \tag{2.11}$$

The CMA improves on the approximation provided by the t-model by constructing a series representation of Equation (2.10) in powers of t. Assuming analyticity of $e^{-s\mathcal{L}}$ and $e^{sQ\mathcal{L}}$, Equation (2.10) can be expanded around $s = 0$, using Taylor series to obtain

$$\begin{aligned}\mathcal{M}_k &= Pe^{t\mathcal{L}}\int_0^t\left(\sum_{i=0}^{\infty}\frac{(-1)^i s^i}{i!}\mathcal{L}^i\right)P\mathcal{L}\left(\sum_{j=0}^{\infty}\frac{s^j}{j!}(Q\mathcal{L})^j\right)Q\mathcal{L}u_{0k}ds,\\ &= Pe^{t\mathcal{L}}\left(\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}\frac{(-1)^i t^{i+j+1}}{i!j!(i+j+1)}\mathcal{L}^i P\mathcal{L}(Q\mathcal{L})^j Q\mathcal{L}u_{0k}\right),\end{aligned} \tag{2.12}$$

where it is assumed that the integrand is sufficiently smooth so that the order of the integral and the infinite sums may be exchanged in the second expression [61]. Writing out the

expansion for the first two terms yields

$$\mathcal{M}_k = \underbrace{tPe^{t\mathcal{L}}P\mathcal{L}Q\mathcal{L}u_{0k}}_{t-model} - \frac{1}{2}t^2Pe^{t\mathcal{L}}[\underbrace{\mathcal{L}P\mathcal{L}Q\mathcal{L}}_{A} - \underbrace{P\mathcal{L}Q\mathcal{L}Q\mathcal{L}}_{B}]u_{0k} + \mathcal{O}(t^3), \tag{2.13}$$

where the first term ($\mathcal{O}(t)$) in Equation (2.13), is again the t-model as found previously. For the second term ($\mathcal{O}(t^2)$) in Equation (2.13), there are two terms to deal with, indicated as A and B. The B term is projected prior to the Liouville operator so that the evolution only depends on the resolved variables. The A term is more problematic as it is not projected prior to the Liouville operator. This means that when the evolution operator $e^{t\mathcal{L}}$ is applied, the A term requires knowledge of the dynamics of both the resolved and the unresolved variables and leads back to a system that is again no longer closed in the resolved variables. To close this system, note that the $e^{t\mathcal{L}}\mathcal{L}P\mathcal{L}Q\mathcal{L}u_{0k}$ term is the evolution of $e^{t\mathcal{L}}P\mathcal{L}Q\mathcal{L}u_{0k}$ [70], and may be written as

$$\frac{\partial}{\partial t}e^{t\mathcal{L}}P\mathcal{L}Q\mathcal{L}u_{0k} = e^{t\mathcal{L}}\mathcal{L}P\mathcal{L}Q\mathcal{L}u_{0k}, \tag{2.14}$$

which matches the form presented in Equation (2.2). Projecting yields

$$\frac{\partial}{\partial t}Pe^{t\mathcal{L}}P\mathcal{L}Q\mathcal{L}u_{0k} = Pe^{t\mathcal{L}}\mathcal{L}P\mathcal{L}Q\mathcal{L}u_{0k}, \tag{2.15}$$

to which the MZ formalism can again be applied,

$$\begin{aligned}\frac{\partial}{\partial t}Pe^{t\mathcal{L}}P\mathcal{L}Q\mathcal{L}u_{0k} &= Pe^{t\mathcal{L}}P\mathcal{L}P\mathcal{L}Q\mathcal{L}u_{0k} + P\int_0^t e^{(t-s)\mathcal{L}}P\mathcal{L}e^{sQ\mathcal{L}}Q\mathcal{L}P\mathcal{L}Q\mathcal{L}u_{0k}ds, \\ &= Pe^{t\mathcal{L}}P\mathcal{L}P\mathcal{L}Q\mathcal{L}u_{0k} + \mathcal{O}(t).\end{aligned} \tag{2.16}$$

This can now be substituted back into Equation (2.13) for term A to obtain an expression which is $\mathcal{O}(t^2)$ and closed in the resolved variables,

$$\mathcal{M}_k = tPe^{t\mathcal{L}}P\mathcal{L}Q\mathcal{L}u_{0k} - \frac{1}{2}t^2Pe^{t\mathcal{L}}P\mathcal{L}\left[P\mathcal{L} - Q\mathcal{L}\right]Q\mathcal{L}u_{0k} + \mathcal{O}(t^3), \tag{2.17}$$

where the $\mathcal{O}(t)$ term in Equation (2.16) is omitted as it contributes at $\mathcal{O}(t^3)$. This process can then be repeated for the construction of higher-order terms and was automated in [60].

In general, the CMA for the memory term can be written as

$$\mathcal{M}_k = \sum_{i=1}^{\infty} \frac{(-1)^{i+1} t^i}{i!} R_k^i(\hat{\mathbf{u}}). \tag{2.18}$$

Under this convention, the first-order contribution to the memory term is written as

$$R_k^1(\hat{\mathbf{u}}) = P e^{t\mathcal{L}} P\mathcal{L}Q\mathcal{L}u_{0k}, \tag{2.19}$$

and the second-order $\mathcal{O}(t^2)$ contribution to the memory term is written as

$$R_k^2(\hat{\mathbf{u}}) = P e^{t\mathcal{L}} P\mathcal{L}\left[P\mathcal{L} - Q\mathcal{L}\right] Q\mathcal{L}u_{0k}. \tag{2.20}$$

The third-order $\mathcal{O}(t^3)$ contribution to the memory is then written as

$$R_k^3(\hat{\mathbf{u}}) = P e^{t\mathcal{L}} P\mathcal{L}\left[P\mathcal{L}P\mathcal{L} - 2P\mathcal{L}Q\mathcal{L} - 2Q\mathcal{L}P\mathcal{L} + Q\mathcal{L}Q\mathcal{L}\right] Q\mathcal{L}u_{0k}, \tag{2.21}$$

while the fourth-order contribution $\mathcal{O}(t^4)$ is written as

$$\begin{aligned} R_k^4(\hat{\mathbf{u}}) = P e^{t\mathcal{L}} P\mathcal{L}\big[&P\mathcal{L}P\mathcal{L}P\mathcal{L} - 3P\mathcal{L}P\mathcal{L}Q\mathcal{L} - 5P\mathcal{L}Q\mathcal{L}P\mathcal{L} - 3Q\mathcal{L}P\mathcal{L}P\mathcal{L} \\ &+3P\mathcal{L}Q\mathcal{L}Q\mathcal{L} + 5Q\mathcal{L}P\mathcal{L}Q\mathcal{L} + 3Q\mathcal{L}Q\mathcal{L}P\mathcal{L} - Q\mathcal{L}Q\mathcal{L}Q\mathcal{L}\big] Q\mathcal{L}u_{0k}. \end{aligned} \tag{2.22}$$

The resulting differential equation for each resolved variable can then be expressed as

$$\frac{dPu_k}{dt} = R_k^0(\hat{\mathbf{u}}) + \mathcal{M}_k, \tag{2.23}$$

where the $R_k^0(\hat{\mathbf{u}})$ is given by Equation (2.9).

2.3 Optimal renormalization for long time prediction

The complete memory approximation (Equation (2.18)) assumes the various evolution operators are analytic in time so that Taylor series expansions may be utilized to expand the memory term (refer to Section 2.2 for details). In practice, direct applications of the CMA and simpler memory approximations have produced reduced order models which are unstable [60, 61, 70, 71]. The development of an instability is not completely unexpected though, due to the usage of Taylor expansions around $s = 0$ (the current time t), which are series representations of the past. Continually increasing t requires looking further into the past, and since only the first few terms of the Taylor expansions are utilized, these approximations are no longer expected to be a very good way to account for the distant past. To combat the development of an instability, the process of renormalization [22] has been used to stabilize the models referenced above by augmenting each of the terms in the memory approximation by an additional coefficient, such that each term represents an effective memory. The resulting differential equation for each resolved variable is then written as

$$\frac{dPu_k}{dt} = R_k^0(\hat{\mathbf{u}}) + \sum_{i=1}^{n} \alpha_i(t) t^i R_k^i(\hat{\mathbf{u}}), \tag{2.24}$$

where $\alpha_i(t)$ are the time-dependent renormalization coefficients and ultimately replace the $\frac{(-1)^{i+1}}{i!}$ term in Equation (2.18).

To determine the appropriate coefficients for the renormalization process, a full system of size M' must be simulated, time steps for which the full simulation is well-resolved must be determined, an appropriate metric to match between the full system and the ROM must be selected, and an ansatz for the form of the renormalization coefficients must be developed. In essence, the goal of the renormalization procedure is to attempt to extend the range of validity of the ROMs given by Equation (2.23) by assuming that the functional form of the expansion is correct and adjusting the coefficients so that the ROM produces the same values for a known physical quantity as a well-resolved full simulation.

For the models considered as part of this work (one-dimensional inviscid Burgers and three-dimensional Euler equations), it was found that if the models are not renormalized, the resulting simulations are unstable for all models except the t-model. To renormalize these models, a Fourier expansion $u(t,x) = \sum_{k\in F\cup G} u_k(t)e^{ikx}$, was utilized to obtain a system of ODEs for each model. To conform with the notation outlined in Section 2.1, let $F = [-N+1,\ldots,N-1]$ and $G = [-M+1,\ldots,-N,N,\ldots,M-1]$ for $N < M$. Furthermore, let $\mathbf{u} = \{u_k\}_{k\in F\cup G}$, $\hat{\mathbf{u}} = \{u_k\}_{k\in F}$, and $\tilde{\mathbf{u}} = \{u_k\}_{k\in G}$. For the models considered, $M = 2N$ due to the quadratic nonlinearity. Note this notation is for the one-dimensional case. Refer to Section 2.4.2 for details in higher dimensions. Figure 2.1 shows representative results for the evolution of the energy contained in the resolved modes ($N = 12$) of first- through fourth-order reduced order models without renormalization for the inviscid Burgers equation $\partial u/\partial t + u\partial u/\partial x = 0$, on the periodic domain $x \in [0, 2\pi]$, and with the initial condition $u_0(x) = \sin(x)$. The reduced order model is compared to a well-resolved shock-capturing scheme (refer to Section 2.4.1 for additional details).

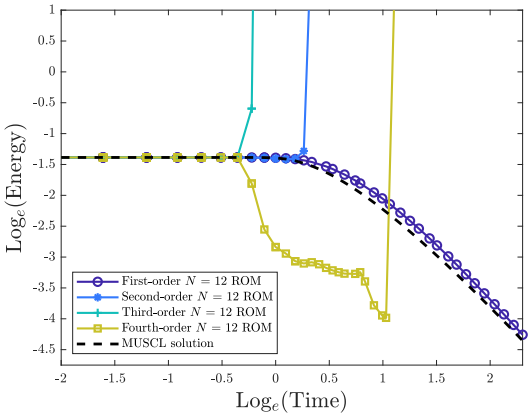


Figure 2.1: Energy contained in the resolved modes of order $n = 1, 2, 3, 4$ ROMs of the inviscid Burgers equation for $N = 12$ without renormalization.

The rates of change of energy in each of the resolved modes $E_k(t) = |u_k|^2$, was specified as the metric to match between the full system and the ROMs. This choice of metric proceeds from the knowledge that energy moves from low-frequency to high-frequency modes as a

shock develops (in the case of inviscid Burgers), and the Markov term on its own is incapable of capturing this phenomenon since it conserves energy in the resolved modes. The total energy is defined as [15]

$$E(t) = \frac{1}{2}\sum_{k} |u_k|^2, \tag{2.25}$$

and the rate of change of energy of a particular mode for the full system is

$$\Delta E_k(t) = R_k(\mathbf{u})\overline{u}_k + u_k\overline{R}_k(\mathbf{u}). \tag{2.26}$$

The rate of change of energy of a particular mode of the reduced system will consist of contributions from all terms given in Equation (2.24). The termwise contribution to the rate of change of energy for a particular mode of the reduced system is

$$\Delta E_k^i(t) = R_k^i(\hat{\mathbf{u}})\overline{u}_k + u_k\overline{R}_k^i(\hat{\mathbf{u}}). \tag{2.27}$$

To determine the renormalization coefficients, a full system of size $M' \geq 2N$ was simulated, where N is the size of the desired ROM. The fact that the Markov term conserves energy in the resolved modes means that computing the total energy flowing through the t-model at each step,

$$\Delta E_{F'}^1(t) = \frac{1}{2}t\sum_{k\in F'} R_k^1(\hat{\mathbf{u}})\overline{u}_k + u_k\overline{R}_k^1(\hat{\mathbf{u}}), \tag{2.28}$$

allows for the determination of the times for which the model is still well-resolved and where setting $F' = \{k \in [-M'/2+1, M'/2-1]\}$, provides a conservative estimate. To fit the renormalization coefficients through the matching of the rates of change of energy between the full and reduced model, the full solution must be well-resolved, but at the same time, the magnitudes of the rates of change of energy must also be greater than machine precision when performing calculations in double precision. The criterion utilized for determining the snapshots where the full system is well-resolved was specified as $\Delta E_{F'}^1(t) < 10^{-10}$, and only the time instances where this criterion was satisfied were stored, $T^* = \{t \in [0, t^*]\}$.

In an earlier version of this work [59], it was posited that the time-dependent renormalization coefficients took the form

$$\alpha_i(t) = a_i t^{-i\tau}, \tag{2.29}$$

where τ was allowed to continually vary between $\tau = 0$, which corresponds to the t-model and higher-order versions thereof, and $\tau = 1$, where there is no explicit time dependence on the memory. After collecting the required snapshots,

$$C_{N,n}(\mathbf{a}, \tau) = \sum_{k \in F} \sum_{t \in T^*} \left(\Delta E_k - \Delta E_k^0 - \sum_{i=1}^{n} a_i t^{-i\tau} t^i \Delta E_k^i \right)^2 \tag{2.30}$$

was minimized, where $\mathbf{a}$ is a vector of the prefactors. To determine the value of τ, a search procedure was performed over the range $[0, 1]$ with an increment of 0.01 between each successive value. For each value of τ, the prefactors $\mathbf{a}$, were fit by minimizing Equation (2.30) and tabulating the corresponding error and τ value. Once the entire range of τ values had been traversed, the τ value and corresponding prefactors resulting in the minimum error were extracted for evolving the ROM.

This earlier work highlighted several key points. First, a full simulation of size $M' >> M$ is required to estimate the optimal value of τ. A large value of M' is required so that the full system can advance in time far enough that a robust transfer of energy from the resolved to the unresolved modes can be established. Second, it was found the minimization problem presented by Equation (2.30) suffered from collinearity and manifested as ill-conditioning for the least-squares problem. Third, it was found that if the value of τ is fixed near the optimal value and the prefactors $\mathbf{a}$, are plotted as a function of N, robust scaling laws $a_i = \beta_i^n N^{\gamma_i^n}$, are obtained. Furthermore, from the scaling laws, it was found that the exponents γ_i^n were relatively independent of the number of memory terms, indicating that each additional memory term was making corrections to the previously captured behavior and adding additional physics not present in the previous terms. Additionally, it was determined that $\gamma_i^n \approx -i$. Fourth, the magnitude of the prefactors decreased with N, which indicated

that the renormalized expansion was indeed perturbative, with $1/N$ serving as the small quantity for the perturbative expansion. Fifth, the signs of the renormalized coefficients agreed with the coefficients in Equation (2.18) $\frac{(-1)^{i+1}}{i!}$, but were different in magnitude.

After further analysis of the results presented in [59], it was found that the first and second points regarding the highly delicate nature of accurately estimating τ and the ill-conditioning of the least-squares problem warranted further investigation. In an effort to obtain more reliable and robust estimates of τ, the initial ansatz for the time-dependent renormalization coefficients was revised to

$$\alpha_i(t) = \beta_i \left[(1/N)t^{-\tau}\right]^i . \tag{2.31}$$

Factoring out $(1/N)^i$ from the previous prefactors $\mathbf{a}$, results in a reduced condition number for the least-squares problem. Additionally, Equation (2.30) was revised to include a global error component in addition to the previously utilized local error

$$\begin{aligned} C_{N,n}(\beta,\tau) = &\sum_{k\in F}\sum_{t\in T^*}\left(\Delta E_k - \Delta E_k^0 - \sum_{i=1}^{n}(1/N)^i\,\beta_i t^{-i\tau}t^i\Delta E_k^i\right)^2 \\ &+ \sum_{t\in T^*}\left[\sum_{k\in F}\Delta E_k - \sum_{k\in F}\Delta E_k^0 - \sum_{i=1}^{n}(1/N)^i\,\beta_i t^{-i\tau}t^i\sum_{k\in F}\Delta E_k^i\right]^2 . \end{aligned} \tag{2.32}$$

The search procedure used previously and an optimization routine were investigated for estimating the value of τ and the β_i values. For the optimization routine, a simplex [39] search method was utilized with additional penalty terms added to Equation (2.32) to enforce the correct signs on each of the prefactors β_i, and to enforce τ to be ≥ 0.

2.4 Applications and numerical results

In this section, results are presented for the one-dimensional inviscid Burgers (Section 2.4.1) equation and the three-dimensional Euler equations (Section 2.4.2). The one-dimensional

inviscid Burgers equation and the three-dimensional Euler equations were chosen to investigate the effectiveness of the time-dependent perturbative renormalization approach as they are both multiscale in nature and share a similar quadratic nonlinearity. The framework is applied first to the inviscid Burgers equation—which develops a finite-time singularity and for which the solution is well known—and serves as a proof-of-concept. The framework is then applied to the three-dimensional Euler equations, where the formation of a finite-time singularity, when initialized from a smooth initial condition, remains an open question. Furthermore, modern, high-resolution, brute force simulations are only feasible for short times.

2.4.1 Application 1: 1D Inviscid Burgers equation

Consider the inviscid form of the one-dimensional Burgers equation,

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0, \tag{2.33}$$

on the periodic domain $x \in [0, 2\pi]$, and with initial condition $u_0(x) = \sin(x)$. The inviscid Burgers equation is often viewed as a simplified version of the Euler equations [58] and used as a model for developing both theory and numerical methods [43]. Due to the lack of any regularization accompanying the quadratic nonlinearity, the inviscid Burgers equation is capable of developing discontinuities in the velocity field (shocks) in finite time when initialized from smooth initial conditions [4]. For the initial condition specified, this problem produces a singularity in the form of a standing shock at $T = 1$, and for times beyond $T = 1$, the shock dominates the dynamics of the system. In the absence of any regularization to prevent the formation of shocks, a standard Galerkin spectral method based on a Fourier expansion will not converge to the true solution. This lack of convergence is due to the fact that the spectral method conserves the L^2 norm of the solution [5, 68] and there is no mechanism to account for the energy that is being consumed by the shock. To account for the drain of energy from the resolved modes and to accurately capture the evolution of the energy, the system needs to be augmented with a memory term.

A Fourier expansion,

$$u(t,x) = \sum_{k \in F \cup G} u_k(t) e^{ikx}, \tag{2.34}$$

was utilized to convert the PDE into a system of ODEs for the expansion coefficients. Following the conventions in Section 2.1, $\mathbf{u} = \{u_k(t)\}_{k \in F \cup G}$, $\hat{\mathbf{u}} = \{u_k(t)\}_{k \in F}$, and $\tilde{\mathbf{u}} = \{u_k(t)\}_{k \in G}$. $F = \{k \in [-N+1, \ldots, N-1]\}$, $G = [-M+1, \ldots, -N, N, \ldots, M-1]$ and $M = 2N$. As required by the choice of projector, the specified initial condition lies entirely in the projected domain. The equation of motion for a given Fourier mode is then

$$\frac{du_k}{dt} = R_k(\mathbf{u}) = -\frac{ik}{2} \sum_{\substack{p+q=k \\ p,q \in F \cup G}} u_p u_q. \tag{2.35}$$

All convolution sums resulting from the quadratic nonlinearity were dealiased using the 3/2 rule [11]. Each memory order term $R_k^i(\hat{\mathbf{u}})$, required for the right-hand side of Equation (2.24) can be found in Appendix B.1. All systems of differential equations were solved using a Runge–Kutta–Dormand–Prince integrator with adaptive step size and absolute error tolerance 10^{-14}.

The predictions for the ROMs are compared to the first N modes of a monotonic upstream-centered scheme for conservation laws (MUSCL) solution [58, 83] with $\Delta x = \frac{2\pi}{10000}$, and capable of capturing the shock that develops at $T = 1$. Furthermore, the MUSCL scheme utilized is convergent to the exact slope value of -2 [41] for the long-time evolution of the energy. The slope presented in Figure 2.3 was calculated using the data in the window $15 \leq t \leq 500$, and the relative error was calculated using

$$E_N^n(t) = \frac{\sum_{k \in F} |u_k^{N,n}(t) - u_k(t)|^2}{\sum_{k \in F} |u_k(t)|^2}, \tag{2.36}$$

where $u_k^{N,n}(t)$ is the solution of the n-order ROM of size N and $u_k(t)$ is the MUSCL solution.

To estimate the renormalization coefficients and the value of the optimal τ, $M' = 524288$ and the simplex optimization routine with the additional penalty terms added to Equation

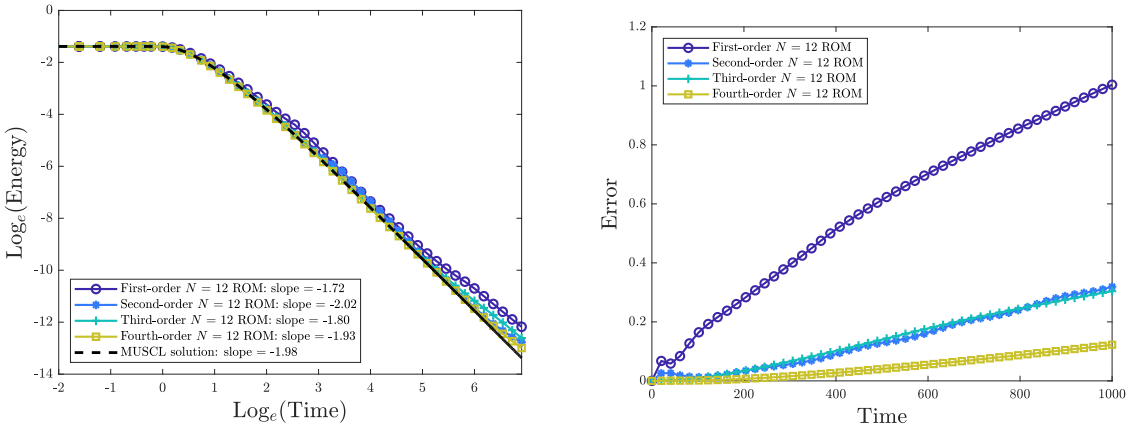
(2.32) were utilized. The optimizer was initialized using $\tau = 0.5$, and the corresponding coefficients were found by solving the least-squares version of Equation (2.32) for the specified τ value. Using the criterion $\Delta E^1_{F'}(t) < 10^{-10}$, $t^* = 0.9990$ for the value of M' specified; therefore, the estimation of the renormalization coefficients is based only on data in advance of the shock forming at $T = 1$. The coefficients and optimal τ value determined for the fourth-order model ($n = 4$) by the optimization routine are shown in Table 2.1 for $N = 6, 8, 10, 12$. Refer to Appendix C.1, Figures C.1, C.2, and C.3 for results showing the convergence of the renormalization coefficients and the optimal τ for various full systems of size M'.

	$N = 6$	$N = 8$	$N = 10$	$N = 12$
β_1	4.1500	4.3639	4.6313	4.8991
β_2	-6.7424	-7.3450	-8.1346	-8.9532
β_3	5.1027	5.7014	6.4053	7.1479
β_4	-1.8049	-1.7817	-1.8946	-2.0619
$(1/N)\beta_1$	6.9166×10^{-1}	5.4548×10^{-1}	4.6313×10^{-1}	4.0826×10^{-1}
$(1/N)^2\beta_2$	-1.8729×10^{-1}	-1.1477×10^{-1}	-8.1346×10^{-2}	-6.2175×10^{-2}
$(1/N)^3\beta_3$	2.3623×10^{-2}	1.1135×10^{-2}	6.4053×10^{-3}	4.1365×10^{-3}
$(1/N)^4\beta_4$	-1.3926×10^{-3}	-4.3499×10^{-4}	-1.8946×10^{-4}	-9.9438×10^{-5}
τ	0.20	0.32	0.38	0.39

Table 2.1: Estimated renormalization coefficients and optimal τ found for the fourth-order ROMs of the inviscid Burgers equation for $N = 6, 8, 10, 12$.

Figure 2.2 shows the energy evolution and relative error for $n = 1, 2, 3, 4$ order ROMs for $N = 12$ and the optimal τ value found for the fourth-order model. Due to the perturbative nature of the ROMs, the fourth-order ROM results are converged, as shown in Figure 2.2. The estimated renormalization coefficients of the $n = 1, 2, 3$ order ROMs for $N = 12$ and for the optimal value of τ found for the fourth-order model can be found in Appendix C.1, Table C.1. Figure 2.3 shows the evolution of the energy contained in the resolved modes of fourth-order ROMs of various size N and the relative error for the temporal domain $t \in [0, 1000]$. For each of the ROMs, the optimal τ value and corresponding coefficients shown in Table 2.1 were utilized. As evidenced by Figure 2.3b, good agreement is obtained between the

MUSCL solution and the ROMs for the full temporal domain when utilizing renormalization coefficients estimated using only data for $t < 1$. A spectral calculation with $M' = 16384$ is also shown for comparison in Figure 2.3a. Despite the large value of M' specified, the spectral calculation loses all predictive capability shortly after $T = 1$.

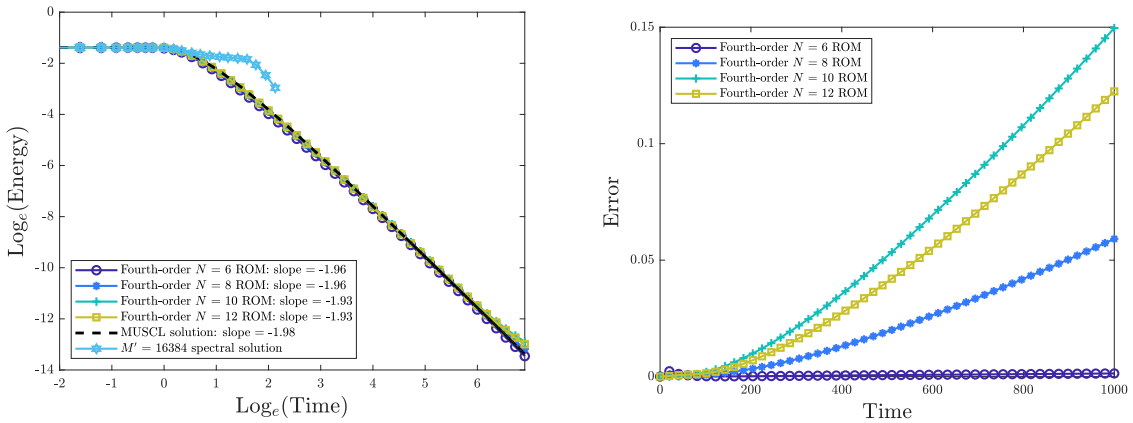


(a) Energy evolution

(b) Relative error

Figure 2.2: Energy contained in the resolved modes (a) and the relative error (b) of order $n = 1, 2, 3, 4$ ROMs of the inviscid Burgers equation for $N = 12$ and the optimal τ found for the fourth-order model.

Figure 2.4 shows the contribution of each of the memory terms to the rate of change of energy in the resolved modes, $\frac{1}{2}\sum_{k\in F}\alpha_i(t)t^i\Delta E_k^i(t)$, for the $N = 12$ fourth-order ROM. The layering of the memory terms can be seen in Figure 2.4, where for each additional higher-order term, the contribution to the total rate of change of energy is reduced. This layering provides further indication that the proposed expansion is perturbative. Furthermore, it was found that the contributions from the first- and third-order terms were negative definite, while the contributions from the second- and fourth-order terms were positive definite. Refer to Appendix C.1, Figure C.4 for predictions showing the point-wise convergence of the fourth-order $N = 12$ ROM in real space for the temporal interval $t \in [0, 10]$.



(a) Energy evolution (b) Relative error

Figure 2.3: Energy contained in the resolved modes (a) and the relative error (b) of fourth-order ROMs of the inviscid Burgers equation, which utilized the optimal τ for $N = 6, 8, 10, 12$.

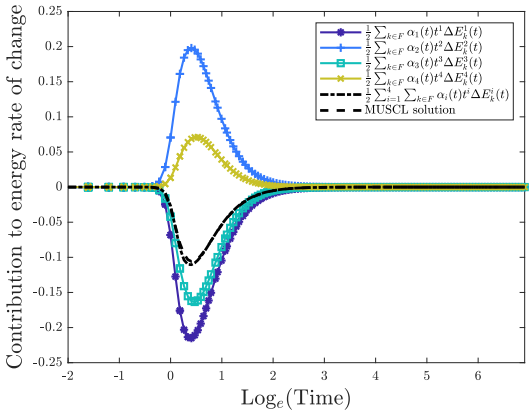


Figure 2.4: The contribution of each order memory term to the rate of change of energy in the resolved modes for the $N = 12$ fourth-order ROM of the inviscid Burgers equation utilizing the optimal τ value.

2.4.2 Application 2: 3D Euler equations

Next, consider the three-dimensional Euler equations,

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p, \quad \nabla \cdot \mathbf{u} = 0, \tag{2.37}$$

on a triply periodic domain in the cube $[0, 2\pi]^3$, and with the smooth Taylor-Green initial condition,

$$\mathbf{u}_0(\mathbf{x}) = \begin{bmatrix} \sin(x_1)\cos(x_2)\cos(x_3) \\ -\cos(x_1)\sin(x_2)\cos(x_3) \\ 0 \end{bmatrix}. \tag{2.38}$$

The three-dimensional Euler equations serve as a model for the evolution of an incompressible, inviscid fluid [54] and one of the long-standing open questions in mathematical fluid dynamics is whether the three-dimensional Euler equations develop a singularity in finite time when initialized from a smooth initial condition [20,25,48]. Refer to [29] and references therein for example studies on the existence of a potential singularity. Despite this open question regarding the formation of a singularity, the fact that there is a cascade of energy from the large to the small scales is well established and conforms with the choice to use the rates of change of energy in each of the resolved modes as the metric to match between the full system and the ROMs for the renormalization procedure. Furthermore, this cascade of energy leads to similar issues as those seen for the inviscid Burgers equations when using a standard Galerkin spectral method based on a Fourier expansion. As the energy is transferred from the large to the small scales and eventually reaches the smallest available resolution, energy begins to accumulate in the smallest scales and eventually causes even current state-of-the-art simulations to become under-resolved after a short time [69]. To account for the cascade of energy and to attempt to accurately capture the evolution of the energy, the system again needs to be augmented with a memory term.

A Fourier expansion,

$$\mathbf{u}(t, \mathbf{x}) = \sum_{\mathbf{k}} \mathbf{u}_{\mathbf{k}}(t) e^{i\mathbf{k}\cdot\mathbf{x}}, \tag{2.39}$$

was utilized to convert the PDE into a system of ODEs for the expansion coefficients, where $\mathbf{k}$ is now a three-dimensional wavevector, and the sum in Equation (2.39) is over all possible wavevectors. Following the conventions in Section 2.1, $\mathbf{u} = \{\mathbf{u}_{\mathbf{k}} \mid \mathbf{k} \in F \cup G\}$, $\hat{\mathbf{u}} = \{\mathbf{u}_{\mathbf{k}} \mid \mathbf{k} \in F\}$, $\tilde{\mathbf{u}} = \{\mathbf{u}_{\mathbf{k}} \mid \mathbf{k} \in G\}$, $F = \{\mathbf{k} \in [-N+1, N-1]^3\}$, and $F \cup G = \{\mathbf{k} \in [-M+1, M-1]^3\}$.

The equation of motion for a given Fourier mode $\mathbf{u_k}$ is then

$$\frac{d\mathbf{u_k}}{dt} = \mathbf{R_k}(\mathbf{u}) = -i \sum_{\substack{\mathbf{p}+\mathbf{q}=\mathbf{k} \\ \mathbf{p},\mathbf{q}\in F\cup G}} \mathbf{k} \cdot \mathbf{u_p} A_\mathbf{k} \mathbf{u_q}, \tag{2.40}$$

where $A_\mathbf{k}$ is the incompressibility projection matrix [24] and defined as

$$A_\mathbf{k} = I - \frac{\mathbf{k}\mathbf{k}^T}{|\mathbf{k}|^2}, \tag{2.41}$$

with I being the identity matrix. All convolution sums resulting from the quadratic nonlinearity were dealiased using the 3/2 rule [11]. Each memory order term $\mathbf{R}^i_\mathbf{k}(\hat{\mathbf{u}})$, required for the right-hand side of Equation (2.24) can be found in Appendix B.2. All systems of differential equations were solved using a Runge–Kutta–Dormand–Prince integrator with adaptive step size and absolute error tolerance 10^{-14}.

The ansatz presented in Equation (2.31), Equation (2.32), and data for $\Delta E_\mathbf{k}(t)$ from a well-resolved full system were used to estimate the renormalization coefficients. As a result of the computational constraints dictating the use of $M' = 48$ in all three directions, the available resolution was not large enough to allow for the establishment of a robust transfer of energy from the resolved to unresolved modes and therefore did not allow for robust estimates for the optimal value of τ. It was found that for both the original cost function (Equation (2.30)) and the revised cost function (Equation (2.32)), the renormalized ROMs were unstable for $\tau \in [0.0, 0.4]$. Refer to Figure 2.5 for results of fourth-order ROMs for $N = 12$ and $\tau = 0.0, 0.2, \ldots, 1.0$ and Table C.2 in Appendix C.2 for the estimated renormalization coefficients for $\tau = 0.0, 0.2, \ldots, 0.8$. This behavior differs from that found for the one-dimensional Burgers equations, indicating the more nuanced nature of the three-dimensional Euler equations, which is likely due to the formation of complex, small-scale structures. Based on the limited available resolution for the full system, which does not allow for the prediction of the optimal τ, and the fact that the behavior of the solution to the three-dimensional Euler equations with a smooth initial condition is unknown [1, 15, 48],

results are presented for $\tau = 1.0$. This choice of τ is based on the goal to produce ROMs that remain stable for long times.

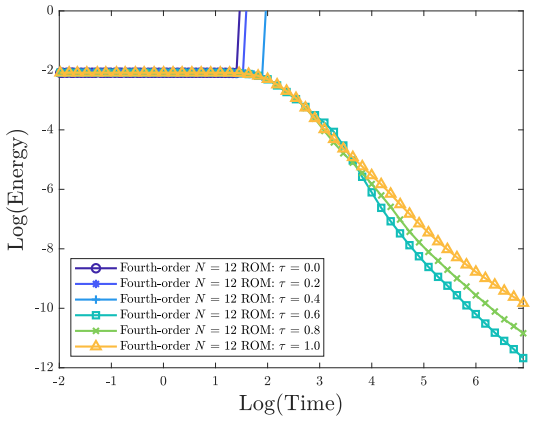


Figure 2.5: Energy contained in the resolved modes of fourth-order ROMs of the 3D Euler equations for $N = 12$ and $\tau = 0.0, 0.2, \ldots, 1.0$.

The renormalization coefficients corresponding to $\tau = 1.0$ are shown in Table 2.2 for $N = 6, 8, 10, 12$. Using the criterion $\Delta E^1_{F'}(t) < 10^{-10}$, $t^* = 1.2016$ for the value of M' specified. Similar to the one-dimensional inviscid Burgers equation, the results for the fourth-order ROMs appear to converge with increasing order as shown in Figure 2.6. The estimated renormalization coefficients for the $n = 1, 2, 3$ order ROMs for $N = 12$ and $\tau = 1.0$ can be found in Appendix C.2, Table C.3. Figure 2.7 shows the evolution of the energy contained in the resolved modes of fourth-order ROMs of various sizes for the temporal domain $t \in [0, 1000]$. Figure 2.7 shows that as time progresses, the results become stratified, which indicates significant activity in the high-frequency modes. This is evidenced by the fact that as the resolution of the ROM increases, the remaining energy in the system decreases. Furthermore, two decay rates in time for the ejection of energy from the resolved modes were found [67]. Table 2.3 shows the initial decay time (time at which 10% of the initial energy has left the resolved modes), the initial decay rate (slope from the data for which 50% to 90% of the energy leaves the resolved modes), and the second decay rate (slope from the data for which over 99.5% of the energy has left the resolved modes). No real space predictions

are presented for the three-dimensional Euler equations due to the small resolutions of the constructed ROMs.

	$N = 6$	$N = 8$	$N = 10$	$N = 12$
β_1	3.7878	3.7986	3.8391	3.8442
β_2	-6.1422	-5.8960	-5.8440	-5.7523
β_3	4.9722	4.2960	4.0344	3.8353
β_4	-1.3646	-1.0546	-0.9316	-0.8580
$(1/N)^1\beta_1$	6.3130×10^{-1}	4.7483×10^{-1}	3.8391×10^{-1}	3.2035×10^{-1}
$(1/N)^2\beta_2$	-1.7062×10^{-1}	-9.2125×10^{-2}	-5.8440×10^{-2}	-3.9947×10^{-2}
$(1/N)^3\beta_3$	2.3020×10^{-2}	8.3906×10^{-3}	4.0344×10^{-3}	2.2195×10^{-3}
$(1/N)^4\beta_4$	-1.0530×10^{-3}	-2.5746×10^{-4}	-9.3160×10^{-5}	-4.1377×10^{-5}

Table 2.2: Estimated renormalization coefficients corresponding to $\tau = 1.0$ found for the fourth-order ROMs of the 3D Euler equations for $N = 6, 8, 10, 12$.

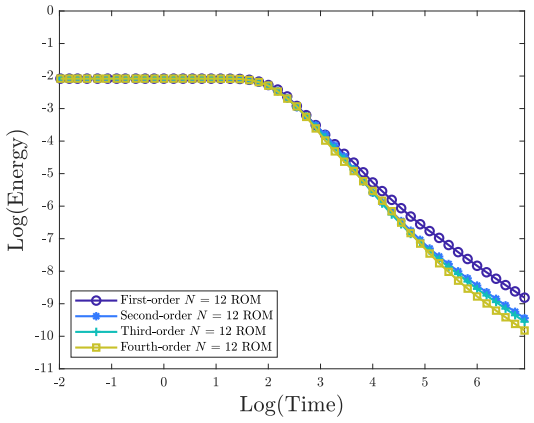


Figure 2.6: Energy contained in the resolved modes of order $n = 1, 2, 3, 4$ ROMs of the 3D Euler equations for $N = 12$ and $\tau = 1.0$.

Figure 2.8 shows the contribution of each of the memory terms to the rate of change of energy in the resolved modes for the $N = 12$ fourth-order ROM. Similar to the results for the one-dimensional inviscid Burgers equation, a layering of the memory terms was found, where with each additional higher-order term, the contribution to the total rate of change of

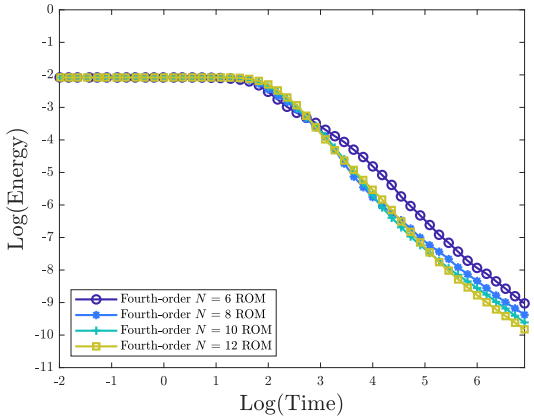


Figure 2.7: Energy contained in the resolved modes of fourth-order ROMs of the 3D Euler equations which utilized $\tau = 1.0$ for $N = 6, 8, 10, 12$.

N	Initial decay time	Initial decay rate	Second decay rate
6	4.8841	-1.0310	-1.2242
8	5.4037	-1.5717	-1.1832
10	5.8205	-1.6432	-1.1582
12	6.1574	-1.9017	-1.2856

Table 2.3: Energy decay rates of fourth-order ROMs of the 3D Euler equations which utilized $\tau = 1.0$ for $N = 6, 8, 10, 12$.

energy is reduced. With the exception of the first-order memory term being negative definite, all higher-order memory terms were not found to have a definite sign. Refer to Figure C.5 and Figure C.6 in Appendix C.2 for plots of the rate of change of energy in the resolved modes for $\tau = 0.6$ and $\tau = 0.8$. The behavior of the solution to the three-dimensional Euler equations initialized from the Taylor-Green initial condition remains unknown, especially for long times. However, recent large-scale direct numerical simulations indicated a peak for the rate of change energy at around time $t = 8 - 9$ [26], which is in good agreement with the presented results that only utilize $N = 12$ resolved modes and are capable of being simulated on a standard workstation computer. Refer to Table 2.4 for peak time and peak rate of change of energy for fourth-order $N = 12$ ROMs for $\tau = 0.6, 0.8, 1.0$. Refer to Tables

C.4, C.5, and C.6 in Appendix C.2 for the peak time and peak rate of change of energy for fourth-order models of size $N = 6, 8, 10$.

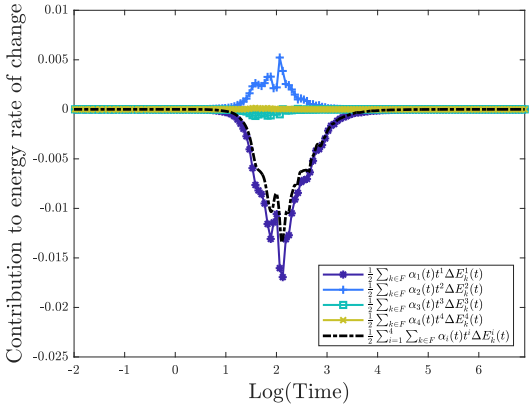


Figure 2.8: The contribution of each order memory term to the rate of change of energy in the resolved modes for the $N = 12$ fourth-order ROM of the 3D Euler equations for $\tau = 1.0$.

τ	Peak time	Peak value
0.6	8.5818	-1.4972×10^{-2}
0.8	8.5535	-1.3510×10^{-2}
1.0	8.2219	-1.3345×10^{-2}

Table 2.4: Peak time and value of the total rate of change of energy in the resolved modes for the $N = 12$ fourth-order ROMs of the 3D Euler equations for $\tau = 0.6, 0.8, 1.0$.

2.5 Summary

Section 2.3 presented the framework for time-dependent perturbative renormalization using data from a well-resolved but limited full simulation to facilitate the stable, long-time evolution of multiscale systems. Section 2.4 illustrated the effectiveness of the presented time-dependent approach for solving multiscale PDEs for long times.

The results presented in Section 2.4.1 served as a proof-of-concept of this approach as the presented results showed good agreement between the ROMs utilizing the estimated

renormalization coefficients and the optimal τ, and a well-resolved shock-capturing scheme for the one-dimensional Burgers equation, as evidenced by Figure 2.3. Furthermore, the fourth-order models were found to be converged (Figure 2.2), and a layering of the memory terms (Figure 2.4) indicated that the expansion was indeed perturbative. This framework was then extended to the three-dimensional Euler equations with results presented in Section 2.4.2. Due to the fact that the solution of the three-dimensional Euler equations initialized from the Taylor-Green initial condition is unknown, results were provided for the stable value $\tau = 1.0$, as a large enough full simulation was not available to facilitate the determination of the optimal τ value. The results for the fourth-order models appear to converge (Figure 2.6) and a layering of the memory terms was again found (Figure 2.8). Presented results were compared to recent large-scale direct numerical simulations [26] and found to be in agreement for the time of the peak rate of change of energy based on an $N = 12$ size ROM that is capable of being simulated up to $t = 1000$ in a few days on a standard workstation computer.

In aggregate, the presented results illustrate the effectiveness of the time-dependent perturbative renormalization framework for solving multiscale PDEs. Furthermore, all the terms present in the ROMs come directly from applying the MZ formalism without the need for the addition of an extra viscous term as done in vanishing viscosity methods [57, 72]. Additionally, when a large enough full simulation is available, the presented approach can be automated to determine both the renormalization coefficients and the optimal τ parameter.

Chapter 3

MACHINE-LEARNING-BASED SPECTRAL METHODS FOR PARTIAL DIFFERENTIAL EQUATIONS

In this chapter, work completed toward the second research objective pertaining to the development of a scientific machine learning framework for the evolution of PDEs is presented. In Section 3.1, an overview of deep neural networks for the approximation of operators based on DeepONets is presented. Section 3.2 presents the method for constructing custom-made basis functions (Section 3.2.1) and an assessment of the approximation capabilities of the custom-made basis functions (Section 3.2.2). Section 3.3 outlines the procedure for evolving PDEs using the custom-made basis functions, while Section 3.4 provides results for the linear advection (Section 3.4.1) and advection-diffusion (Section 3.4.2) equations and the nonlinear viscous Burgers (Section 3.4.3), Korteweg–de Vries (Section 3.4.4), Kuramoto–Sivashinsky (Section 3.4.5), and inviscid Burgers (Section 3.4.6) equations.

An earlier version of this work is available on the arXiv [50], and the library of Julia [6] codes developed as part of this work are publicly available at:
`https://github.com/brekmeuris/DrMZ.jl`

3.1 Deep operator neural networks

Many applications of neural networks and deep learning are based on the universal approximation theorem for functions [21]. However, there also exists a universal approximation theorem for operators [13]. The universal approximation theorem (UAT) outlined in [13] guarantees that a two-layer neural network can approximate a nonlinear operator to arbi-

trary accuracy,

$$\left| \mathcal{G}[s](y) - \sum_{k=1}^{N} \sum_{i=1}^{M} c_i^k \sigma \left(\sum_{j=1}^{m} \xi_{ij}^k s(x_j) + \theta_i^k \right) \sigma(\omega_k \cdot y + \zeta_k) \right| < \epsilon, \tag{3.1}$$

where $\mathcal{G}$ is a nonlinear continuous operator, σ is an activation function, and $c_i^k, \xi_{ij}^k, \zeta_k, \theta_i^k$ are constants. The theoretical result presented in Equation (3.1) was recently made computationally tractable through a specific deep operator neural network (DeepONet) structure [46], which consists of a branch and trunk setup of the form

$$\mathcal{G}_{NN}[s](y) = \sum_{k=1}^{N} \underbrace{b_k[s]}_{Branch} \underbrace{\gamma_k(y)}_{Trunk}, \tag{3.2}$$

where the branch and trunk networks are two separate deep neural networks, and N is the network width.

The choice of two networks is driven by the two separate inputs required to obtain the output $\mathcal{G}_{NN}[s](y)$, the function s, at discrete "sensor" locations x_j's [13, 46], as shown in Equation 3.1, and the desired location of the output y. Depending on the operator being analyzed, y may also be multidimensional, which prevents the direct concatenation of the two inputs and further highlights the need for more than one network. As presented in Equation (3.2), the output of the branch network and the trunk network are multiplied together to provide an approximation of $\mathcal{G}[s](y)$.

All data points used to train the model require three components, the two inputs (s and y) and the ground truth target value to form the triplet $(s, y, \mathcal{G}[s](y))$. The only constraint on the data is that for all input functions s, the sensor locations at which s is provided must be the same. To learn a given operator $\mathcal{G}$, the DeepONet is trained by sampling s from a chosen function space. Then for each s used for training, the corresponding system must be solved to yield the ground truth target value at a specified number of randomly sampled y output locations. Since multiple y values are required for each input function s, the input

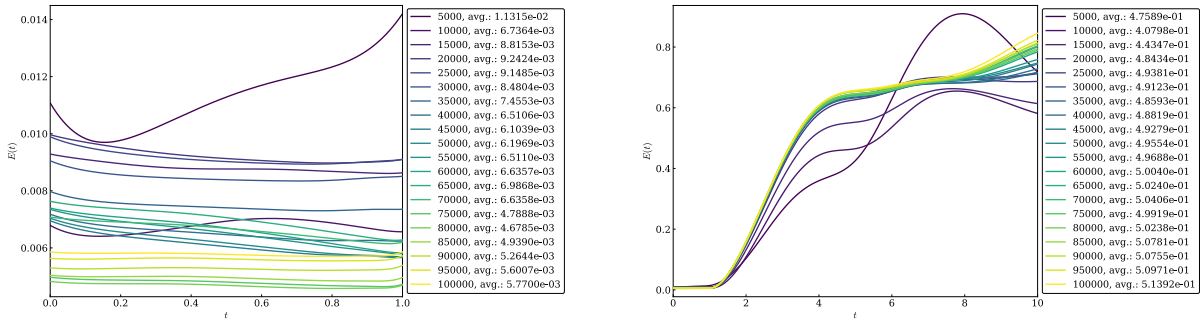
function will appear multiple times in each dataset,

$$
\begin{aligned}
&(s_1, y_1, \mathcal{G}[s_1](y_1)) && \text{Input function } s_1 \\
&(s_1, y_2, \mathcal{G}[s_1](y_2)) && \\
&\qquad\vdots && \\
&(s_1, y_p, \mathcal{G}[s_1](y_p)) && \\
&(s_2, y_1, \mathcal{G}[s_2](y_1)) && \text{Repeat for new input function } s_2 \\
&\qquad\vdots &&
\end{aligned}
$$

where this is repeated for all input functions. For the PDE examples considered, feedforward neural networks were employed for all branch and trunk networks, and the initial condition $u_0(x)$ was specified for the input function s. The values for y were randomly sampled from the (t, x) solution space.

The architecture presented in Equation (3.2) facilitates the accurate solution of PDEs at a reduced computational cost (post-training) compared to more traditional scientific computing methods. Refer to [46] for examples of DeepONets for solving PDEs and [23, 40] for analyses of the errors associated with DeepONets. Figure 3.1 shows representative results of the relative errors for a range of training epochs for a DeepONet trained to solve the advection equation, $\partial u/\partial t + \partial u/\partial x = 0$, on the periodic domain $x \in [0, 2\pi]$, and with the initial condition $u_0(x) = \sin^2(x/2)$. The DeepONet was trained for the temporal domain $t \in [0, 1]$. Figure 3.1a shows that for times within the temporal training domain of $t \in [0, 1]$, the DeepONet learns quickly, and for the number of training epochs shown, the approximated solution is accurate (relative errors $< 1.5\%$). However, as evidenced by Figure 3.1b, once the temporal evaluation interval is extended beyond the temporal training interval, the accuracy of the solution degrades rapidly. The loss of accuracy when attempting to extrapolate beyond the temporal training domain highlighted by Figure 3.1 is not seen as a particular issue with, or specific to the DeepONet approach and architecture, rather, it is an issue commonly seen when working with deep neural networks, especially feedforward neural

networks [80].



(a) $t \in [0, 1]$ (b) $t \in [0, 10]$

Figure 3.1: Relative errors for the solution of the advection equation approximated using a DeepONet trained for $t \in [0, 1]$ and for a range of training epochs (up to 10^5). Temporal evolution interval $t \in [0, 1]$ which lies entirely within training interval (a), temporal evolution interval $t \in [0, 10]$ which requires extrapolation beyond training interval (b).

3.2 Machine-learned custom-made basis functions

The framework presented overcomes the extrapolation issues common to deep neural networks and the required selection of an appropriate choice of basis for a spectral method by utilizing the excellent representational capabilities of DeepONets to identify candidate basis functions. These candidate basis functions are then transformed into a hierarchical, orthonormal basis and coupled with a spectral method to solve PDEs of interest. The construction presented is described in the context of DeepONets; however, in principle, the presented technique can be combined with any operator regression technique that can identify candidate basis functions [44, 45].

3.2.1 Construction of custom-made basis functions

To generate the initial set of candidate basis functions, a DeepONet is trained to approximate the solution operator $\mathcal{G}$, for a PDE on the spatial domain Ω, that maps the initial condition u_0, to the solution. The construction of the candidate basis functions begins by evaluating

the candidate trunk network functions $\{\gamma_k\}_{1\leq k\leq N}$, at $t = 0$, so that the time dependence is fixed and to allow for the generation of a set of frozen-in-time trunk network functions $\{\tau_k\}_{1\leq k\leq N}$. The value $t = 0$ was selected based on the fact that for increasing t, it was found that the spatial variation of a given trunk network function decreases. Figure 3.2 shows the first six frozen-in-time trunk network functions for the advection equation.

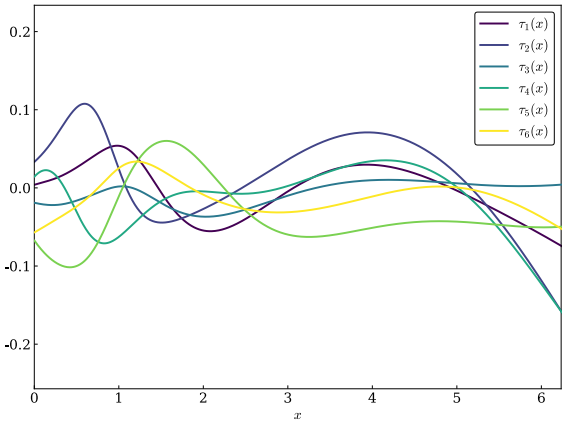


Figure 3.2: Frozen-in-time trunk network functions for the advection equation.

The orthonormal set of basis functions is generated by first defining the L^2 inner product on the spatial domain Ω, by

$$\langle \psi_l, \psi_m \rangle = \int_\Omega \overline{\psi_l(x)} \psi_m(x) dx, \tag{3.3}$$

and the approximation to Equation (3.3) as

$$\langle \psi_l, \psi_m \rangle \approx \sum_{i=1}^{M} \overline{\psi_l(x_i)} \psi_m(x_i) w_i, \tag{3.4}$$

where $\{(x_i, w_i)\}_{1\leq i\leq M}$ is a quadrature rule on Ω and $\langle \psi_l, \psi_m \rangle = \delta_{lm}$, for an orthonormal basis. The matrix A is then constructed by evaluating each of the frozen-in-time trunk network functions at the specified quadrature nodes, such that $A_{ik} = \tau_k(x_i)$. The singular value decomposition (SVD) is then utilized to transform the set of frozen-in-time trunk

network functions into an orthonormal set of basis functions. Generating a set of orthonormal functions is also possible using the well-known Gram–Schmidt procedure [2, 74] through the QR decomposition; however, this procedure is dependent on the initial ordering of $\{\tau_k\}$ and may not ultimately provide a hierarchy.

It is theoretically possible to compute the SVD, $B = USV^*$, of $W^{1/2}A$ and use the singular values $S = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_N)$ along with the right singular vectors $V = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N)$ to construct

$$\phi_k = \sigma_k^{-1} \sum_{l=1}^{N} (\mathbf{v}_k)_l \, \tau_l, \tag{3.5}$$

where $\{\phi_k\}_{1 \le k \le N}$ represent the custom-made hierarchical orthonormal basis for $\mathcal{S} = \text{span}\left(\{\tau_k\}_{1 \le k \le N}\right)$. However, in practice, the division by the singular values—which may decay rapidly—can cause the calculations of ϕ_k based on Equation (3.5) to suffer from large errors.

To circumvent the potentially large errors associated with using Equation (3.5), the construction instead utilizes the product of the left singular vectors $U = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N)$ and $W^{-1/2}$ to obtain the values of $\{\phi_k\}$ at the quadrature nodes

$$\phi_k(x_i) = \left(W^{-1/2}U\right)_{ik} \quad \text{for } 1 \le i \le M, \ 1 \le k \le N. \tag{3.6}$$

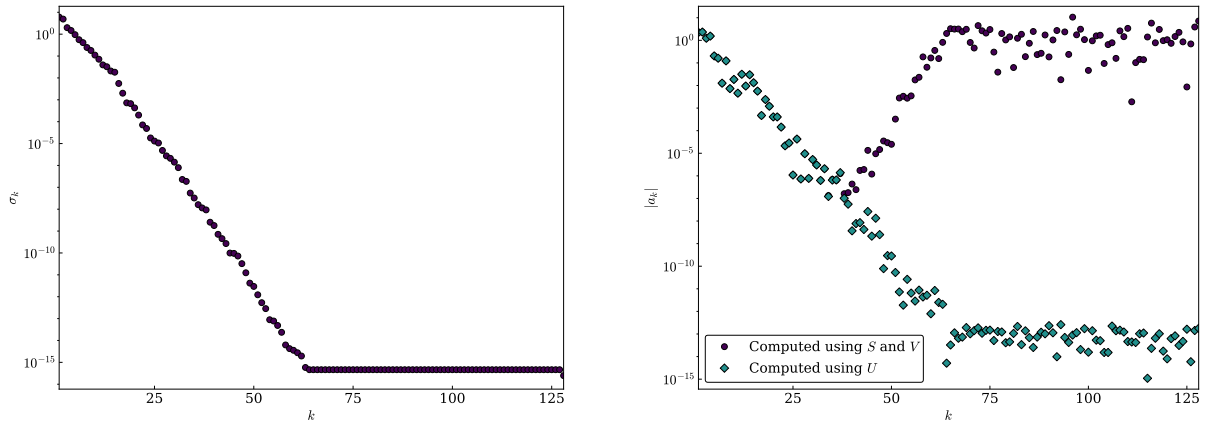
To return to the space of functions and allow for the evaluation of the custom-made basis at locations away from the prescribed quadrature points, an orthogonal polynomial expansion is utilized. Let $\{q_n\}_{0 \le n \le L}$ be the orthonormal Legendre polynomials on Ω with $L < M$ and define $\{\tilde{\phi}_k\}_{1 \le k \le N}$ by

$$\tilde{\phi}_k = \sum_{n=0}^{L} \left(\sum_{i=1}^{M} q_n(x_i) \phi_k(x_i) w_i \right) q_n. \tag{3.7}$$

The process of projecting onto Legendre polynomials and the associated polynomial expansion is guaranteed to be accurate for smooth functions and for large L, serve as a good approximation to $\{\phi_k\}$. Furthermore, for $L = M - 1$, Equation (3.7) is equivalent to polynomial interpolation through $\{\phi_k(x_i)\}_{1 \le i \le M}$. It should be noted this step is only required

for interpolation of the custom-made basis.

Figure 3.3 shows the comparison between the two construction methods presented in Equation (3.5) and Equation (3.7) for the advection equation $\partial u/\partial t + \partial u/\partial x = 0$, on the periodic domain $x \in [0, 2\pi]$, trained for $t \in [0, 1]$, and with $N = 128$. Figure 3.3 highlights the ill-conditioning of the construction method presented in Equation (3.5), which utilizes the S and V matrices of the SVD decomposition. Using Equation (3.5), the decay of the magnitudes of the expansion coefficients $a_k = \left\langle \phi_k, e^{\sin(x)} \right\rangle$, is interrupted around $k \approx 40$, while utilizing the U matrix and Equation (3.7) to construct $\tilde{\phi}_k$ leads to a decay of the expansion coefficients to nearly 10^{-15} using $L = 127$, and with $M = L + 1$. This approach, which utilizes the SVD for orthonormalization of the frozen-in-time trunk network functions evaluated at the specified quadrature nodes, coupled with a polynomial expansion based on the Legendre polynomials, provides a stable and accurate method for obtaining a hierarchical basis from the candidate trunk network functions. From this point forward, Equation (3.7) is utilized, and the tilde is dropped from $\{\tilde{\phi}_k\}$ in all subsequent notation.



(a) (b)

Figure 3.3: The singular values (a) and absolute value of the expansion coefficients (b) corresponding to the advection equation. (b) shows the expansion coefficients obtained for the function $f(x) = e^{\sin(x)}$ when using Equation (3.5) (S and V matrices) and Equation (3.7) (U matrix).

The singular values $\{\sigma_k\}$ shown in Figure 3.3a provide insight into the contribution of each of the custom-made basis functions. Once the singular value corresponding to a given custom-made basis function falls below a certain threshold, the functions are largely noise and do not have a significant contribution to the system. For the PDE examples considered, a threshold of 10^{-13} was specified, and only the custom-made basis functions with corresponding singular values above this threshold were utilized. The elimination of the custom-made basis functions below this threshold leads to both computational savings and more robust solutions. Additionally, this can be seen as a form of model reduction but without the inclusion of any memory terms, such as those outlined in Section 2.1.

3.2.2 Approximation capabilities of custom-made basis functions

The decay shown in Figure 3.3b implies the custom-made orthonormal basis $\{\phi_k\}_{1\leq k\leq N}$, constructed using Equation (3.7) is capable of approximating smooth functions on the specified spatial domain Ω. The errors made in approximating the orthonormal Legendre polynomials can be utilized as a means to assess the approximation capabilities of the custom-made basis. Since the construction presented in Section 3.2.1 begins with the trained trunk net functions $\{\gamma_k\}_{1\leq k\leq N}$, this analysis accounts for the approximation errors in addition to the estimation and optimization errors. For this analysis, the domain of $\Omega = [0, 2\pi]$, is utilized.

The orthogonal projection of the function $f : [0, 2\pi] \mapsto \mathbb{C}$ can be defined as

$$Pf = \sum_{k=1}^{N} \langle \phi_k, f \rangle \phi_k, \tag{3.8}$$

and from the projection theorem,

$$\|f - Pf\| = \min_{g\in\mathcal{S}} \|f - g\|, \tag{3.9}$$

where

$$\|f\| = \sqrt{\int_{\Omega} [f(x)]^2 dx}. \tag{3.10}$$

Let $\{q_n\}_{n\geq 0}$ be the orthonormal Legendre polynomials on Ω and let

$$\mathcal{L}_R f = \sum_{n=0}^{R} \langle q_n, f \rangle q_n \tag{3.11}$$

denote the Legendre expansion of f. Using the fact $P\mathcal{L}_R f \in \mathcal{S}$ and Equation (3.9), it follows that

$$\|f - Pf\| \leq \|f - P\mathcal{L}_R f\|, \tag{3.12}$$

and from the triangle inequality,

$$\|f - Pf\| \leq \|f - \mathcal{L}_R f\| + \|\mathcal{L}_R f - P\mathcal{L}_R f\|. \tag{3.13}$$

Using Parseval's theorem yields

$$\|f - \mathcal{L}_R f\|^2 = \sum_{n\geq R+1} |\langle q_n, f \rangle|^2, \tag{3.14}$$

while Equation (3.11) yields

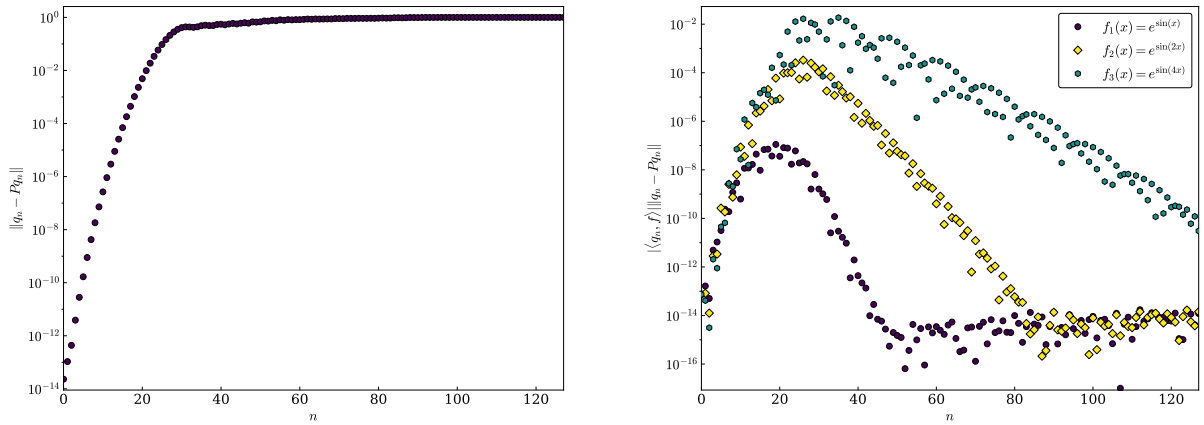
$$\|\mathcal{L}_R f - P\mathcal{L}_R f\| = \left\| \sum_{n=0}^{R} \langle q_n, f \rangle (q_n - Pq_n) \right\| \leq \sum_{n=0}^{R} |\langle q_n, f \rangle| \|q_n - Pq_n\|. \tag{3.15}$$

Plugging Equation (3.14) and Equation (3.15) into Equation (3.13),

$$\|f - Pf\| \leq \left(\sum_{n\geq R+1} |\langle q_n, f \rangle|^2 \right)^{1/2} + \sum_{n=0}^{R} |\langle q_n, f \rangle| \|q_n - Pq_n\|. \tag{3.16}$$

For smooth functions and a sufficiently large R, the second term on the right-hand side of Equation (3.16) will dictate the upper error bound, as the first term will decay rapidly due to the decay rate of the Legendre expansion coefficients for smooth functions. Therefore, the approximation error of a smooth function may be bounded in terms of the errors made in approximating the Legendre polynomials using the custom-made basis. Figure 3.4a

shows the errors made in approximating the Legendre polynomials using the custom-made basis functions, where for increasing n, the errors made in the approximation increase in an exponential fashion, indicating that the custom-made basis has difficulty in approximating the higher frequency functions—which correspond to larger n. If the entire second term on the right-hand side of Equation (3.16) is considered, the errors associated with the higher frequency Legendre polynomials are damped by the rapidly decreasing Legendre expansion coefficients. Figure 3.4b shows the dampening of the error for three functions with increasing frequencies, $f_1(x) = e^{\sin(x)}$, $f_2(x) = e^{\sin(2x)}$, and $f_3(x) = e^{\sin(4x)}$. For increasing frequency, $f_1(x) \to f_2(x) \to f_3(x)$, the dampening of the error associated with the higher frequency Legendre polynomials is diminished. For these results, a trunk network trained for the advection equation $\partial u/\partial t + \partial u/\partial x = 0$, on the periodic domain $x \in [0, 2\pi]$, and for $t \in [0, 1]$, was utilized to generate the custom-made basis. Fifty-three custom-made basis functions which satisfy the singular value threshold of 10^{-13}, as described in Section 3.2.1, were used for the analysis.



(a) $\|q_n - Pq_n\|$ (b) $|\langle q_n, f\rangle| \|q_n - Pq_n\|$

Figure 3.4: Errors in approximating the n-th Legendre polynomial using the custom-made basis functions (a). The upper error bound, $|\langle q_n, f\rangle| \|q_n - Pq_n\|$ computed for three different functions with increasing frequency (b).

3.3 Evolution of dynamical systems using custom-made basis functions

The general framework for solving time-dependent PDEs using the custom-made basis functions begins by writing the approximate solution as a linear combination of the basis functions $\{\phi_k\}$

$$u^r(t,x) = \sum_{k=1}^{r} a_k(t)\phi_k(x), \tag{3.17}$$

where r is the number of basis functions utilized from the orthonormal custom-made basis $\{\phi_k\}$. Next, consider a time-dependent PDE

$$\frac{\partial u}{\partial t} + \mathcal{N}[u] = 0, \quad t > 0,\ x \in [a,b], \tag{3.18}$$

where $\mathcal{N}$ is a potentially nonlinear differential operator with initial condition $u(0,x) = u_0(x)$, and with appropriately prescribed boundary conditions.

To obtain a system of ODEs in terms of the expansion coefficients, Equation (3.17) is inserted into Equation (3.18), and the Galerkin condition [9] is applied,

$$\left\langle \phi_m, \frac{\partial u^r}{\partial t} + \mathcal{N}[u^r] \right\rangle = 0, \quad \text{for } 1 \le m \le r, \tag{3.19}$$

which yields a system of ODEs

$$\frac{da_m(t)}{dt} = -\left\langle \phi_m, \mathcal{N}\left[\sum_{k=1}^{r} a_k(t)\phi_k\right] \right\rangle, \quad \text{for } 1 \le m \le r. \tag{3.20}$$

The expansion coefficients of the initial condition required to close the system of ODEs presented in Equation (3.20) are specified as

$$a_m(0) = \langle \phi_m, u_0(x) \rangle, \quad \text{for } 1 \le m \le r. \tag{3.21}$$

If the orthonormal basis functions utilized individually satisfy the boundary conditions of the PDE, Equation (3.20) can be used directly to solve for the time evolution of the

expansion coefficients. If, however, the basis functions utilized do not individually satisfy the boundary conditions of the PDE—as in the case of the presented custom-made basis functions—discontinuous Galerkin methods may be utilized, where Equation (3.20) is decomposed further by performing integration by parts and appropriately defining values to the resulting flux terms [65]. Refer to Appendix F for additional details. The tau-method [32] was also evaluated for the treatment of the boundary conditions but was found to be less robust than the discontinuous Galerkin approach.

3.4 Applications and numerical results

In this section, results are presented for six one-dimensional PDEs: advection, advection-diffusion, viscous Burgers, Korteweg–de Vries, Kuramoto–Sivashinsky, and inviscid Burgers. These six PDEs were chosen to investigate the effectiveness of the approach outlined in Section 3.2.1 and Section 3.3 due to the fact they exhibit a wide range of dynamics, are mathematical models describing physical phenomena, and are commonly chosen as test cases for the evaluation of numerical solvers. Additionally, the four nonlinear PDEs share a similar quadratic nonlinearity and are differentiated by different regularization mechanisms. Additional details about each of the selected example PDEs are provided in the subsections below. For all the presented examples, the domain was specified as $\Omega = [0, 2\pi]$, periodic boundary conditions were imposed, and the initial condition was denoted by $u_0(x)$.

For each example PDE, a DeepONet was trained to approximate the solution operator $\mathcal{G}$, that maps the initial condition $u_0(x)$, to the solution $u(t, x)$, for $t \in [0, 1]$. The ground truth data for the advection, advection-diffusion, viscous Burgers, Korteweg–de Vries, and Kuramoto–Sivashinsky equations was generated by writing the solution in terms of a 128- (advection, advection-diffusion, viscous Burgers) or a 512- (Korteweg–de Vries, Kuramoto–Sivashinsky) mode Fourier expansion, with the most negative mode set to zero and convolution sums dealiased using the 3/2 rule [11]. A Fourier expansion was utilized where applicable due to the fact that the Fourier basis functions are typically considered the gold standard and optimal choice for periodic domains. The ground truth for the inviscid Burgers equation

was generated by using a MUSCL (monotonic upwind scheme for conservation laws) scheme with a second-order Roe scheme for the flux, a minmod slope limiter [58, 83], and with the spatial domain discretized using 4096 points. Refer to Appendix E for additional details on generating the ground truth data, the DeepONet structure, and the DeepONet training parameters.

The custom-made bases were constructed as outlined in Section 3.2.1 for all PDEs. The r number of basis functions used to solve the PDE was specified to be the set of functions with a corresponding singular value larger than 10^{-13}. All the examples presented were solved on a 128-node Gauss–Legendre quadrature grid, and Gauss–Legendre quadrature was utilized for the calculation of inner products. Differentiation of the custom-made basis functions was performed using automatic differentiation. The quadratic nonlinear terms were computed in modal space, with all necessary triple product integrals computed in advance. Refer to Appendix K for preliminary testing of a pseudo-spectral transform based on the custom-made basis functions. With the exception of the inviscid Burgers equation, the same adaptive step size numerical integrator and tolerances used to generate the ground truth data for training the DeepONets were utilized to evolve the custom-made basis function solutions when using the discontinuous Galerkin approach (refer to Appendix E for additional details). For the inviscid Burgers equation evolved using the custom-made basis functions, a Runge–Kutta–Dormand–Prince integrator with adaptive step size, relative error tolerance 10^{-10}, and absolute tolerance 10^{-14} [63] was utilized. The solutions were saved at time values of 10^{-3} apart for the linear PDEs and 10^{-4} apart for the nonlinear PDEs.

For each example PDE, with the exception of the inviscid Burgers equation, an extended temporal interval of $t \in [0, 10]$ is presented, which includes times beyond the temporal training interval of $t \in [0, 1]$ for evaluation of the temporal extrapolation capabilities of the presented framework. Additionally, three initial conditions were considered, one initial condition which was taken from within the training distribution (taken from DeepONet testing data) and two that fall outside the training distribution, $u_0(x) = \sin(x)$, and $u_0(x) = e^{\sin(x)}$. Two initial conditions taken from outside the training distribution were considered

so that the extrapolation capabilities in terms of input function space could be evaluated. Refer to Appendix G for plots of the in-distribution initial condition used for each PDE and the two out-of-distribution initial conditions. For the advection, advection-diffusion, viscous Burgers, Korteweg–de Vries, and Kuramoto-Sivashinsky equations, two different parameter values were also considered for each PDE; the value for which the DeepONet was trained and an alternative value. The alternative parameter values provide a third means of testing the extrapolation capabilities of the presented framework.

The error presented for each PDE is a relative Euclidean l^2 error defined by

$$E(t) = \frac{\|u^r(t,\cdot) - u_{\mathrm{G}}(t,\cdot)\|}{\|u_{\mathrm{G}}(t,\cdot)\|}, \tag{3.22}$$

where u_{G} is the ground truth solution, and u^r is the custom-made basis function solution using r custom-made basis functions. The Fourier expansion coefficients are used to approximate the Fourier solution at the non-uniform quadrature nodes. For the inviscid Burgers equation and the MUSCL solution, the solution computed on the 4096-point spatial grid is interpolated using piecewise linear interpolation to approximate the solution at the non-uniform quadrature nodes. The average relative error over $[0, T]$ is defined by

$$\overline{E} = \frac{1}{T}\int_0^T E(t)dt. \tag{3.23}$$

All errors presented are based on one random DeepONet training and do not necessarily signify the best or worst case. To indicate the effects of random initialization of the deep neural networks, the mean testing error and corresponding standard deviation based on three training runs are presented in Table 3.1 for each example PDE.

Figure 3.5 shows the decaying singular value spectrum for all six of the test PDEs and highlights the hierarchical structure of the custom-made basis functions corresponding to each PDE. Additionally, the variation in the decay rates between each of the spectrums indicate that the trunk net space of functions is linked to the complexity of the dynamics

PDE	Mean	Standard deviation
Advection	3.37×10^{-5}	3.59×10^{-6}
Advection-diffusion	6.90×10^{-5}	2.69×10^{-5}
Viscous Burgers	2.36×10^{-3}	1.79×10^{-4}
Korteweg–de Vries	3.05×10^{-2}	2.95×10^{-5}
Kuramoto–Sivashinsky	6.58×10^{-2}	4.24×10^{-4}
Inviscid Burgers	1.23×10^{-2}	2.44×10^{-4}

Table 3.1: DeepONet mean testing errors and standard deviation for all example PDEs based on three training runs each.

that each of the PDEs exhibit. This linking between the decay rates and the complexity of the corresponding PDE supports the claim that these are indeed custom-made basis functions for the PDE of interest.

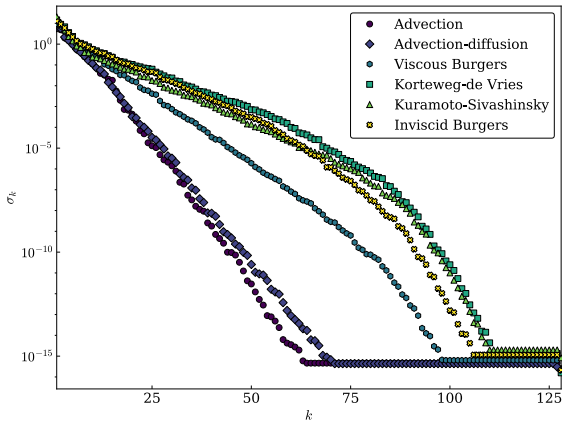


Figure 3.5: Singular value spectrum of the custom-made basis functions for the advection, advection-diffusion, viscous Burgers, Korteweg–de Vries, Kuramoto–Sivashinsky, and inviscid Burgers equations.

3.4.1 Application 1: 1D Advection equation

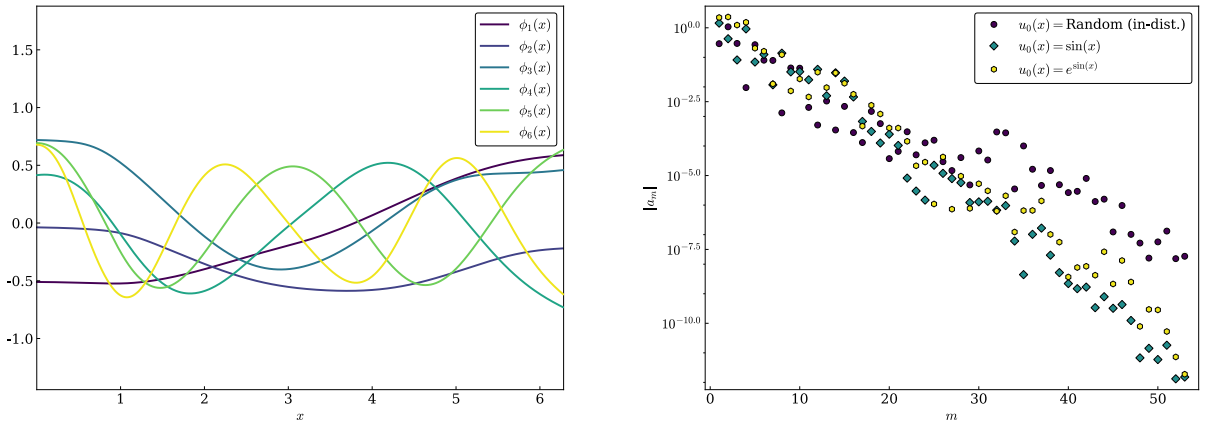
Consider the one-dimensional linear advection equation,

$$\frac{\partial u}{\partial t} + \alpha \frac{\partial u}{\partial x} = 0, \tag{3.24}$$

with the parameter $\alpha = 1.0$. The advection equation models the quantity u, as it is advected with velocity α. Alternatively, it is commonly referred to as the transport equation as it is used to model the transport of a substance in a uniform field with velocity α [53].

The custom-made basis functions for the advection equation were found to be generally ordered in terms of increasing oscillatory behavior as shown in Figure 3.6a while the decreasing expansion coefficients of the initial conditions shown in Figure 3.6b further highlights the hierarchical structure of the custom-made basis functions. The ordering of the custom-basis functions in terms of increasing oscillatory behavior and the rapid decay of the expansion coefficients of the initial conditions are two properties that the custom-made basis functions share with basis functions commonly employed in spectral methods, e.g., Fourier series, where Fourier basis functions are ordered in terms of increasing frequency and the corresponding coefficients decay rapidly for smooth functions. The hierarchical structure allows for smooth functions to be represented to an arbitrarily high accuracy by taking the truncation value r, for the custom-made basis functions to be sufficiently large. The ability to truncate at a finite number of basis functions while maintaining an arbitrarily high accuracy facilitates the usage of spectral methods.

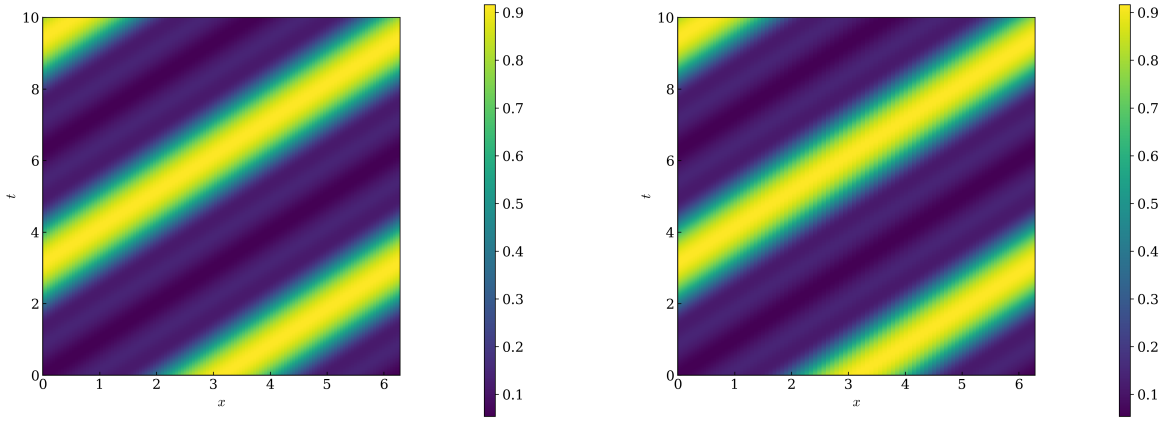
Figure 3.7 presents results for the spatiotemporal evolution using the 128-mode Fourier expansion and the $r = 53$ custom-made basis functions for the in-distribution initial condition. The spatiotemporal plots illustrate that the presented procedure appropriately captures the traveling wave and the periodicity of the solution as it moves in and out of the domain. For all three initial conditions, good agreement is obtained between the Fourier solution and the custom-made basis solution—even for the initial conditions outside the training domain—as evidenced by the evolution of the relative errors shown in Figure 3.8. Spatiotemporal plots



(a) Custom-made basis functions (b) Initial condition expansion coefficients $|a_m(0)|$

Figure 3.6: Custom-made basis functions (a) and corresponding initial condition expansion coefficients (b) for the advection equation when using $r = 53$ custom-made basis functions.

for the two out-of-distribution initial conditions can be found in Appendix G.1 (Figures G.3 and G.4).



(a) Fourier solution (b) Custom-made basis solution

Figure 3.7: Spatiotemporal evolution of the advection equation for the temporal interval $t \in [0, 10]$ and the random in-distribution initial condition.

The alternative parameter $\alpha = -4.0$, for which the DeepONet was not trained, was also

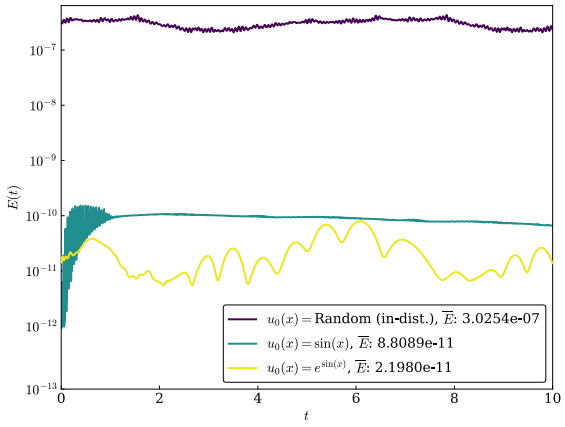


Figure 3.8: Relative error evolution of the advection equation for the temporal interval $t \in [0, 10]$ and the three test initial conditions.

tested. The spatiotemporal evolution for $\alpha = -4.0$ and the in-distribution initial condition using the 128-mode Fourier expansion and the $r = 53$ custom-basis functions are shown in Figure 3.9. The evolution of the relative errors for the alternative parameter value and for the three initial conditions shown in Figure 3.10 highlight the ability of the outlined framework to also extrapolate to parameter spaces outside the training domain. Spatiotemporal plots for the two out-of-distribution initial conditions can be found in Appendix G.1 (Figure G.5 and G.6).

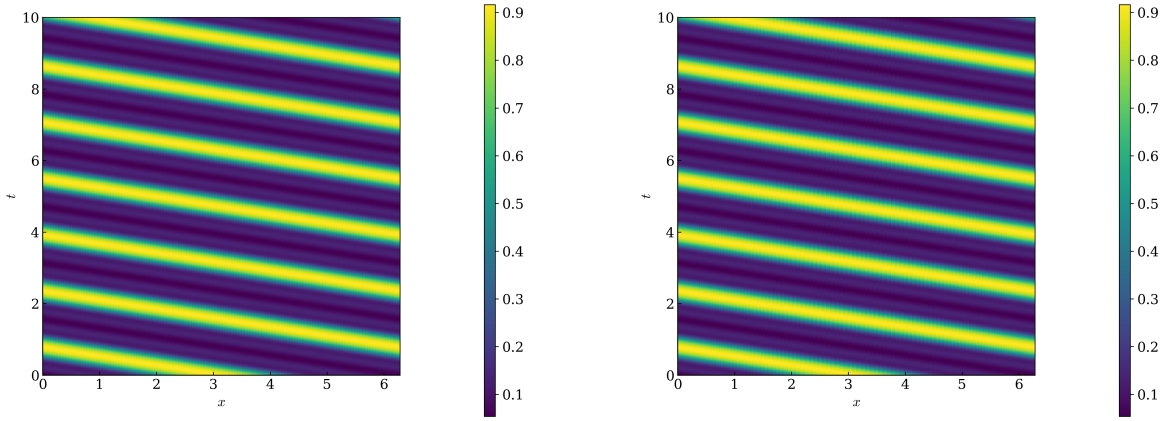
3.4.2 Application 2: 1D Advection-diffusion equation

Next, consider the one-dimensional linear advection-diffusion equation,

$$\frac{\partial u}{\partial t} + \alpha \frac{\partial u}{\partial x} - \nu \frac{\partial^2 u}{\partial x^2} = 0, \tag{3.25}$$

with the parameters $\alpha = 1.0$ and $\nu = 0.1$. The advection-diffusion equation is similar to the advection equation in that it models the quantity u, as it is advected with velocity α, but also contains an additional second-order diffusive term with corresponding viscosity ν.

Figure 3.11 presents the results for the spatiotemporal evolution using the 128-mode



(a) Fourier solution

(b) Custom-made basis solution

Figure 3.9: Spatiotemporal evolution of the advection equation for the temporal interval $t \in [0, 10]$, the random in-distribution initial condition, and with $\alpha = -4.0$.

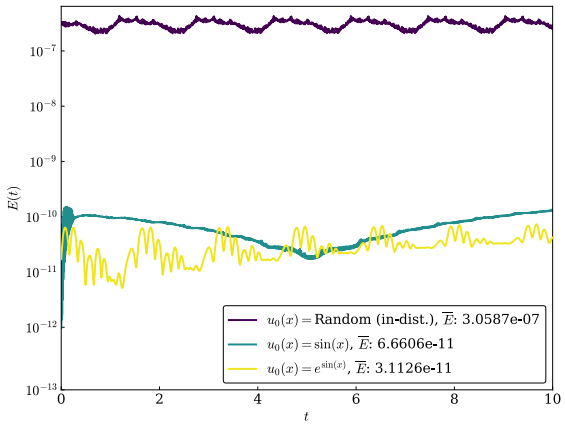
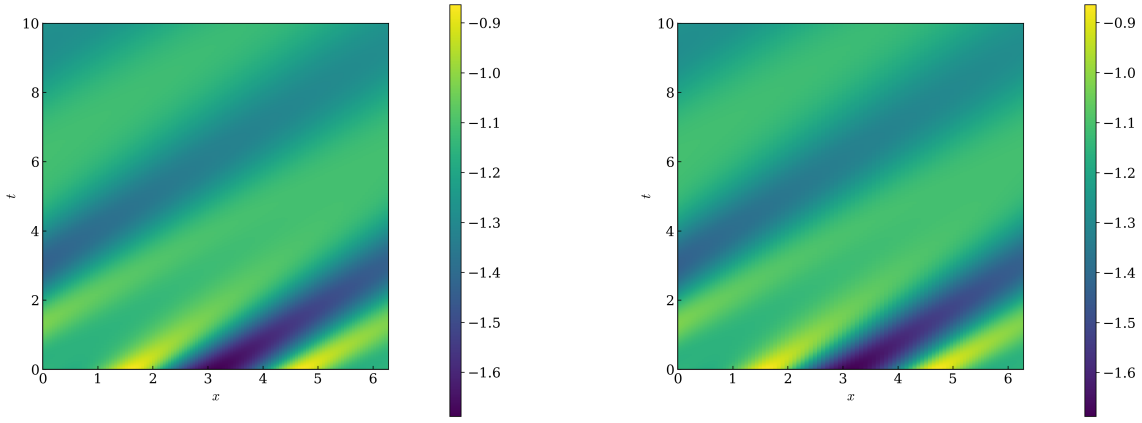


Figure 3.10: Relative error evolution of the advection equation for the temporal interval $t \in [0, 10]$, the three test initial conditions, and with $\alpha = -4.0$.

Fourier expansion and the $r = 59$ custom-made basis functions for the random in-distribution initial condition. The spectral solution, which utilizes the custom-made basis functions, not only appropriately captures the traveling wave and the periodicity of the solution as it moves in and out of the domain but also accurately captures the diffusive nature of Equation (3.25). Figure 3.12 presents the evolution of the relative errors for the three initial conditions and

shows strong agreement is again obtained for all three of the presented initial conditions. Plots of the custom-made basis functions (Figure G.7a), the expansion coefficients (Figure G.7b), and the two out-of-distribution initial conditions (Figures G.9 and G.10) can be found in Appendix G.2.



(a) Fourier solution

(b) Custom-made basis solution

Figure 3.11: Spatiotemporal evolution of the advection-diffusion equation for the temporal interval $t \in [0, 10]$ and the random in-distribution initial condition.

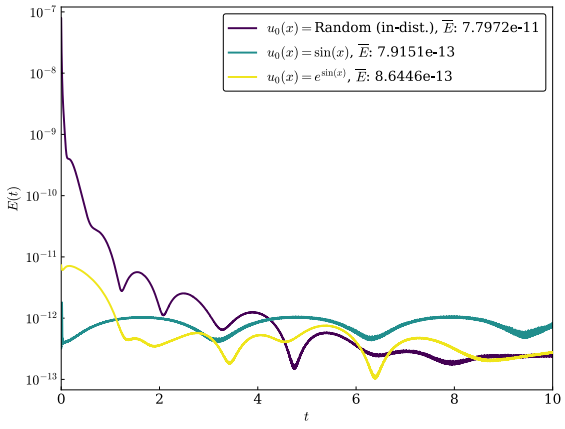
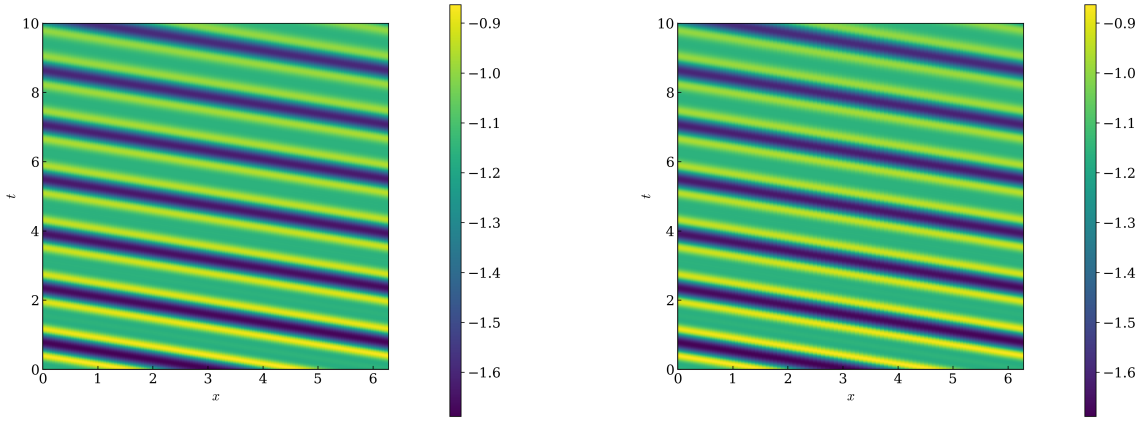


Figure 3.12: Relative error evolution of the advection-diffusion equation for the temporal interval $t \in [0, 10]$ and the three test initial conditions.

The alternative parameters $\alpha = -4.0$, and $\nu = 0.01$, for which the DeepONet was not trained, were also tested. The spatiotemporal evolution for $\alpha = -4.0$, $\nu = 0.01$, and the in-distribution initial condition using the 128-mode Fourier expansion and the $r = 59$ custom-basis functions are shown in Figure 3.13. The low relative errors presented in Figure 3.14 for the alternative parameter values $\alpha = -4.0$ and $\nu = 0.01$—which represents a more complex parameter space than that presented for the advection equation—further highlight the ability to extrapolate to parameter spaces outside the training domain. Spatiotemporal plots for the two out-of-distribution initial conditions can be found in Appendix G.2 (Figure G.11 and G.12).



(a) Fourier solution

(b) Custom-made basis solution

Figure 3.13: Spatiotemporal evolution of the advection-diffusion equation for the temporal interval $t \in [0, 10]$, the random in-distribution initial condition, and with $\alpha = -4.0$, $\nu = 0.01$.

3.4.3 Application 3: 1D Viscous Burgers equation

Next, consider the one-dimensional nonlinear viscous Burgers equation,

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = 0, \tag{3.26}$$

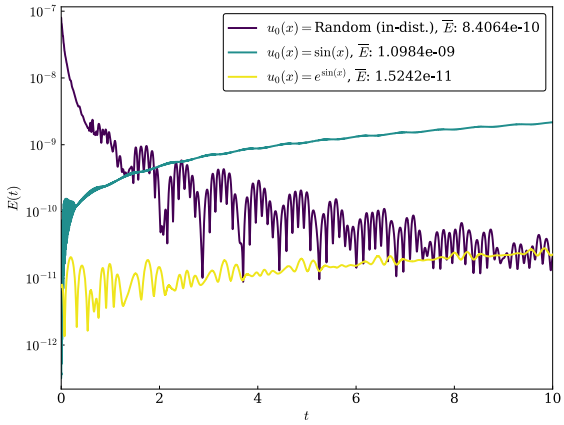
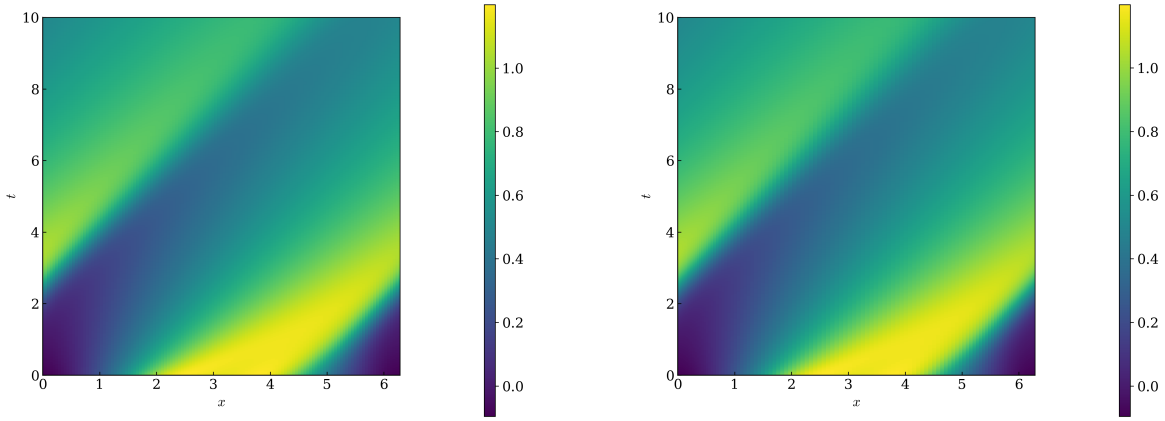


Figure 3.14: Relative error evolution of the advection-diffusion equation for the temporal interval $t \in [0, 10]$, the three test initial conditions, and with $\alpha = -4.0$, $\nu = 0.01$.

with the parameter $\nu = 0.1$. The viscous Burgers equation combines nonlinear wave propagation with a second-order diffusion term and is a very simplified version of the equations for viscous fluid flow [53]. In addition to being a simplified model for viscous fluid flow, the viscous Burgers equation also has applications in cosmology and is commonly used as a test case for numerical solvers [4,7]. For $\nu > 0$, the diffusive term has the effect of smoothing out any shock discontinuities.

The results for the spatiotemporal evolution using the 128-mode Fourier expansion and the $r = 91$ custom-made basis functions are shown in Figure 3.15 and illustrate that the presented procedure accurately captures the traveling, smoothed-out shock that develops when initialized from the random in-distribution initial condition. The evolution of the relative errors for the three initial conditions are presented in Figure 3.16 and demonstrate that the framework can be extended to nonlinear problems while maintaining the ability to extrapolate both temporally and in terms of the input function space. Plots of the custom-made basis functions (Figure G.13a) and the expansion coefficients (Figure G.13b) can be found in Appendix G.3. Spatiotemporal plots for the two out-of-distribution initial conditions can also be found in Appendix G.3 (Figures G.15 and G.16). The spatiotemporal plots presented for the initial condition $u_0(x) = \sin(x)$ (Figure G.15), demonstrate the ability

of the framework to capture both traveling and standing, smoothed-out shocks.



(a) Fourier solution

(b) Custom-made basis solution

Figure 3.15: Spatiotemporal evolution of the viscous Burgers equation for the temporal interval $t \in [0, 10]$ and the random in-distribution initial condition.

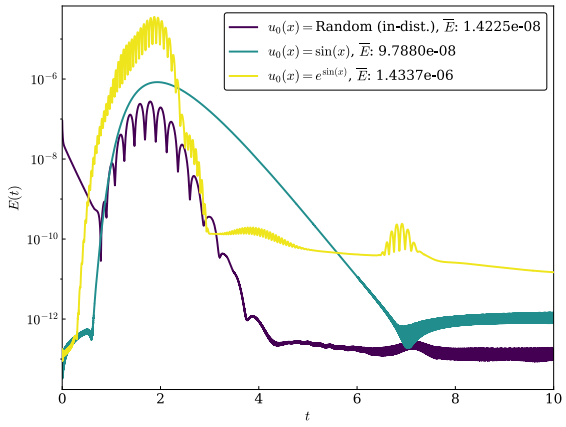
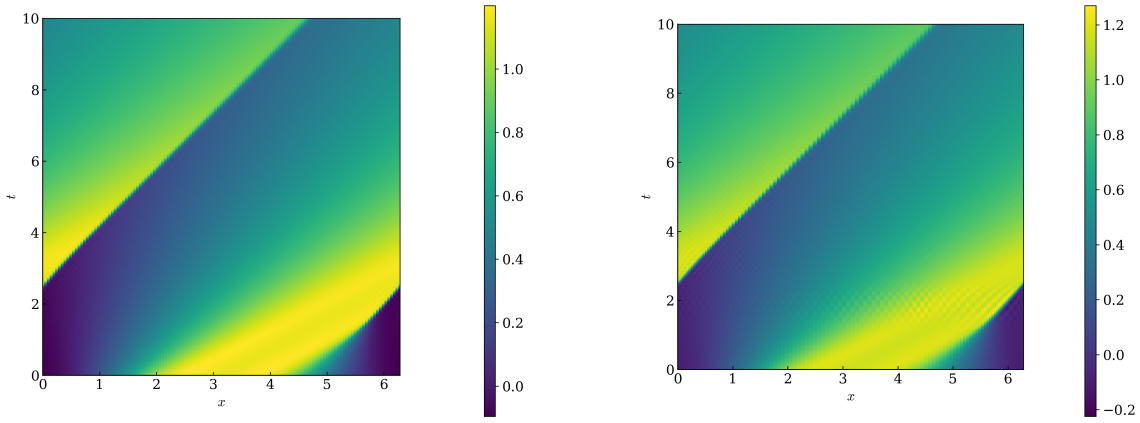


Figure 3.16: Relative error evolution of the viscous Burgers equation for the temporal interval $t \in [0, 10]$ and the three test initial conditions.

The alternative parameter $\nu = 0.01$, for which the DeepONet corresponding to the viscous Burgers equation was not trained, was also tested. Recall, in the limit as $\nu \to 0$, the singular, inviscid Burgers equation is recovered. The spatiotemporal evolution for $\nu = 0.01$,

and the in-distribution initial condition using the 128-mode Fourier expansion and the $r = 91$ custom-basis functions are shown in Figure 3.17, while the evolution of the relative errors for the alternative parameter value are shown in Figure 3.18. The results for the alternative parameter value $\nu = 0.01$ qualitatively capture the solution characteristics but highlight the difficulties associated with lowering the viscosity ν, which leads to the formation of a steeper wavefront and results in greater average relative errors than those found for the more smoothed-out shocks corresponding to $\nu = 0.1$. Despite the decreased accuracy when compared to the larger viscosity value for which the custom-made basis functions were developed, the performance is comparable to physics-informed DeepONets for $\nu = 0.01$ [76] without the need to retrain the underlying DeepONet. Spatiotemporal plots for the two out-of-distribution initial conditions can be found in Appendix G.3 (Figure G.17 and G.18).



(a) Fourier solution

(b) Custom-made basis solution

Figure 3.17: Spatiotemporal evolution of the viscous Burgers equation for the temporal interval $t \in [0, 10]$, the random in-distribution initial condition, and with $\nu = 0.01$.

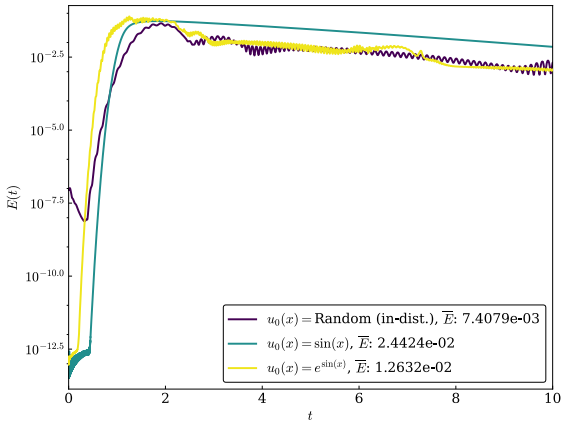


Figure 3.18: Relative error evolution of the viscous Burgers equation for the temporal interval $t \in [0, 10]$, the three test initial conditions, and with $\nu = 0.01$.

3.4.4 Application 4: 1D Korteweg–de Vries equation

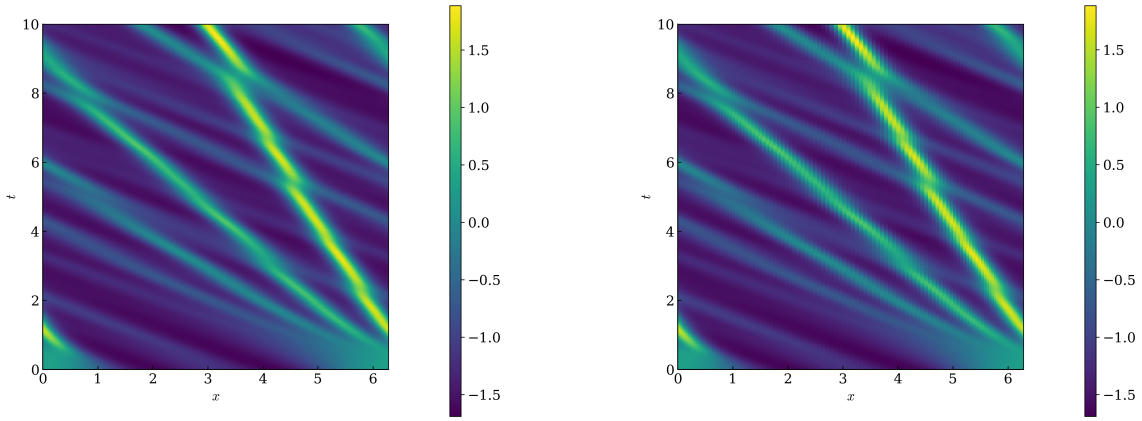
Next, consider the one-dimensional nonlinear Korteweg–de Vries equation,

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \delta^2 \frac{\partial^3 u}{\partial x^3} = 0, \tag{3.27}$$

with parameter $\delta = 0.1$. The Korteweg–de Vries equation combines nonlinear wave propagation with a third-order dispersion term and allows for localized traveling wave solutions known as "solitons" [53]. When two solitons collide, there is a relative phase shift, but their individual profiles emerge unchanged. The Korteweg–de Vries equation is commonly used as a model for shallow-water waves, collisionless-plasma magnetohydrodynamic waves, and long waves in anharmonic crystals [84]. As the strength of dispersion δ, trends towards zero, the oscillations present in the solution become more closely spaced [51].

The results for the spatiotemporal evolution using the 512-mode Fourier expansion and the $r = 106$ custom-made basis functions are shown in Figure 3.19. The spatiotemporal plots illustrate that the presented procedure accurately captures the nonlinearly interacting solitons (light-colored streaks) and the periodicity of the solution when initialized from the random in-distribution initial condition. The evolution of the relative errors for the three ini-

tial conditions are shown in Figure 3.20, which highlights how the framework can be extended to nonlinear problems with a similar quadratic nonlinearity, but very different regularization mechanisms, while still maintaining the ability to extrapolate both temporally and in terms of the input function space. Plots of the custom-made basis functions (Figure G.19a) and the expansion coefficients (Figure G.19b) can be found in Appendix G.4. Spatiotemporal plots for the two out-of-distribution initial conditions can also be found in Appendix G.4 (Figures G.21 and G.22).



(a) Fourier solution

(b) Custom-made basis solution

Figure 3.19: Spatiotemporal evolution of the Korteweg–de Vries equation for the temporal interval $t \in [0, 10]$ and the random in-distribution initial condition.

The alternative parameter $\delta = \sqrt{0.005}$, for which the DeepONet was not trained, was also tested. The spatiotemporal evolution for $\delta = \sqrt{0.005}$, and the in-distribution initial condition using the 512-mode Fourier expansion and the $r = 106$ custom-basis functions are shown in Figure 3.21. Similar to the viscous Burgers equation, qualitatively, the solution characteristics are well captured, but a decrease in accuracy is found for the alternative parameter value for all three initial conditions when utilizing the same number of custom-made basis functions and the same 128-node Gauss–Legendre quadrature grid. Spatiotemporal plots for the two out-of-distribution initial conditions can be found in Appendix G.4 (Figure

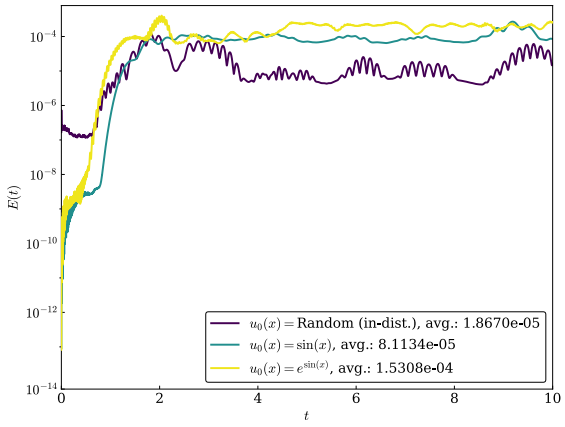
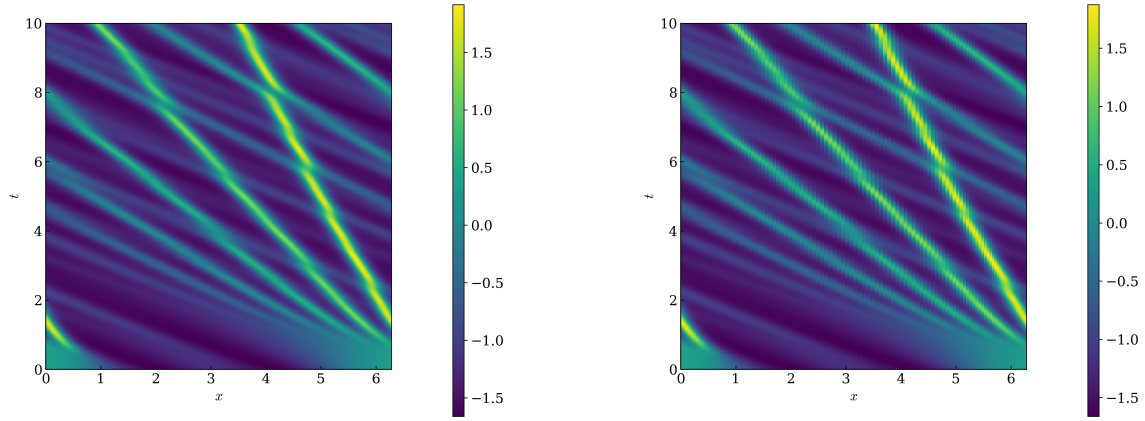


Figure 3.20: Relative error evolution of the Korteweg–de Vries equation for the temporal interval $t \in [0, 10]$ and the three test initial conditions.

G.23 and G.24).



(a) Fourier solution

(b) Custom-made basis solution

Figure 3.21: Spatiotemporal evolution of the Korteweg–de Vries equation for the temporal interval $t \in [0, 10]$, the random in-distribution initial condition, and with $\delta = \sqrt{0.005}$.

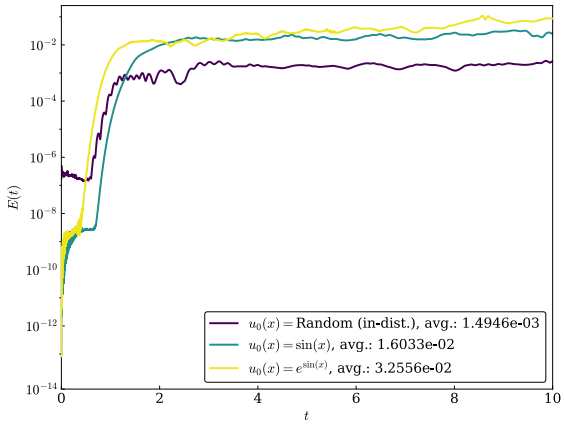


Figure 3.22: Relative error evolution of the Korteweg–de Vries equation for the temporal interval $t \in [0, 10]$, the three test initial conditions, and with $\delta = \sqrt{0.005}$.

3.4.5 Application 5: 1D Kuramoto–Sivashinsky equation

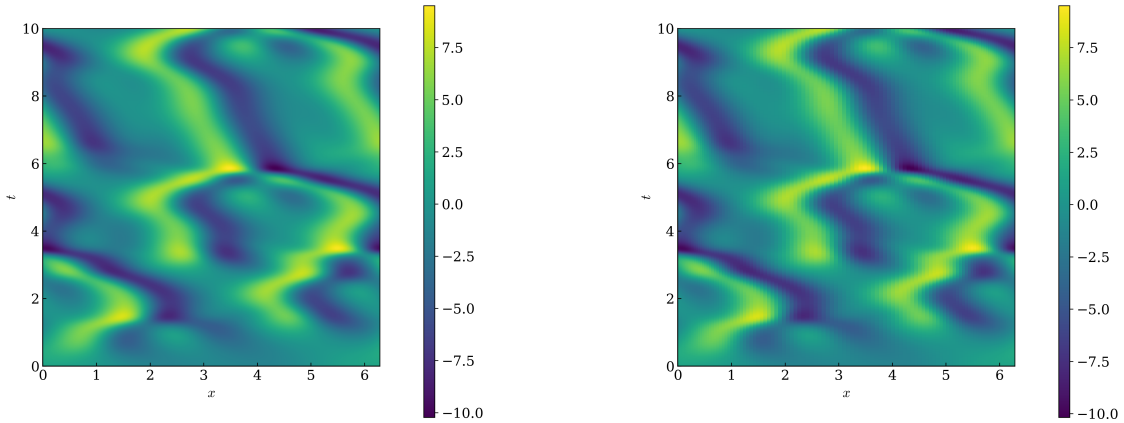
Next, consider the one-dimensional nonlinear Kuramoto–Sivashinsky equation,

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{\partial^2 u}{\partial x^2} + \beta\frac{\partial^4 u}{\partial x^4} = 0, \tag{3.28}$$

with parameter $\beta = 0.085$. The Kuramoto–Sivashinsky equation combines the quadratic nonlinearity present in the other nonlinear PDEs with a destabilizing second-order diffusion term and a stabilizing fourth-order dissipation term. The Kuramoto–Sivashinsky equation can exhibit chaotic behavior and is commonly used as a model for flame propagation and reaction-diffusion dynamics [28, 55]. The parameter value $\beta = 0.085$ was chosen due to the fact that it belongs in the first window that supports chaotic solutions [35, 36].

The results for the spatiotemporal evolution using the 512-mode Fourier expansion and the $r = 105$ custom-made basis functions are shown in Figure 3.23 and illustrate that the presented framework accurately captures the complex dynamics as well as the periodicity of the solution when initialized from the random in-distribution initial condition. The evolution of the relative errors for the three initial conditions are shown in Figure 3.24 and highlight how the framework can be applied to nonlinear problems with a quadratic nonlinearity

and with multiple different regularization mechanisms, while still maintaining the ability to extrapolate both temporally and in terms of the input function space. Plots of the custom-made basis functions (Figure G.25a) and the expansion coefficients (Figure G.25b) can be found in Appendix G.5. Spatiotemporal plots for the two out-of-distribution initial conditions can also be found in Appendix G.5 (Figures G.27 and G.28).



(a) Fourier solution

(b) Custom-made basis solution

Figure 3.23: Spatiotemporal evolution of the Kuramoto–Sivashinsky equation for the temporal interval $t \in [0, 10]$ and the random in-distribution initial condition.

The alternative parameter $\beta = 0.05$, for which the DeepONet was not trained, was also tested. The value $\beta = 0.05$ was chosen due to the fact that it belongs in the second window that supports chaotic solutions [35, 36]. The spatiotemporal evolution for $\beta = 0.05$, and the in-distribution initial condition using the 512-mode Fourier expansion and the $r = 105$ custom-basis functions are shown in Figure 3.25, while the evolution of the relative error for the alternative parameter value and for the three initial conditions are shown in Figure 3.26. The solutions to the Kuramoto–Sivashinsky equation are dependent on the value of β, and depending on the selected value, the long time-behavior of the solution can vary from being steady-state to complicated, and time-oscillatory [55]. Therefore, the results presented in Figure 3.26 provide further strong evidence of the ability of the presented framework to

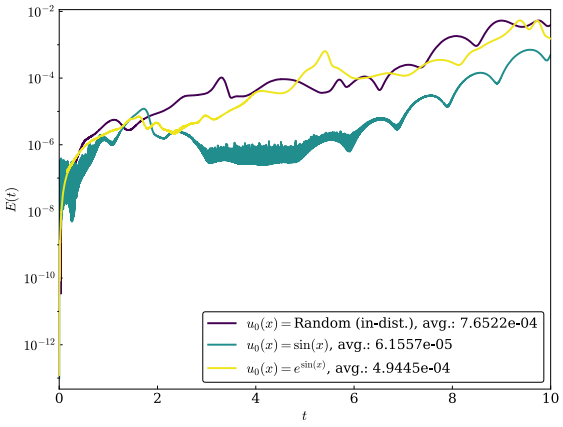
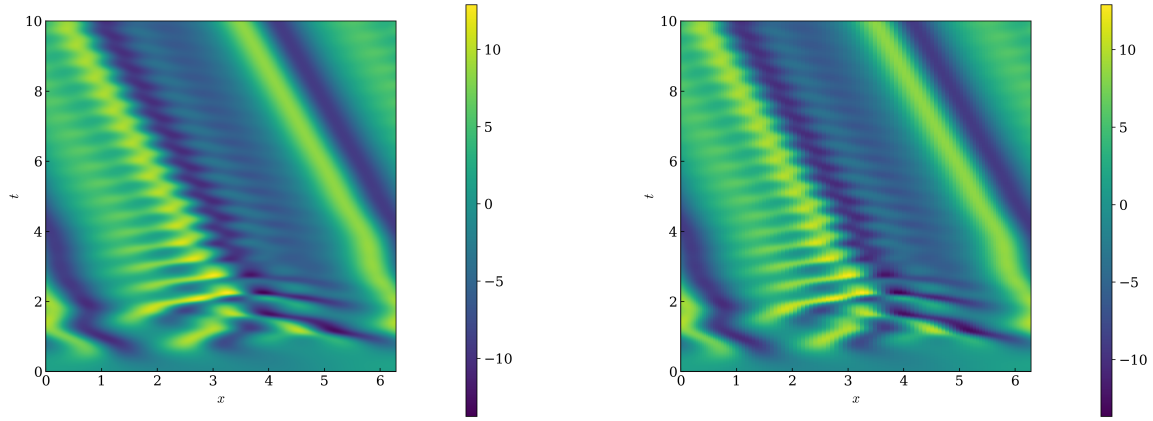


Figure 3.24: Relative error evolution of the Kuramoto–Sivashinsky equation for the temporal interval $t \in [0, 10]$ and the three test initial conditions.

extrapolate to parameter spaces outside the training domain. Spatiotemporal plots for the two out-of-distribution initial conditions can be found in Appendix G.5 (Figure G.29 and G.30).



(a) Fourier solution

(b) Custom-made basis solution

Figure 3.25: Spatiotemporal evolution of the Kuramoto–Sivashinsky equation for the temporal interval $t \in [0, 10]$, the random in-distribution initial condition, and with $\beta = 0.05$.

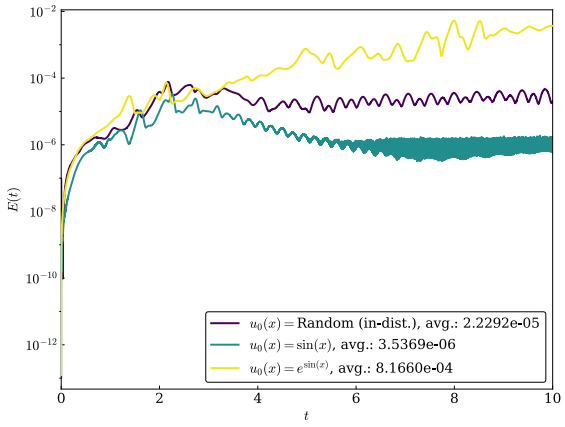


Figure 3.26: Relative error evolution of the Kuramoto–Sivashinsky equation for the temporal interval $t \in [0, 10]$, the three test initial conditions, and with $\beta = 0.05$.

3.4.6 Application 6: 1D Inviscid Burgers equation

As a final example, consider the one-dimensional inviscid Burgers equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0. \tag{3.29}$$

As discussed in Section 2.4.1, the inviscid Burgers equation is often viewed as a simplified analog of the Euler equations [58]. Equation (3.29) contains the same quadratic nonlinearity as the other nonlinear models, but unlike the other models considered, is devoid of any regularization. Due to the lack of any regularization, solutions to the inviscid Burgers equation are capable of developing discontinuities in the velocity field, i.e., shocks, in finite time when initialized from smooth initial conditions [4].

The results for the spatiotemporal evolution using the MUSCL scheme with 4096 spatial discretization points and the $r = 101$ custom-made basis functions are shown in Figure 3.27 for the in-distribution initial condition and for $t \in [0, 0.8296]$—which is shorter than the corresponding DeepONet training temporal domain of $t \in [0, 1]$. The final solution times presented are specified as the time when the energy in a given time step exceeds 100.1% of the energy in the previous time step or when the absolute value of the difference between

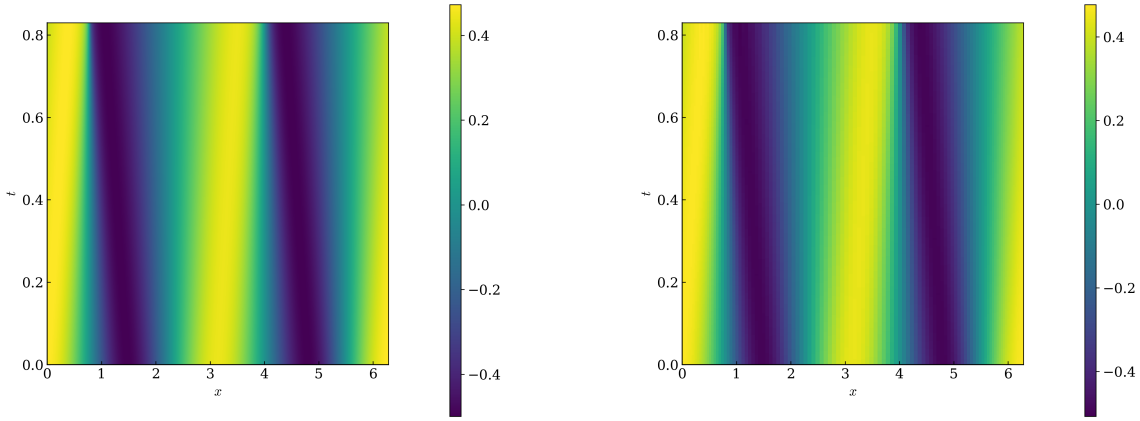
$u(t, 2\pi)$ and $u(t, 0)$ exceeds 10^{-2}. The increasing inaccuracy of the results occurs due to the fact that the solutions form shocks in finite time, and the presented spectral method based on the custom-made basis functions does not contain a mechanism to accurately eject the energy that is being consumed by the shock. Figure 3.28 shows that for times in advance of the shock, strong agreement is obtained between the MUSCL solution and the custom-made basis function solution, but around the time the shock develops, the solution becomes increasingly inaccurate, and eventually, the specified solution exit criteria are reached.

This increasing level of inaccuracy should not be viewed as a shortcoming of the presented framework; rather, this is an issue commonly encountered when using spectral methods for the evolution of singular PDEs. Similar to the discussion provided in Section 2.4.1, the ability of the inviscid Burgers equation to develop shocks motivated the usage of a MUSCL solution to generate the ground truth data for training the DeepONet, as a Fourier expansion would also not converge to the true solution [5]. To account for the ejection of energy and to accurately capture the temporal evolution of the energy, the system needs to be augmented by a memory term, as was presented for spectral methods based on a Fourier expansion in Chapter 2. Plots of the custom-made basis functions (Figure G.31a) and the expansion coefficients (Figure G.31b) can be found in Appendix G.3. Spatiotemporal plots for the two out-of-distribution initial conditions can also be found in Appendix G.3 (Figures G.33 and G.34).

3.5 Summary

Section 3.2 presented a general framework for using DeepONets to identify candidate functions that can be transformed into a hierarchical orthonormal basis, while Section 3.3 outlined the procedure for using these custom-made basis functions in a spectral method to solve PDEs. Section 3.4 illustrated the effectiveness of the presented scientific machine learning approach for solving both linear and nonlinear one-dimensional PDEs that serve as mathematical models for a range of physical phenomena on periodic domains.

The specified initial conditions and temporal domain of $t \in [0, 10]$ for the results presented



(a) MUSCL solution

(b) Custom-made basis solution

Figure 3.27: Spatiotemporal evolution of the inviscid Burgers equation for the temporal interval $t \in [0, 0.8296]$ and the random in-distribution initial condition.

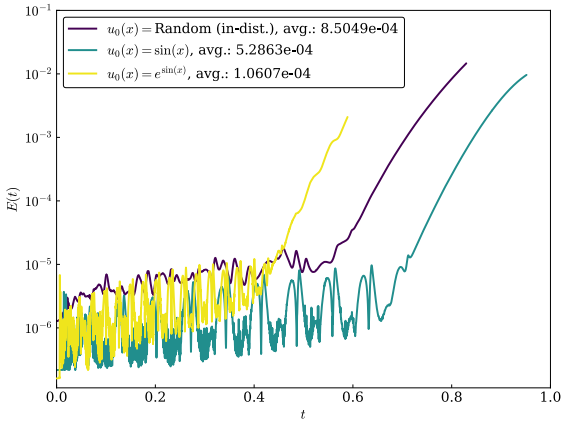


Figure 3.28: Relative error evolution of the inviscid Burgers equation for the temporal interval $t \in [0, 0.8296]$ for the in-distribution initial condition, $t \in [0, 0.9515]$ for $u_0(x) = \sin(x)$, and $t \in [0, 0.5892]$ for $u_0(x) = e^{\sin(x)}$.

in Section 3.4 illustrate the interpolation and extrapolation capabilities of the framework as evidenced by Figures 3.8, 3.12, 3.16, 3.20, and 3.24, which show strong agreement with the Fourier solution for the in-distribution initial condition as well as the two out-of-distribution initial conditions over the entire temporal domain. The framework applied to the inviscid Burgers equation also shows strong agreement with the MUSCL solution for all three test

initial conditions in advance of the shock (Figure 3.28). To allow for further temporal evolution, the addition of a memory term is required (refer to Chapter 2) to model the singular inviscid Burgers equation and is a common requirement of spectral methods, and not specific to the presented framework.

The results presented for the advection, advection-diffusion, viscous Burgers, Korteweg–de Vries, and Kuramoto–Sivashinsky equations in Figures 3.10, 3.14, 3.18, 3.22, and 3.26 show agreement with the Fourier solutions but at a reduced level of accuracy for several of the example PDEs for parameters for which the DeepONet utilized to identify the candidate basis functions was not trained. In each of these cases, the modification of the PDE parameters results in different solution characteristics than those present in the DeepONet training data, such as steeper wavefronts and more closely spaced oscillations in the case of the viscous Burgers and Korteweg–de Vries equations, respectively.

This framework provides many opportunities for extension even when only considering one-dimensional PDEs on periodic domains. Preliminary work on two potential extensions included as part of this work are time-sampling the custom-made basis functions and constructing periodic custom-made basis functions through the use of a feature expansion [47].

1. **Time-sampling the custom-made basis functions.** For all the work presented in this chapter, the candidate functions obtained from DeepONets were evaluated at $t = 0$. This evaluation time was only an initial assumption for the appropriate time for evaluating the trunk network functions, and there is nothing that limits the evaluation time to be $t = 0$. In Appendix H, the concept of time-sampling the custom-made basis functions is investigated, where the candidate trunk network functions $\{\gamma_k\}_{1\leq k\leq N}$, are sampled at equally spaced times with a sampling rate of $\Delta t = 0.05$. Following the time-sampling procedure, more custom-made basis functions are identified that exceed the singular value threshold of 10^{-13} than are identified when using the frozen-in-time approach and the approximation errors for the same three functions presented in Section 3.2.2 are reduced when utilizing the time-sampling approach along with the same

128-node Gauss–Legendre quadrature grid. Furthermore, the average relative errors over the temporal interval $t \in [0, 10]$, are decreased for all three test initial conditions for the advection-diffusion and Korteweg–de Vries equations. Refer to Appendix H, specifically Figure H.4 and Tables H.2, H.3, H.4, H.5, and H.6 for more detailed results.

2. **Development of periodic custom-made basis functions.** When using the standard DeepONet architecture, the resulting custom-made basis functions do not individually satisfy the boundary conditions and require the use of discontinuous Galerkin methods to enforce the boundary conditions when evolving PDEs. Through the use of a feature expansion combined with DeepONets for periodic problems, the trunk network functions can be forced to individually satisfy the boundary conditions. The concept of using a feature expansion based on two Fourier basis functions ($x \mapsto \{\sin(x), \cos(x)\}$), applied to the spatial component of the trunk network input is investigated in Appendix I. A similar approach, as presented in Section 3.2.1, is utilized, but with a 128-node equally spaced grid specified and B-Splines used to evaluate the custom-made basis at locations away from the quadrature nodes. For all example PDEs, a greater number of periodic custom-made basis functions that satisfy the singular value threshold of 10^{-13} are identified than when using the procedure outlined in Section 3.2.1. Additionally, for the advection and viscous Burgers equations, there is an improvement in the relative average error for all three test initial conditions for the temporal domain $t \in [0, 10]$. The usage of periodic custom-made basis functions also requires less computational machinery for evolving the system of ODEs for the expansion coefficients, as the additional terms flux terms found in the discontinuous Galerkin approach are no longer necessary.

In aggregate, the presented results illustrate the effectiveness of the scientific machine learning framework for solving PDEs in a spectral fashion. For each of the example PDEs, with the exception of the inviscid Burgers equation, the results are compared against a Fourier solution based on a 128- or 512-mode Fourier expansion and show strong agreement.

The presented framework should not be considered as an alternative to the gold standard Fourier basis functions for periodic domains, nor should it be expected to outperform the Fourier basis functions as they are the optimal choice. The presented results should instead be viewed as a proof-of-concept with the potential to generalize well to complex domains where classical bases are not immediately available for use in a spectral method. Wall-clock times for both solution methods are presented in Appendix J for comparison. Additionally, the ability to truncate to r custom-made basis functions based on a specified singular value threshold can be seen as a form of model reduction without the memory term, more similar to POD. However, unlike POD, the presented framework does not rely directly on snapshots of the PDE to extract the basis functions; therefore, requiring far sparser data sets to extract basis functions that are capable of temporal extrapolation along with extrapolation in terms of input function and PDE parameter space.

Chapter 4
CONCLUSIONS AND FUTURE DIRECTIONS

The two research objectives outlined in Section 1.3 were investigated in Chapter 2 and Chapter 3, with results presented to demonstrate the effectiveness of the time-dependent perturbative renormalization (first research objective) and the scientific machine learning (second research objective) frameworks. The conclusions based on the work presented are outlined in Section 4.1, while opportunities for further investigation are outlined in Section 4.2.

4.1 Conclusions

Key contributions, insights, and results are outlined for the two research objectives below.

1. First research objective:

    - **Key contributions.** A key contribution towards the first research objective was the introduction of the parameter τ to allow for time-dependent renormalization, which ultimately serves as a means to control the time decay of the memory. In addition to the introduction of this parameter, the time-dependent renormalization framework was developed and applied to two nonlinear PDEs; the one-dimensional inviscid Burgers equation and the three-dimensional Euler equations.
    - **Key insights.** A key insight was the fact that the selection of the proper value of τ is necessary for the stable and accurate evolution of reduced order models constructed using the complete memory approximation. Furthermore, the selection of the optimal value of τ is a delicate procedure, and the value is problem specific. However, despite this delicate nature, the estimation of the renormalization

coefficients and τ are possible using only data from before the singularity (one-dimensional inviscid Burgers equation) or before the full system becomes under-resolved (three-dimensional Euler equations). Additionally, if a large enough full system is available, the selection of the optimal τ value can be automated.

- **Key results.** The fourth-order ROMs for both the one-dimensional inviscid Burgers and three-dimensional Euler equations exhibited convergence, and the individual memory terms were layered, indicating the presented framework is indeed perturbative in nature. Results for the evolution of the energy of the fourth-order ROMs of the inviscid Burgers equation were in good agreement with a well-resolved shock-capturing scheme for the temporal domain $t \in [0, 1000]$. Additionally, the results for the peak rate of change of energy and corresponding peak time for fourth-order, $N = 12$ size ROMs of the Euler equations were found to be in good agreement with large-scale direct numerical simulations.

2. Second research objective:

   - **Key contributions.** The key contribution towards the second research objective was the development of a scientific machine learning framework for identifying candidate basis functions from a trained operator neural network and subsequently turning these candidate basis functions into an orthonormal basis for the evolution of dynamical systems using a spectral method. This framework was applied to six one-dimensional PDEs: the advection, advection-diffusion, viscous Burgers, Korteweg–de Vries, Kuramoto–Sivashinsky, and inviscid Burgers equations.
   - **Key insights.** A key insight was the fact that operator neural networks, and more specifically DeepONet, are capable of identifying quality candidate basis functions. Once identified by the DeepONet, the candidate basis functions can be turned into an orthonormal basis using the singular value decomposition. The usage of the Legendre polynomials allows the accurate evaluation of the orthonormal

basis at locations away from the specified quadrature grid. The procedure allows for a reduction in system size where the reduced set of basis functions can be identified through the singular values. Furthermore, based on analysis of the singular value spectrums of each example PDE considered, the constructed basis functions do appear to be truly custom-made for the corresponding PDE for which the DeepONet was trained.

- **Key results.** The constructed sets of custom-made basis functions allow for the evolution of PDEs using a spectral approach and show good agreement with Fourier solutions for the advection, advection-diffusion, viscous Burgers, Korteweg–de Vries, and Kuramoto–Sivashinsky equations, and with a shock-capturing scheme for the inviscid Burgers equation. Furthermore, the framework allows for extrapolation in terms of input function space and temporally (with the exception of the inviscid Burgers equation). Additionally, for the advection, advection-diffusion, viscous Burgers, Korteweg–de Vries, and Kuramoto–Sivashinsky equations, extrapolation in terms of the PDE parameters also shows agreement with Fourier solutions.

4.2 Future directions

Both of these frameworks provide many opportunities for further investigation and extension beyond what was presented. The natural next steps are to apply each of these frameworks to additional PDEs, such as the Korteweg–de Vries–Burgers and nonlinear Schrödinger equations for the first research objective, and the Allen–Cahn equation for the second research objective. Additional items that have been identified as future work for the two research objectives are outlined below.

1. First research objective:

   (a) The revised ansatz for the time-dependent renormalization coefficients $\alpha_i(t) = \beta_i \left[(1/N)t^{-\tau}\right]^i$, results in a reduced condition number compared to $\alpha_i(t) = a_i t^{-i\tau}$;

however, the resulting condition number is still too large to allow for the accurate estimation of the coefficients for large values of N. Furthermore, the condition number increases with increasing N, while at the same time, the coefficients decrease in magnitude. This limits the applicability of the approach to only small resolutions for the ROMs. Additional techniques should be explored for addressing the ill-conditioning, such as lower-rank approximations to the matrix of independent variables [10], Tikhonov regularized regression [8], etc. Approaches that may allow this issue to be circumvented through the use of deep learning methods, such as generative adversarial networks [31], would also be an interesting area of investigation. Furthermore, the possibility of reformulating the Taylor series expansion present in the complete memory approximation, prior to renormalization, using orthogonal polynomials could be investigated. Additionally, approaches that allow for the robust estimation of the renormalization coefficients and optimal τ while utilizing a smaller M' sized full simulation should also be explored. A preliminary investigation into lower-rank approximations is presented in Appendix D.

(b) In conjunction with the investigation into ways to allow for the more robust estimation of the renormalization coefficients and optimal τ, the current MATLAB code base for the three-dimensional Euler equations should be updated and parallelized so that larger full and reduced order system simulations are also possible. Currently, even on a relatively powerful computer, only small M' (and N) values are possible and prevent the estimation of the optimal τ value and resolutions large enough for more detailed analysis and visualization. The results presented focused on producing ROMs that were stable for long times due to the limited full system available, but larger and more finely-resolved full and ROM simulations are required to further probe the three-dimensional Euler equations.

(c) The $\alpha_i(t) = a_i t^{-i\tau}$, and revised $\alpha_i(t) = \beta_i \left[(1/N)t^{-\tau}\right]^i$, are just two possible choices

for the time-dependent renormalization coefficients. More elaborate choices, such as making τ wavenumber and/or order of the memory term dependent, would be an interesting avenue for investigation. Additionally, alternative forms, such as $\alpha_i(t) = a_i e^{-i\tau t}$, should also be considered in future work. Furthermore, the required M' to estimate the renormalization coefficients and the optimal τ, along with the maximum attainable N for any new form, should be investigated.

2. Second research objective:

   (a) Two extensions of the custom-made basis framework were outlined in Section 3.5 with preliminary results provided in the appendices; time-sampling of the custom-made basis functions (Appendix H) and the development of custom-made basis functions that individually satisfy the boundary conditions (Appendix I). Both of these approaches warrant further investigation. For the time-sampling results presented in Appendix H, the same number of quadrature nodes as the frozen-in-time construction were utilized. However, for the time-sampling approach, the usage of a higher-order quadrature rule allows for a greater number of custom-made basis functions to be generated and presents an interesting area of investigation. Additionally, analysis into the performance of the standard custom-made basis functions evolved using a discontinuous Galerkin approach versus the custom-made basis functions that individually satisfy the boundary conditions and are evolved using a Galerkin approach would be intriguing.

   (b) For each of the example PDEs presented, a DeepONet was trained specifically for each PDE. However, this raises an interesting question that warrants further investigation: can the custom-made basis functions identified for a PDE that exhibits more complex dynamics be utilized for one that exhibits less complex dynamics? Preliminary results are presented in Appendix L, where the custom-made basis functions developed for the viscous Burgers equation are used to evolve

the advection equation, and the custom-made basis functions developed for the Korteweg–de Vries equation are used to evolve the viscous Burgers equation.

(c) Development of a fast forward and fast inverse custom-made basis transform. Improvements in the wall-clock times of the custom-made basis results can be achieved through the usage of a pseudo-spectral transform which directly calculates the forward transform (inner product calculation) and inverse transform (series expansion calculation) as shown in Appendix J. Although this provides an improvement in the wall-clock times for nonlinear PDEs over the usage of the triple product integral and represents a preliminary step toward reducing the wall-clock times for nonlinear problems, a fast forward and fast inverse custom-made basis transform developed using DeepONets may allow for additional performance improvements when computing the nonlinear terms in real space. The forward transform based on a DeepONet would have the trunk network fixed to output the custom-made basis functions, and the branch network would be trained so that it would output the expansion coefficients. The inverse transform based on a DeepONet would take as input to the branch network the expansion coefficients, while the input to the trunk network would be the spatial locations. Development of the fast inverse transform is underway, and preliminary results are presented in Appendix M for the viscous Burgers equation.

(d) Optimization of the DeepONet parameters to improve the quality of the custom-made basis functions. Parameters, such as network width, network depth, activation functions, etc., have been preliminarily investigated; however, a more detailed investigation into the full parameter space and how the various parameters impact the resulting custom-made basis functions would be insightful. Furthermore, applying the custom-made basis framework to alternative operator neural network architectures, such as the physics-informed DeepONet [76], would be an interesting area of investigation.

(e) The DeepONet underlying the construction of the custom-made basis functions was trained for 50000 epochs for each of the example PDEs. Ultimately, there needs to be a formalization of the convergence criterion for the DeepONets used to identify the candidate basis functions. This would require further investigation. One potential avenue is the monitoring of convergence of the singular value spectrums as a function of training epoch as opposed to more traditional stopping criteria, such as monitoring the training and testing mean squared errors as functions of training epoch.

(f) All the problems presented were one-dimensional and had periodic boundary conditions. These problems served as a proof-of-concept and allowed for comparisons against Fourier solutions. In these cases, it was not expected that the constructed custom-made basis functions would outperform the optimal Fourier basis functions for periodic domains. The next steps would be to consider alternative boundary conditions using both the standard custom-made basis construction and the construction which uses a feature expansion or hard constraints so that the custom-made basis functions individually satisfy the boundary conditions [47]. Additionally, time-independent problems would be an interesting area of study. Extending the framework to problems on more complex domains in higher dimensions should be investigated. This extension will require the development of alternative approaches to using the Legendre polynomials, such as spline-based interpolation, for evaluation of the custom-made basis functions at locations away from the quadrature grid.

(g) Combining the model reduction framework presented as part of the first research objective with the work presented as part of the second research objective is a very intriguing area of future work. Starting with the one-dimensional inviscid Burgers equation on a periodic domain and adding a memory term to allow for the energy to be drained from the resolved scales would be an interesting first step.

Furthermore, the ability to develop custom-made basis functions that individually satisfy the boundary conditions allows for the application of the Mori-Zwanzig formalism to a wider range of problems beyond periodic domains. The use of the custom-made basis functions that individually satisfy the boundary conditions would likely make the computation of the memory terms more straightforward than if the standard custom-made basis functions were utilized.

In closing, the frameworks developed as part of this dissertation provide a bridge between scientific computing and machine learning and aim to advance the state of the art in the simulation of multiscale systems.

BIBLIOGRAPHY

[1] DS Agafontsev, EA Kuznetsov, and AA Mailybaev. Development of high vorticity structures in incompressible 3d euler equations. *Physics of Fluids*, 27(8):085102, 2015.

[2] Sheldon Axler. *Linear algebra done right*, volume 2. Springer, 2015.

[3] Nathan Baker, Frank Alexander, Timo Bremer, Aric Hagberg, Yannis Kevrekidis, Habib Najm, Manish Parashar, Abani Patra, James Sethian, Stefan Wild, et al. Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence. Technical report, USDOE Office of Science (SC), Washington, DC (United States), 2019.

[4] Jérémie Bec and Konstantin Khanin. Burgers turbulence. *Physics reports*, 447(1-2):1–66, 2007.

[5] Bernstein, D. Optimal prediction of Burgers's equation. *Multiscale Modeling & Simulation*, 6(1):27–52, 2007.

[6] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[7] Mayur P Bonkile, Ashish Awasthi, C Lakshmi, Vijitha Mukundan, and VS Aswin. A systematic literature review of burgers' equation with recent advances. *Pramana*, 90(6):1–21, 2018.

[8] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[9] Boyd, J. P. *Chebyshev and Fourier Spectral Methods*. Dover, 2001.

[10] Brunton, S. L. & Kutz, J. N. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2019.

[11] Canuto, C., Hussaini, M.Y., Quarteroni, A., & Zhang, T. A. *Spectral Methods in Fluid Dynamics*. Springer-Verlag Berlin Heidelberg, 1988.

[12] Anindya Chatterjee. An introduction to the proper orthogonal decomposition. *Current science*, pages 808–817, 2000.

[13] Chen, T. & Chen, H. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995.

[14] Alexandre J Chorin, Ole H Hald, and Raz Kupferman. Optimal prediction and the mori–zwanzig representation of irreversible processes. *Proceedings of the National Academy of Sciences*, 97(7):2968–2973, 2000.

[15] Chorin, A. J. *Vorticity and turbulence*. Springer Science & Business Media New York, 1994.

[16] Chorin, A. J., & Hald, O. H. *Stochastic Tools in Mathematics and Science (Vol. 1)*. New York: Springer, 2009.

[17] Chorin, A. J. & Stinis, P. Problem reduction, renormalization, and memory. *Communications in Applied Mathematics and Computational Science*, 1(1):1–27, 2007.

[18] Chorin, A. J., Hald, O. H., & Kupferman, R. Optimal prediction with memory. *Physica D: Nonlinear Phenomena*, 166(3-4):239–257, 2002.

[19] Bernardo Cockburn and Chi-Wang Shu. The local discontinuous galerkin method for time-dependent convection-diffusion systems. *SIAM Journal on Numerical Analysis*, 35(6):2440–2463, 1998.

[20] Peter Constantin. On the euler equations of incompressible fluids. *Bulletin of the American Mathematical Society*, 44(4):603–621, 2007.

[21] Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[22] Delamotte, B. A hint of renormalization. *American Journal of Physics*, 72(2):170–184, 2004.

[23] Beichuan Deng, Yeonjong Shin, Lu Lu, Zhongqiang Zhang, and George Em Karniadakis. Convergence rate of deeponets for learning operators arising from advection-diffusion equations. *arXiv preprint arXiv:2102.10621*, 2021.

[24] Doering, C. R. & Gibbon, J.D. *Applied analysis of the Navier-Stokes equations*, volume 12. Cambridge University Press, 1995.

[25] Gregory L Eyink. Dissipative anomalies in singular euler flows. *Physica D: Nonlinear Phenomena*, 237(14-17):1956–1968, 2008.

[26] Niklas Fehn, Martin Kronbichler, Peter Munch, and Wolfgang A Wall. Numerical evidence of anomalous energy dissipation in incompressible euler flows: towards grid-converged results for the inviscid taylor–green problem. *Journal of Fluid Mechanics*, 932, 2022.

[27] Myron Flickner, James Hafner, Eduardo J Rodriguez, and Jorge LC Sanz. Periodic quasi-orthogonal spline bases and applications to least-squares curve fitting of digital images. *IEEE Transactions on Image Processing*, 5(1):71–88, 1996.

[28] Uriel Frisch, Zhen Su She, and Olivier Thual. Viscoelastic behaviour of cellular solutions to the kuramoto-sivashinsky model. *Journal of Fluid Mechanics*, 168:221–240, 1986.

[29] John D Gibbon. The three-dimensional euler equations: Where do we stand? *Physica D: Nonlinear Phenomena*, 237(14-17):1894–1904, 2008.

[30] Dror Givon, Raz Kupferman, and Andrew Stuart. Extracting macroscopic dynamics: model problems and algorithms. *Nonlinearity*, 17(6):R55, 2004.

[31] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[32] David Gottlieb and Steven A Orszag. *Numerical analysis of spectral methods: theory and applications*. SIAM, 1977.

[33] Hald, O. H. & Stinis, P. Optimal prediction and the rate of decay for solutions of the euler equations in two and three dimensions. *Proceedings of the National Academy of Sciences*, 104(16):6527–6532, 2007.

[34] Holmes, P., Lumley, J.L., & Berkooz, G. *Turbulence, coherent structures, dynamical systems, and symmetry*. Cambridge monographs on mechanics. Cambridge University Press, Cambridge ; New York, 1996.

[35] James M Hyman and Basil Nicolaenko. The kuramoto-sivashinsky equation: a bridge between pde's and dynamical systems. *Physica D: Nonlinear Phenomena*, 18(1-3):113–126, 1986.

[36] James M Hyman, Basil Nicolaenko, and Stéphane Zaleski. Order and complexity in the kuramoto-sivashinsky model of weakly turbulent interfaces. *Physica D: Nonlinear Phenomena*, 23(1-3):265–292, 1986.

[37] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[38] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[39] Jeffrey C Lagarias, James A Reeds, Margaret H Wright, and Paul E Wright. Convergence properties of the nelder–mead simplex method in low dimensions. *SIAM Journal on optimization*, 9(1):112–147, 1998.

[40] Samuel Lanthaler, Siddhartha Mishra, and George Em Karniadakis. Error estimates for deeponets: A deep learning framework in infinite dimensions. *arXiv preprint arXiv:2102.09618*, 2021.

[41] Peter D Lax. *Hyperbolic systems of conservation laws and the mathematical theory of shock waves*. SIAM, 1973.

[42] LeCun, Y., Bengio, Y., & Hinton, G. Deep learning. *nature*, 521(7553):436–444, 2015.

[43] Randall J LeVeque et al. *Finite volume methods for hyperbolic problems*, volume 31. Cambridge university press, 2002.

[44] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[45] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.

[46] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

[47] Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *arXiv preprint arXiv:2111.05512*, 2021.

[48] Guo Luo and Thomas Y Hou. Potentially singular solutions of the 3d axisymmetric euler equations. *Proceedings of the National Academy of Sciences*, 111(36):12968–12973, 2014.

[49] MATLAB. *R2019b*. The MathWorks Inc., Natick, Massachusetts, 2019.

[50] Brek Meuris, Saad Qadeer, and Panos Stinis. Machine-learning custom-made basis functions for partial differential equations. *arXiv preprint arXiv:2111.05307*, 2021.

[51] Robert M Miura. The korteweg–devries equation: a survey of results. *SIAM review*, 18(3):412–459, 1976.

[52] George Neofotistos, Marios Mattheakis, Georgios D Barmparis, Johanne Hizanidis, Giorgos P Tsironis, and Efthimios Kaxiras. Machine learning with observers predicts complex spatiotemporal behavior. *Frontiers in Physics*, 7:24, 2019.

[53] Peter J Olver. *Introduction to partial differential equations*. Springer, 2014.

[54] Ronald L Panton. *Incompressible flow*. John Wiley & Sons, 2013.

[55] Demetrios T Papageorgiou and Yiorgos S Smyrlis. The route to chaos for the Kuramoto-Sivashinsky equation. *Theoretical and Computational Fluid Dynamics*, 3(1):15–42, 1991.

[56] Parish, E. J. & Duraisamy, K. Non-markovian closure models for large eddy simulations using the Mori-Zwanzig formalism. *Physical Review Fluids*, 2(1):014604–1–014604–28, 2017.

[57] Richard Pasquetti. Spectral vanishing viscosity method for les: sensitivity to the svv control parameters. *Journal of Turbulence*, 6(6):N12, 2005.

[58] Richard H Pletcher, John C Tannehill, and Dale Anderson. *Computational fluid mechanics and heat transfer*. CRC press, 2012.

[59] Jacob Price, Brek Meuris, Madelyn Shapiro, and Panos Stinis. Optimal renormalization of multiscale systems. *Proceedings of the National Academy of Sciences - PNAS*, 118(37):1, 2021.

[60] Price, J. R. *Multiscale Techniques for Nonlinear Dynamical Systems: Applications and Theory*. PhD thesis, University of Washington, 2018.

[61] Price, J., Stinis, P. Renormalized reduced order models with memory for long time prediction. *Multiscale Modeling & Simulation*, 17(1):68–91, 2019.

[62] Tong Qin, Kailiang Wu, and Dongbin Xiu. Data driven governing equations approximation using deep neural networks. *Journal of Computational Physics*, 395:620–635, 2019.

[63] Christopher Rackauckas and Qing Nie. Differentialequations.jl–a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software*, 5(1), 2017.

[64] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Multistep neural networks for data-driven discovery of nonlinear dynamical systems. *arXiv preprint arXiv:1801.01236*, 2018.

[65] Chi-Wang Shu. Discontinuous Galerkin methods: general approach and stability. *Numerical solutions of partial differential equations*, 201, 2009.

[66] Chi-Wang Shu et al. Different formulations of the discontinuous Galerkin method for the viscous terms. *Advances in Scientific Computing*, pages 144–155, 2001.

[67] Charles G Speziale and Peter S Bernard. The energy decay in self-preserving isotropic turbulence revisited. *Journal of Fluid Mechanics*, 241:645–667, 1992.

[68] Panagiotis Stinis. A hybrid method for the inviscid burgers equation. *Discrete & Continuous Dynamical Systems*, 9(4):793, 2003.

[69] Stinis, P. Higher order Mori–Zwanzig models for the Euler equations. *Multiscale Modeling & Simulation*, 6(3):741–760, 2007.

[70] Stinis, P. Renormalized reduced models for singular PDEs. *Communications in Applied Mathematics and Computational Science*, 8(1):39–66, 2013.

[71] Stinis, P. Renormalized Mori–Zwanzig-reduced models for systems without scale separation. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2176):1–13, 2015.

[72] Eitan Tadmor. Convergence of spectral methods for nonlinear conservation laws. *SIAM Journal on Numerical Analysis*, 26(1):30–44, 1989.

[73] Eitan Tadmor. A review of numerical methods for nonlinear partial differential equations. *Bulletin of the American Mathematical Society*, 49(4):507–554, 2012.

[74] Trefethen, L. N. & Bau III, D. *Numerical linear algebra*, volume 50. Siam, 1997.

[75] Angelo Vulpiani and Roberto Livi. *The Kolmogorov legacy in physics*, volume 636. Springer, 2003.

[76] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science advances*, 7(40):eabi8605, 2021.

[77] E Weinan. Machine learning and computational mathematics. *arXiv preprint arXiv:2009.14596*, 2020.

[78] Weinan, E. *Principles of Multiscale Modeling*. Cambridge University Press, 2011.

[79] Nick Winovich, Karthik Ramani, and Guang Lin. Convpde-uq: Convolutional neural networks with quantified uncertainty for heterogeneous elliptic partial differential equations on varied domains. *Journal of Computational Physics*, 394:263–279, 2019.

[80] Keyulu Xu, Mozhi Zhang, Jingling Li, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. How neural networks extrapolate: From feedforward to graph neural networks. *arXiv preprint arXiv:2009.11848*, 2020.

[81] Yan Xu and Chi-Wang Shu. Local discontinuous Galerkin methods for the Kuramoto–Sivashinsky equations and the Ito-type coupled KdV equations. *Computer methods in applied mechanics and engineering*, 195(25-28):3430–3447, 2006.

[82] Jue Yan and Chi-Wang Shu. A local discontinuous Galerkin method for KdV type equations. *SIAM Journal on Numerical Analysis*, 40(2):769–791, 2002.

[83] HQ Yang and AJ Przekwas. A comparative study of advanced shock-capturing schemes applied to Burgers' equation. *Journal of Computational Physics*, 102(1):139–159, 1992.

[84] Norman J Zabusky and Martin D Kruskal. Interaction of "solitons" in a collisionless plasma and the recurrence of initial states. *Physical review letters*, 15(6):240, 1965.

[85] Zwanzig, R. Memory effects in irreversible thermodynamics. *Physical Review*, 124(4):983–992, 1961.

Appendix A

TRANSFORMING A SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS INTO A SYSTEM OF PARTIAL DIFFERENTIAL EQUATIONS

This appendix outlines the procedure for transforming a system of nonlinear ODEs into a system of linear PDEs. This transformation into a system of linear PDEs is required so that the right-hand side of Equation (2.2) can be decomposed. The following proof is based on [16, 60].

Consider a system of nonlinear ODEs of size M,

$$\frac{d\mathbf{u}(\mathbf{u}_0, t)}{dt} = \mathbf{R}(\mathbf{u}(\mathbf{u}_0, t)), \quad \mathbf{u}(\mathbf{u}_0, 0) = \mathbf{u}_0. \tag{A.1}$$

This system can be transformed into a system of linear PDEs,

$$\frac{\partial}{\partial t} e^{t\mathcal{L}} u_{0j} = e^{t\mathcal{L}} \mathcal{L} u_{0j}, \quad \text{for } 1 \leq j \leq M, \tag{A.2}$$

with the Liouville operator $\mathcal{L}$, given by

$$\mathcal{L} = \sum_{i=1}^{M} R_i(\mathbf{u}_0) \frac{\partial}{\partial u_{0i}}. \tag{A.3}$$

Note that $\mathcal{L} u_{0j} = R_j(\mathbf{u}_0)$.

To show this transformation, begin by letting $g(\mathbf{u}_0)$ be a smooth function of $\mathbf{u}_0$ and define $\phi(\mathbf{u}_0, t) = g(\mathbf{u}(\mathbf{u}_0, t))$. Note that $\phi(\mathbf{u}_0, 0) = g(\mathbf{u}(\mathbf{u}_0, 0)) = g(\mathbf{u}_0)$. Next, take the time

derivative of this function

$$\begin{aligned}\frac{\partial \phi(\mathbf{u}_0, t)}{\partial t} &= \sum_i \frac{\partial g(\mathbf{u}(\mathbf{u}_0, t))}{\partial u_{0i}} \frac{\partial u_i(\mathbf{u}_0, t)}{\partial t} \\ &= \sum_i R_i(\mathbf{u}(\mathbf{u}_0, t)) \frac{\partial g(\mathbf{u}(\mathbf{u}_0, t))}{\partial u_{0i}}. \end{aligned} \tag{A.4}$$

To proceed, it must be shown that

$$\sum_i R_i(\mathbf{u}(\mathbf{u}_0, t)) \frac{\partial g(\mathbf{u}(\mathbf{u}_0, t))}{\partial u_{0i}} = \sum_i R_i(\mathbf{u}_0) \frac{\partial g(\mathbf{u}(\mathbf{u}_0, t))}{\partial u_{0i}}, \tag{A.5}$$

which requires proof of the following identity,

$$\mathbf{R}(\mathbf{u}(\mathbf{u}_0, t)) = D_{\mathbf{u}_0}\mathbf{u}(\mathbf{u}_0, t)\mathbf{R}(\mathbf{u}_0), \tag{A.6}$$

where $D_{\mathbf{u}_0}\mathbf{u}(\mathbf{u}_0, t)$ is the Jacobian matrix of $\mathbf{u}(\mathbf{u}_0, t)$ given by

$$D_{u_{0j}} u_i(\mathbf{u}_0, t) = \frac{\partial u_i(\mathbf{u}_0, t)}{\partial u_{0j}}. \tag{A.7}$$

Define $\mathbf{F}(\mathbf{u}_0, t)$ as the difference between the left-hand side and the right-hand side of Equation (A.6)

$$\mathbf{F}(\mathbf{u}_0, t) = \mathbf{R}(\mathbf{u}(\mathbf{u}_0, t)) - D_{\mathbf{u}_0}\mathbf{u}(\mathbf{u}_0, t)\mathbf{R}(\mathbf{u}_0), \tag{A.8}$$

where for $t = 0$,

$$\begin{aligned}\mathbf{F}(\mathbf{u}_0, 0) &= \mathbf{R}(\mathbf{u}(\mathbf{u}_0, 0)) - D_{\mathbf{u}_0}\mathbf{u}(\mathbf{u}_0, 0)\mathbf{R}(\mathbf{u}_0) \\ &= \mathbf{R}(\mathbf{u}_0) - D_{\mathbf{u}_0}(\mathbf{u}_0)\mathbf{R}(\mathbf{u}_0) \\ &= \mathbf{R}(\mathbf{u}_0) - I\mathbf{R}(\mathbf{u}_0) \\ &= 0. \end{aligned} \tag{A.9}$$

Next, take the time derivative of $\mathbf{F}(\mathbf{u}_0, t)$,

$$\begin{aligned}
\frac{\partial \mathbf{F}(\mathbf{u}_0,t)}{\partial t} &= \frac{\partial \mathbf{R}(\mathbf{u}(\mathbf{u}_0,t))}{\partial t} - \frac{\partial}{\partial t}(D_{\mathbf{u}_0}\mathbf{u}(\mathbf{u_0},t)\mathbf{R}(\mathbf{u}_0)) \\
&= (D_{\mathbf{u}_0}\mathbf{R})(\mathbf{u}(\mathbf{u}_0,t))\frac{\partial \mathbf{u}(\mathbf{u}_0,t)}{\partial t} - D_{\mathbf{u}_0}\left(\frac{\partial \mathbf{u}(\mathbf{u}_0,t)}{\partial t}\right)\mathbf{R}(\mathbf{u}_0) \\
&= (D_{\mathbf{u}_0}\mathbf{R})(\mathbf{u}(\mathbf{u}_0,t))\frac{\partial \mathbf{u}(\mathbf{u}_0,t)}{\partial t} - D_{\mathbf{u}_0}(\mathbf{R}(\mathbf{u}(\mathbf{u}_0,t)))\mathbf{R}(\mathbf{u}_0) \\
&= (D_{\mathbf{u}_0}\mathbf{R})(\mathbf{u}(\mathbf{u}_0,t))\mathbf{R}(\mathbf{u}(\mathbf{u}_0,t)) - (D_{\mathbf{u}_0}\mathbf{R})(\mathbf{u}(\mathbf{u}_0,t))D_{\mathbf{u}_0}\mathbf{u}(\mathbf{u}_0,t)\mathbf{R}(\mathbf{u}_0) \\
&= (D_{\mathbf{u}_0}\mathbf{R})(\mathbf{u}(\mathbf{u}_0,t))\left[\mathbf{R}(\mathbf{u}(\mathbf{u}_0,t)) - D_{\mathbf{u}_0}\mathbf{u}(\mathbf{u}_0,t)\mathbf{R}(\mathbf{u}_0)\right] \\
&= (D_{\mathbf{u}_0}\mathbf{R})(\mathbf{u}(\mathbf{u}_0,t))\mathbf{F}(\mathbf{u}_0,t). \qquad \text{(A.10)}
\end{aligned}$$

Equation (A.9) and Equation (A.10) indicate $\mathbf{F}(\mathbf{u}_0, t) = 0$, thereby proving Equation (A.6).

Equation (A.6) can now be substituted into Equation (A.4),

$$\begin{aligned}
\frac{\partial \phi(\mathbf{u}_0,t)}{\partial t} &= \sum_i R_i(\mathbf{u}(\mathbf{u}_0,t))\frac{\partial g(\mathbf{u}(\mathbf{u}_0,t))}{\partial u_{0i}} \\
&= \sum_i \left(\sum_j \frac{\partial u_i(\mathbf{u}_0,t)}{\partial u_{0j}} R_j(\mathbf{u}_0)\right)\frac{\partial g(\mathbf{u}(\mathbf{u}_0,t))}{\partial u_{0i}} \\
&= \sum_j R_j(\mathbf{u}_0)\left(\sum_i \frac{\partial g(\mathbf{u}(\mathbf{u}_0,t))}{\partial u_{0i}}\frac{\partial u_i(\mathbf{u}_0,t)}{\partial u_{0j}}\right) \\
&= \sum_j R_j(\mathbf{u}_0)\frac{\partial g(\mathbf{u}(\mathbf{u}_0,t))}{\partial u_{0j}} \\
&= \mathcal{L}\phi(\mathbf{u}_0,t), \qquad \text{(A.11)}
\end{aligned}$$

where $\mathcal{L} = \sum_i R_i(\mathbf{u}_0)\frac{\partial}{\partial u_{0i}}$. Therefore, $\phi(\mathbf{u}_0, t) = g(\mathbf{u}(\mathbf{u}_0, t))$ is the unique solution to Equation (A.11) with $\phi(\mathbf{u}_0, 0) = g(\mathbf{u}_0)$. If $\phi_j(\mathbf{u}_0, t)$ is considered as the solution to Equation (A.11), then for $g(\mathbf{u}_0) = u_{0j}$ and $\phi_j(\mathbf{u}_0, t) = u_j(\mathbf{u}_0, t)$,

$$\frac{du_j(\mathbf{u}_0,t)}{dt} = \frac{\partial \phi_j(\mathbf{u}_0,t)}{\partial t} = \frac{\partial}{\partial t}e^{t\mathcal{L}}u_{0j} = e^{t\mathcal{L}}\mathcal{L}u_{0j}, \qquad \text{(A.12)}$$

where semigroup notation was utilized to write $\phi_j(\mathbf{u}_0, t) = e^{t\mathcal{L}} u_{0j}$. Equation (A.12) now gives the time evolution of the j-th component of $\mathbf{u}(\mathbf{u}_0, t)$ and matches the form presented in Equation (A.2).

Appendix B

COMPLETE MEMORY APPROXIMATIONS

This appendix contains the first- through fourth-order complete memory approximations for the one-dimensional inviscid Burgers equation (Section B.1) and the three-dimensional Euler equations (Section B.2). Additional details on the construction of the approximation and references to software for automatically generating each of these terms can be found in [59–61].

B.1 1D Inviscid Burgers equation

Starting from the right-hand side of Equation (2.35), the convolution sum can be written as

$$C_k(\mathbf{v}, \mathbf{w}) = -\frac{ik}{2} \sum_{k \in F \cup G} v_p w_q, \tag{B.1}$$

and the corresponding full system as

$$\frac{du_k}{dt} = e^{t\mathcal{L}} C_k(\mathbf{u}_0, \mathbf{u}_0) = C_k(\mathbf{u}, \mathbf{u}). \tag{B.2}$$

The Markov term, Equation (2.9), is

$$R_k^0(\hat{\mathbf{u}}) = P e^{t\mathcal{L}} P \mathcal{L} u_{0k}. \tag{B.3}$$

Using the above definitions,

$$R_k^0(\hat{\mathbf{u}}) = P e^{t\mathcal{L}} P C_k(\mathbf{u}_0, \mathbf{u}_0) = P e^{t\mathcal{L}} C_k(\hat{\mathbf{u}}_0, \hat{\mathbf{u}}_0) = C_k(\hat{\mathbf{u}}, \hat{\mathbf{u}}). \tag{B.4}$$

The first-order memory term, Equation (2.19), is

$$R_k^1(\hat{\mathbf{u}}) = Pe^{t\mathcal{L}}P\mathcal{L}Q\mathcal{L}u_{0k}. \tag{B.5}$$

Utilizing the complimentary projector $Q = I - P$, to expand the term $Q\mathcal{L}u_{0k}$,

$$\mathcal{L}u_{0k} - P\mathcal{L}u_{0k} = -\frac{ik}{2}\sum_{\substack{p+q=k\\ p\in F, q\in G}} u_{0p}u_{0q} - \frac{ik}{2}\sum_{\substack{p+q=k\\ p\in G, q\in F}} u_{0p}u_{0q} - \frac{ik}{2}\sum_{\substack{p+q=k\\ p\in G, q\in G}} u_{0p}u_{0q}, \tag{B.6}$$

and using the defined convolution sum yields

$$R_k^1(\hat{\mathbf{u}}) = Pe^{t\mathcal{L}}P\mathcal{L}[2C_k(\hat{\mathbf{u}}_0, \tilde{\mathbf{u}}_0) + C_k(\tilde{\mathbf{u}}_0, \tilde{\mathbf{u}}_0)]. \tag{B.7}$$

Defining the function $\mathcal{L}\hat{\mathbf{u}}_0 = \hat{\mathbf{C}}(\mathbf{u}_0, \mathbf{u}_0)$, where $\hat{\mathbf{C}}(\mathbf{v}, \mathbf{w})$ is the convolution of $\mathbf{v}$ and $\mathbf{w}$ with the unresolved modes ($k \in G$) set to zero and the function $\mathcal{L}\tilde{\mathbf{u}}_0 = \tilde{\mathbf{C}}(\mathbf{u}_0, \mathbf{u}_0)$, where $\tilde{\mathbf{C}}(\mathbf{v}, \mathbf{w})$ is the convolution of $\mathbf{v}$ and $\mathbf{w}$ with the resolved modes ($k \in F$) set to zero and recognizing $\mathcal{L}$ follows the product rule,

$$\mathcal{L}C_k(\mathbf{v}, \mathbf{w}) = C_k(\mathcal{L}\mathbf{v}, \mathbf{w}) + C_k(\mathbf{v}, \mathcal{L}\mathbf{w}), \tag{B.8}$$

allows Equation (B.7) to be expanded,

$$R_k^1(\hat{\mathbf{u}}) = Pe^{t\mathcal{L}}P[2C_k(\hat{\mathbf{C}}(\mathbf{u}_0, \mathbf{u}_0), \tilde{\mathbf{u}}_0) + 2C_k(\hat{\mathbf{u}}_0, \tilde{\mathbf{C}}(\mathbf{u}_0, \mathbf{u}_0)) + 2C_k(\tilde{\mathbf{C}}(\mathbf{u}_0, \mathbf{u}_0), \tilde{\mathbf{u}}_0)]. \tag{B.9}$$

Applying the remaining operators yields

$$R_k^1(\hat{\mathbf{u}}) = 2C_k(\hat{\mathbf{u}}, \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})). \tag{B.10}$$

The outlined set of rules and functions allows for the determination of the higher-order terms. The final expressions for the second- through fourth-order terms as presented in the

supplemental materials associated with [59] are reproduced for reference:

$$R_k^2(\hat{\mathbf{u}}) = 4C_k(\hat{\mathbf{u}}, \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}))) - 2C_k(\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) \tag{B.11}$$

$$\begin{aligned} R_k^3(\hat{\mathbf{u}}) =& 4C_k(\hat{\mathbf{u}}, 2\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - 2\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) + \tilde{\mathbf{C}}(\hat{\mathbf{u}}, -2\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) + \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}))) \\ &+ \tilde{\mathbf{C}}(\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), \hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) + \tilde{\mathbf{C}}(\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}))) \\ &+ 12C_k(\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), \tilde{\mathbf{C}}(\hat{\mathbf{u}}, -\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) + \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}))) \end{aligned} \tag{B.12}$$

$$\begin{aligned} R_k^4(\hat{\mathbf{u}}) =& 8C_k(\hat{\mathbf{u}}, \tilde{\mathbf{C}}(\hat{\mathbf{u}}, 2\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - 3\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) + \tilde{\mathbf{C}}(\hat{\mathbf{u}}, -5\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) + 3\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}))) \\ &+ \hat{\mathbf{C}}(\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), \hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - 2\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) + 3\hat{\mathbf{C}}(\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) \\ &+ 2\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{C}}(\hat{\mathbf{u}}, -3\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) + 5\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) + \tilde{\mathbf{C}}(\hat{\mathbf{u}}, 3\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}))) \\ &+ \tilde{\mathbf{C}}(\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), -3\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) + 2\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) - \hat{\mathbf{C}}(\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}))) \\ &+ \tilde{\mathbf{C}}(\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), \hat{\mathbf{C}}(\hat{\mathbf{u}}, 3\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - 5\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) + \tilde{\mathbf{C}}(\hat{\mathbf{u}}, -3\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) + \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}))) \\ &+ \tilde{\mathbf{C}}(\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), \hat{\mathbf{C}}(\hat{\mathbf{u}}, -\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) + 3\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) + \tilde{\mathbf{C}}(\hat{\mathbf{u}}, 5\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - 3\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})))) \\ &+ 16C_k(\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), 2\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{C}}(\hat{\mathbf{u}}, -\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) + 2\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) + \tilde{\mathbf{C}}(\hat{\mathbf{u}}, 2\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}))) \\ &+ \tilde{\mathbf{C}}(\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), -\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) + \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) - \tilde{\mathbf{C}}(\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}))) \\ &+ 24C_k(\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})), \tilde{\mathbf{C}}(\hat{\mathbf{u}}, -\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) + 2\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}))) \\ &- 24C_k(\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})), \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}))) \end{aligned} \tag{B.13}$$

B.2 3D Euler equations

Starting from the right-hand side of Equation (2.40), the convolution operator can be written as

$$\mathbf{C}_\mathbf{k}(\mathbf{v}, \mathbf{w}) = -i \sum_{\substack{\mathbf{p}+\mathbf{q}=\mathbf{k} \\ \mathbf{p},\mathbf{q} \in F \cup G}} \mathbf{k} \cdot \mathbf{v}_\mathbf{p} A_\mathbf{k} \mathbf{w}_\mathbf{q}, \tag{B.14}$$

and the corresponding full system as

$$\frac{d\mathbf{u_k}}{dt} = e^{t\mathcal{L}}\mathbf{C_k}(\mathbf{u}_0, \mathbf{u}_0) = \mathbf{C_k}(\mathbf{u}, \mathbf{u}). \tag{B.15}$$

The Markov term, Equation (2.9), is

$$\mathbf{R}^0_{\mathbf{k}}(\hat{\mathbf{u}}) = Pe^{t\mathcal{L}}P\mathcal{L}\mathbf{u}_{0\mathbf{k}}, \tag{B.16}$$

and using the above definitions,

$$\mathbf{R}^0_{\mathbf{k}}(\hat{\mathbf{u}}) = Pe^{t\mathcal{L}}P\mathbf{C_k}(\mathbf{u}_0, \mathbf{u}_0) = Pe^{t\mathcal{L}}\mathbf{C_k}(\hat{\mathbf{u}}_0, \hat{\mathbf{u}}_0) = \mathbf{C_k}(\hat{\mathbf{u}}, \hat{\mathbf{u}}). \tag{B.17}$$

The first-order memory term is

$$\mathbf{R}^1_{\mathbf{k}}(\hat{\mathbf{u}}) = Pe^{t\mathcal{L}}P\mathcal{L}Q\mathcal{L}\mathbf{u}_{0\mathbf{k}}. \tag{B.18}$$

Utilizing the complimentary projector $Q = I - P$, to expand the term $Q\mathcal{L}\mathbf{u}_{0\mathbf{k}}$,

$$\mathcal{L}\mathbf{u}_{0\mathbf{k}} - P\mathcal{L}\mathbf{u}_{0\mathbf{k}} = -i \sum_{\substack{\mathbf{p}+\mathbf{q}=\mathbf{k} \\ \mathbf{p}\in F, \mathbf{q}\in G}} \mathbf{k}\cdot\mathbf{u_p} A_{\mathbf{k}}\mathbf{u_q} - i \sum_{\substack{\mathbf{p}+\mathbf{q}=\mathbf{k} \\ \mathbf{p}\in G, \mathbf{q}\in F}} \mathbf{k}\cdot\mathbf{u_p} A_{\mathbf{k}}\mathbf{u_q} - i \sum_{\substack{\mathbf{p}+\mathbf{q}=\mathbf{k} \\ \mathbf{p}\in G, \mathbf{q}\in G}} \mathbf{k}\cdot\mathbf{u_p} A_{\mathbf{k}}\mathbf{u_q}, \tag{B.19}$$

and using the defined convolution operator yields,

$$\mathbf{R}^1_{\mathbf{k}}(\hat{\mathbf{u}}) = Pe^{t\mathcal{L}}P\mathcal{L}[\mathbf{C_k}(\hat{\mathbf{u}}_0, \tilde{\mathbf{u}}_0) + \mathbf{C_k}(\tilde{\mathbf{u}}_0, \hat{\mathbf{u}}_0) + \mathbf{C_k}(\tilde{\mathbf{u}}_0, \tilde{\mathbf{u}}_0)]. \tag{B.20}$$

Using the fact $\mathcal{L}$ follows the product rule and the definitions for $\hat{\mathbf{C}}$ and $\tilde{\mathbf{C}}$ outlined in Section B.1 allows Equation (B.20) to be expanded,

$$\begin{aligned}\mathbf{R}^1_{\mathbf{k}}(\hat{\mathbf{u}}) = Pe^{t\mathcal{L}}P[&\mathbf{C}_{\mathbf{k}}(\hat{\mathbf{C}}(\mathbf{u}_0,\mathbf{u}_0),\tilde{\mathbf{u}}_0) + \mathbf{C}_{\mathbf{k}}(\hat{\mathbf{u}}_0,\tilde{\mathbf{C}}(\mathbf{u}_0,\mathbf{u}_0))\\ &+ \mathbf{C}_{\mathbf{k}}(\tilde{\mathbf{C}}(\mathbf{u}_0,\mathbf{u}_0),\hat{\mathbf{u}}_0) + \mathbf{C}_{\mathbf{k}}(\tilde{\mathbf{u}}_0,\hat{\mathbf{C}}(\mathbf{u}_0,\mathbf{u}_0))\\ &+ \mathbf{C}_{\mathbf{k}}(\tilde{\mathbf{C}}(\mathbf{u}_0,\mathbf{u}_0),\tilde{\mathbf{u}}_0) + \mathbf{C}_{\mathbf{k}}(\tilde{\mathbf{u}}_0,\tilde{\mathbf{C}}(\mathbf{u}_0,\mathbf{u}_0))],\end{aligned} \tag{B.21}$$

and applying the remaining operators yields

$$\mathbf{R}^1_{\mathbf{k}}(\hat{\mathbf{u}}) = \mathbf{C}_{\mathbf{k}}(\hat{\mathbf{u}},\tilde{\mathbf{C}}(\hat{\mathbf{u}},\hat{\mathbf{u}})) + \mathbf{C}_{\mathbf{k}}(\tilde{\mathbf{C}}(\hat{\mathbf{u}},\hat{\mathbf{u}}),\hat{\mathbf{u}}). \tag{B.22}$$

The right-hand side of Equation (B.22) involves convolutions of the same two terms; therefore, defining the function

$$\mathbf{D}(\mathbf{v},\mathbf{w}) = \mathbf{C}(\mathbf{v},\mathbf{w}) + \mathbf{C}(\mathbf{w},\mathbf{v}), \tag{B.23}$$

and the related convolutions $\hat{\mathbf{D}}(\mathbf{v},\mathbf{w})$ and $\tilde{\mathbf{D}}(\mathbf{v},\mathbf{w})$ allows Equation (B.22) to be rewritten as

$$\mathbf{R}^1_{\mathbf{k}}(\hat{\mathbf{u}}) = \mathbf{D}_{\mathbf{k}}(\hat{\mathbf{u}},\tilde{\mathbf{C}}(\hat{\mathbf{u}},\hat{\mathbf{u}})). \tag{B.24}$$

$\mathcal{L}$ also operates on $\mathbf{D}$ in the same way as on $\mathbf{C}$

$$\begin{aligned}\mathcal{L}\mathbf{D} &= \mathcal{L}\mathbf{C}(\mathbf{v},\mathbf{w}) + \mathcal{L}\mathbf{C}(\mathbf{w},\mathbf{v})\\ &= \mathbf{C}(\mathcal{L}\mathbf{v},\mathbf{w}) + \mathbf{C}(\mathbf{v},\mathcal{L}\mathbf{w}) + \mathbf{C}(\mathcal{L}\mathbf{w},\mathbf{v}) + \mathbf{C}(\mathbf{w},\mathcal{L}\mathbf{v})\\ &= \mathbf{D}(\mathcal{L}\mathbf{v},\mathbf{w}) + \mathbf{D}(\mathbf{v},\mathcal{L}\mathbf{w}).\end{aligned} \tag{B.25}$$

The outlined set of rules and functions allows for the determination of the higher-order terms. The final expressions for the second- through fourth-order terms as presented in the

supplemental materials associated with [59] are reproduced for reference:

$$\mathbf{R}^2_{\mathbf{k}}(\hat{\mathbf{u}}) = \mathbf{D}_{\mathbf{k}}(\hat{\mathbf{u}}, \tilde{\mathbf{D}}(\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), \hat{\mathbf{u}})) - \mathbf{D}_{\mathbf{k}}(\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})). \tag{B.26}$$

$$\begin{aligned}
\mathbf{R}^3_{\mathbf{k}}(\hat{\mathbf{u}}) =& \mathbf{D}_{\mathbf{k}}(\hat{\mathbf{u}}, \tilde{\mathbf{D}}(\hat{\mathbf{u}}, \hat{\mathbf{D}}(\hat{\mathbf{u}}, \hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - 2\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) \\
&+ \tilde{\mathbf{D}}(\hat{\mathbf{u}}, \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - 2\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}))) \\
&+ \tilde{\mathbf{D}}(\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - \hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) \\
&+ \tilde{\mathbf{D}}(\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), \hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}))) \\
&+ 3\mathbf{D}_{\mathbf{k}}(\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), \tilde{\mathbf{D}}(\hat{\mathbf{u}}, \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - \hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}))).
\end{aligned} \tag{B.27}$$

$$
\begin{aligned}
\mathbf{R}^4_{\mathbf{k}}(\hat{\mathbf{u}}) =& \mathbf{D}_{\mathbf{k}}(\hat{\mathbf{u}}, \tilde{\mathbf{D}}(\hat{\mathbf{u}}, \hat{\mathbf{D}}(\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), \hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - 2\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) \\
&+ 3\hat{\mathbf{D}}(\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) \\
&+ \tilde{\mathbf{D}}(\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), 2\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - 3\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) \\
&- \tilde{\mathbf{D}}(\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) \\
&+ \hat{\mathbf{D}}(\hat{\mathbf{u}}, \hat{\mathbf{D}}(\hat{\mathbf{u}}, \hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - 3\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) \\
&+ \tilde{\mathbf{D}}(\hat{\mathbf{u}}, 3\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - 5\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}))) \\
&+ \tilde{\mathbf{D}}(\hat{\mathbf{u}}, \hat{\mathbf{D}}(\hat{\mathbf{u}}, 5\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - 3\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) \\
&+ \tilde{\mathbf{D}}(\hat{\mathbf{u}}, 3\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})))) \\
&+ \tilde{\mathbf{D}}(\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), \hat{\mathbf{D}}(\hat{\mathbf{u}}, 3\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - 5\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) \\
&+ \tilde{\mathbf{D}}(\hat{\mathbf{u}}, \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - 3\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}))) \\
&+ \tilde{\mathbf{D}}(\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), \hat{\mathbf{D}}(\hat{\mathbf{u}}, 3\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - \hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) \\
&+ \tilde{\mathbf{D}}(\hat{\mathbf{u}}, 5\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - 3\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})))) \\
&- 4\mathbf{D}_{\mathbf{k}}(\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), \tilde{\mathbf{D}}(\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), \hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) \\
&+ \tilde{\mathbf{D}}(\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}), \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) \\
&+ \tilde{\mathbf{D}}(\hat{\mathbf{u}}, \hat{\mathbf{D}}(\hat{\mathbf{u}}, \hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - 2\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})) \\
&+ \tilde{\mathbf{D}}(\hat{\mathbf{u}}, \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - 2\hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})))) \\
&- 3\mathbf{D}_{\mathbf{k}}(\tilde{\mathbf{D}}(\hat{\mathbf{u}}, \hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})), \tilde{\mathbf{D}}(\hat{\mathbf{u}}, \hat{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}) - 2\tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}))) \\
&- 3\mathbf{D}_{\mathbf{k}}(\tilde{\mathbf{D}}(\hat{\mathbf{u}}, \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}})), \tilde{\mathbf{D}}(\hat{\mathbf{u}}, \tilde{\mathbf{C}}(\hat{\mathbf{u}}, \hat{\mathbf{u}}))).
\end{aligned}
\tag{B.28}
$$

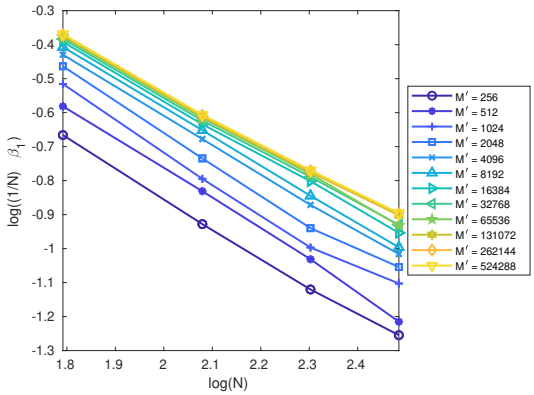
Appendix C

SUPPLEMENTAL OPTIMAL RENORMALIZATION OF MULTISCALE SYSTEMS RESULTS

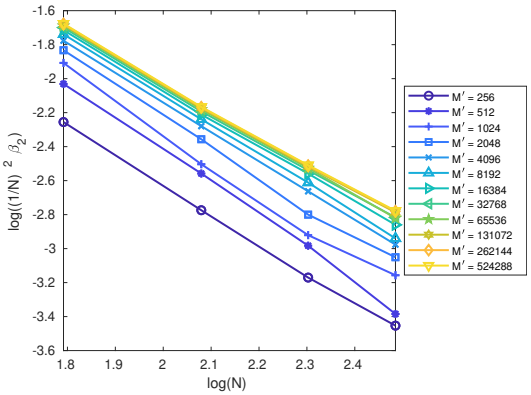
This appendix contains additional results for the one-dimensional inviscid Burgers and three-dimensional Euler equations. Section C.1 presents additional results for the one-dimensional inviscid Burgers equation, and Section C.2 provides additional results for the three-dimensional Euler equations.

C.1 1D Inviscid Burgers equation

Figures C.1, C.2, and C.3 show the convergence of the estimated renormalization coefficients and the optimal value of τ for full systems of size $M' = 256$ to $M' = 524288$. The estimated renormalization coefficients for $n = 1, 2, 3$ order ROMs for $N = 12$ and the optimal τ found for the fourth-order model are shown in Table C.1. Figure C.4 presents the real space solution for the temporal interval $t \in [0, 10]$ for the fourth-order $N = 12$ ROM and the optimal value of τ.

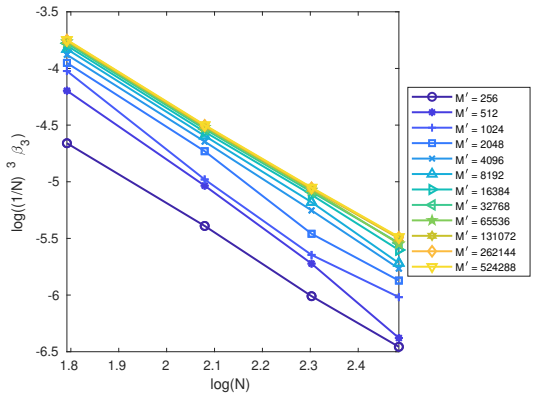


(a) $(1/N)\beta_1$

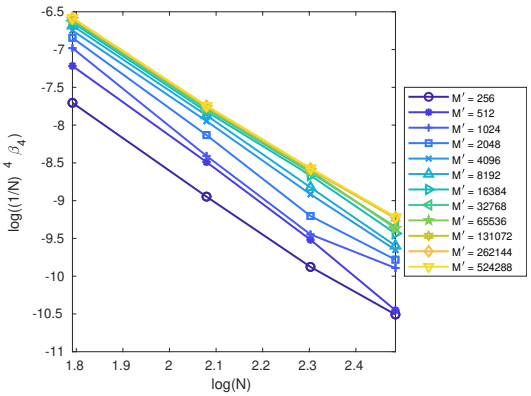


(b) $(1/N)^2\beta_2$

Figure C.1: Estimated renormalization coefficients for the first- and second-order terms for $N = 6, 8, 10, 12$ and for $M' = 256$ to $M' = 524288$.

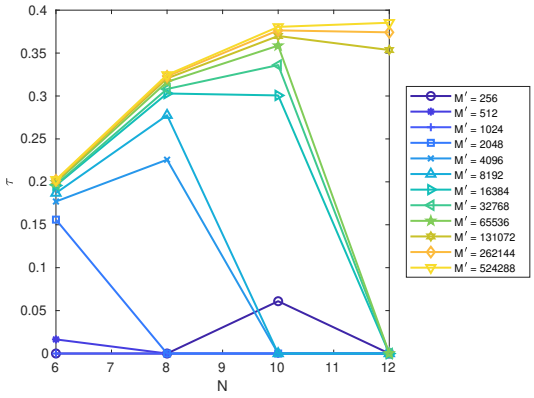


(a) $(1/N)^3\beta_3$



(b) $(1/N)^4\beta_4$

Figure C.2: Estimated renormalization coefficients for the third- and fourth-order terms for $N = 6, 8, 10, 12$ and for $M' = 256$ to $M' = 524288$.

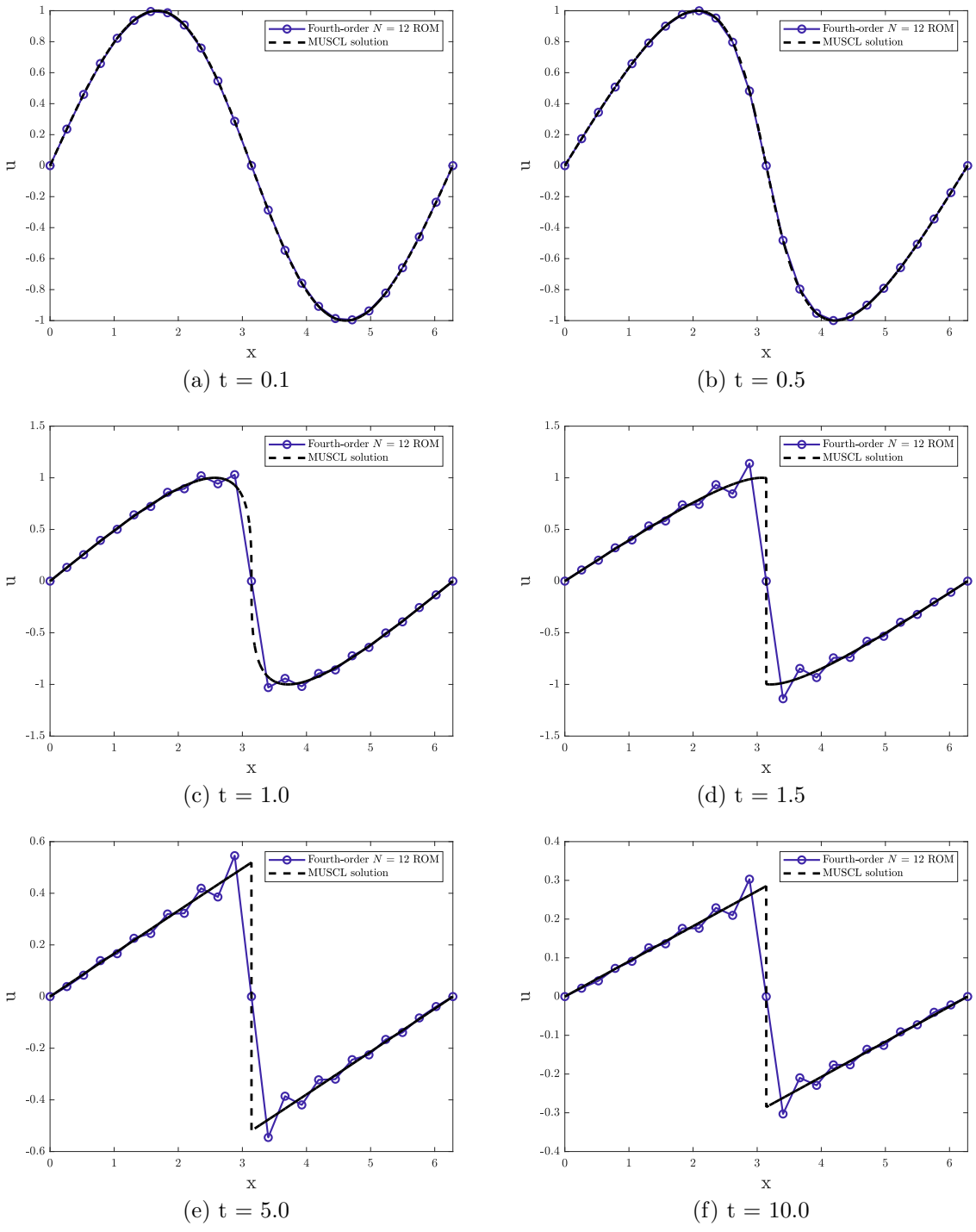


(a) τ

Figure C.3: The optimal value of τ predicted for $N = 6, 8, 10, 12$ and for $M' = 256$ to $M' = 524288$.

	$n = 1$	$n = 2$	$n = 3$
β_1	1.6891	3.4244	3.7228
β_2	–	-2.8144	-5.4238
β_3	–	–	2.8992
$(1/N)\beta_1$	1.4076×10^{-1}	2.8536×10^{-1}	3.1023×10^{-1}
$(1/N)^2\beta_2$	–	-1.9545×10^{-2}	-3.7665×10^{-2}
$(1/N)^3\beta_3$	–	–	1.6778×10^{-3}

Table C.1: Estimated renormalization coefficients for $n = 1, 2, 3$ order ROMs of the inviscid Burgers equation for $N = 12$ and the optimal τ found for the fourth-order model.



(a) t = 0.1 (b) t = 0.5

(c) t = 1.0 (d) t = 1.5

(e) t = 5.0 (f) t = 10.0

Figure C.4: Fourth-order $N = 12$ real space solution of the inviscid Burgers equation for the temporal evolution interval $t \in [0, 10]$.

C.2 3D Euler equations

Table C.2 presents the estimated renormalization coefficients corresponding to $\tau = 0.0, 0.2, \ldots, 0.8$ for fourth-order ROMs and for $N = 12$. The estimated renormalization coefficients for $n = 1, 2, 3$ order ROMs for $N = 12$ and $\tau = 1.0$ are shown in Table C.3. Figures C.5 and C.6 present the contribution of each order memory term to the rate of change of energy in the resolved modes of fourth-order $N = 12$ ROMs for $\tau = 0.6$ and $\tau = 0.8$. Tables C.4, C.5, and C.6 present the peak time and peak value of the total rate of change of energy in the resolved modes for the $N = 6, 8, 10$ fourth-order ROMs and for the stable values of τ.

	$\tau = 0.0$	$\tau = 0.2$	$\tau = 0.4$	$\tau = 0.6$	$\tau = 0.8$
β_1	3.1538	3.2675	3.3840	3.5303	3.6968
β_2	-3.8402	-4.1241	-4.4304	-4.8350	-5.3148
β_3	2.0749	2.3101	2.5780	2.9485	3.4048
β_4	-0.3847	-0.4431	-0.5130	-0.6113	-0.7353
$(1/N)^1\beta_1$	2.6282×10^{-1}	2.7229×10^{-1}	2.8200×10^{-1}	2.9419×10^{-1}	3.0806×10^{-1}
$(1/N)^2\beta_2$	-2.6668×10^{-2}	-2.8640×10^{-2}	-3.0767×10^{-2}	-3.3577×10^{-2}	-3.6908×10^{-2}
$(1/N)^3\beta_3$	1.2008×10^{-3}	1.3369×10^{-3}	1.4919×10^{-3}	1.7063×10^{-3}	1.9704×10^{-3}
$(1/N)^4\beta_4$	-1.8553×10^{-5}	-2.1369×10^{-5}	-2.4739×10^{-5}	-2.9479×10^{-5}	-3.5460×10^{-5}

Table C.2: Estimated renormalization coefficients corresponding to $\tau = 0.0, 0.2, \ldots, 0.8$ found for the fourth-order ROMs of the 3D Euler equations for $N = 12$.

	$n = 1$	$n = 2$	$n = 3$
β_1	1.2868	2.5019	2.8167
β_2	–	-1.7010	-2.5651
β_3	–	–	0.6354
$(1/N)\beta_1$	1.0723×10^{-1}	2.0849×10^{-1}	2.3472×10^{-1}
$(1/N)^2\beta_2$	–	-1.1813×10^{-2}	-1.7813×10^{-2}
$(1/N)^3\beta_3$	–	–	3.6770×10^{-4}

Table C.3: Estimated renormalization coefficients for $n = 1, 2, 3$ order ROMs of the 3D Euler equations for $N = 12$ and $\tau = 1.0$.

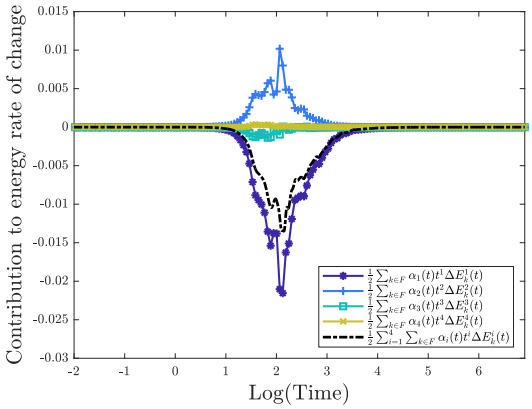


Figure C.5: The contribution of each order memory term to the rate of change of energy in the resolved modes for the $N = 12$ fourth-order ROM of the 3D Euler equations for $\tau = 0.6$.

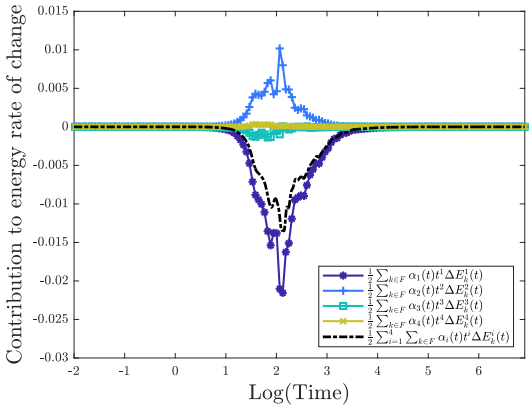


Figure C.6: The contribution of each order memory term to the rate of change of energy in the resolved modes for the $N = 12$ fourth-order ROM of the 3D Euler equations for $\tau = 0.8$.

τ	Peak time	Peak value
0.6	6.2615	-1.6744×10^{-2}
0.8	6.4357	-1.5569×10^{-2}
1.0	6.6409	-1.5114×10^{-2}

Table C.4: Peak time and value of the total rate of change of energy in the resolved modes for the $N = 6$ fourth-order ROMs of the 3D Euler equations for $\tau = 0.6, 0.8, 1.0$.

τ	Peak time	Peak value
0.6	6.4719	-1.5004×10^{-2}
0.8	6.5088	-1.5077×10^{-2}
1.0	6.5445	-1.5006×10^{-2}

Table C.5: Peak time and value of the total rate of change of energy in the resolved modes for the $N = 8$ fourth-order ROMs of the 3D Euler equations for $\tau = 0.6, 0.8, 1.0$.

τ	Peak time	Peak value
0.6	6.6325	-1.4298×10^{-2}
0.8	6.5674	-1.3693×10^{-2}
1.0	6.5522	-1.3298×10^{-2}

Table C.6: Peak time and value of the total rate of change of energy in the resolved modes for the $N = 10$ fourth-order ROMs of the 3D Euler equations for $\tau = 0.6, 0.8, 1.0$.

Appendix D

REDUCING THE ILL-CONDITIONING ASSOCIATED WITH THE ESTIMATION OF THE RENORMALIZATION COEFFICIENTS

This appendix outlines a preliminary investigation into reducing the ill-conditioning associated with estimating the renormalization coefficients. The revised ansatz for the time-dependent renormalization coefficients $\alpha_i(t) = \beta_i \left[(1/N)t^{-\tau}\right]^i$, results in a several order of magnitude reduction in the condition number compared to the $\alpha_i(t) = a_i t^{-i\tau}$; however, the resulting condition number is too large for the accurate estimation of the renormalization coefficients for large values of N. The concept of utilizing the low-rank approximation [10, 74] of the matrix of independent variables was investigated. A rank-r approximation to the matrix of independent variables A, can be obtained using the singular value decomposition

$$\tilde{A} = \sum_{k=1}^{r} \sigma_k \mathbf{u}_k \mathbf{v}_k^* = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^* + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^* + \cdots + \sigma_r \mathbf{u}_r \mathbf{v}_r^*, \tag{D.1}$$

such that $\tilde{A} = \tilde{U}\tilde{S}\tilde{V}^*$, where $\tilde{U}, \tilde{S}, \tilde{V}^*$ are the truncated U, S, V^* matrices. The renormalization coefficients can then be estimated using $\beta = (\tilde{V}\tilde{S}^{-1}\tilde{U}^*)\mathbf{b}$, where $\mathbf{b}$ is the dependent variable.

Rank-$r = 4, 3, 2$ results for $N = 6, 8, \ldots, 24$ for the inviscid Burgers equation are shown in Tables D.1, D.2, and D.3 for fourth-order ROMs. $M' = 524288$ and $\tau = 0.4$ for all the results presented, and in each of the tables, the condition number is presented as the ratio of the largest-to-smallest singular value, i.e., σ_1/σ_r. The approximations for $r < 4$ improve the condition number of the matrix $\tilde{A}$, but the signs found for each of the renormalization coefficients no longer match the signs of the coefficients in Equation (2.18) $\frac{(-1)^{i+1}}{i!}$. The lack

of consistency with the signs leads to ROMs which are less accurate and even unstable in some cases when $r = 2$. Furthermore, rank-r approximations do not aid in the prediction of the optimal τ values for larger values of N.

	$N = 6$	$N = 8$	$N = 10$	$N = 12$	$N = 14$
β_1	3.9011	4.3180	4.6256	4.8976	5.1493
β_2	-6.0844	-7.1709	-8.1057	-8.9439	-9.7312
β_3	4.4992	5.4920	6.3700	7.1371	7.8725
β_4	-1.4935	-1.7093	-1.8858	-2.0604	-2.2542
σ_1/σ_4	1.2141×10^2	1.5503×10^2	1.9795×10^2	2.4261×10^2	2.8688×10^2
	$N = 16$	$N = 18$	$N = 20$	$N = 22$	$N = 24$
β_1	5.3852	5.6080	5.8196	6.0218	6.2157
β_2	-10.4896	-11.2300	-11.9586	-12.6791	-13.3939
β_3	8.6094	9.3602	10.1298	10.9197	11.7301
β_4	-2.4700	-2.7067	-2.9628	-3.2368	-3.5272
σ_1/σ_4	3.3045×10^2	3.7330×10^2	4.1537×10^2	4.5659×10^2	4.9218×10^2

Table D.1: Estimated renormalization coefficients found for the fourth-order ROMs of the inviscid Burgers equation for $N = 6, 8, \ldots, 24$ and $r = 4$.

	$N = 6$	$N = 8$	$N = 10$	$N = 12$	$N = 14$
β_1	3.1858	2.6247	2.5345	2.6740	2.8657
β_2	-2.3193	-1.3806	-1.4211	-1.7036	-2.0177
β_3	-1.6404	-1.4455	-1.3640	-1.4437	-1.5556
β_4	0.4769	0.9176	1.3242	1.6068	1.8245
σ_1/σ_3	4.1153×10^1	3.6182×10^1	3.0021×10^1	2.9530×10^1	3.2402×10^1
	$N = 16$	$N = 18$	$N = 20$	$N = 22$	$N = 24$
β_1	3.0615	3.2487	3.4245	3.5887	3.7419
β_2	-2.3247	-2.6175	-2.8950	-3.1580	-3.4073
β_3	-1.6645	-1.7632	-1.8515	-1.9303	-2.0011
β_4	2.0084	2.1717	2.3203	2.4577	2.5858
σ_1/σ_3	3.5970×10^1	3.9447×10^1	4.2660×10^1	4.5586×10^1	4.8242×10^1

Table D.2: Estimated renormalization coefficients found for the fourth-order ROMs of the inviscid Burgers equation for $N = 6, 8, \ldots, 24$ and $r = 3$.

	$N = 6$	$N = 8$	$N = 10$	$N = 12$	$N = 14$
β_1	1.0027	1.2493	2.1182	2.9528	3.0572
β_2	0.0365	-0.6568	-1.4312	-1.3521	-0.8321
β_3	0.3654	-0.0792	-0.9985	-1.6175	-1.5181
β_4	1.4330	2.0438	1.9546	0.6751	-0.2237
σ_1/σ_2	8.1410×10^0	1.1910×10^1	1.5138×10^1	1.5742×10^1	1.4684×10^1
	$N = 16$	$N = 18$	$N = 20$	$N = 22$	$N = 24$
β_1	2.9840	2.9135	2.8635	2.8305	2.8099
β_2	-0.4890	-0.2805	-0.1449	-0.0507	0.0181
β_3	-1.3267	-1.1794	-1.0727	-0.9942	-0.9351
β_4	-0.6037	-0.7849	-0.8859	-0.9493	-0.9930
σ_1/σ_2	1.3566×10^1	1.2689×10^1	1.2026×10^1	1.1516×10^1	1.1117×10^1

Table D.3: Estimated renormalization coefficients found for the fourth-order ROMs of the inviscid Burgers equation for $N = 6, 8, \ldots, 24$ and $r = 2$.

Appendix E

GENERATION OF GROUND TRUTH DATA AND DEEPONET TRAINING PARAMETERS

This appendix presents the distribution utilized to generate the initial conditions, the procedure for generating the ground truth solutions, and the network parameters (see Table E.1) for the DeepONets used for the construction presented in Section 3.2.1. All branch layers include bias, and the output layer of the branch network does not apply an activation function. Additionally, the network width was constant for all layers. The function $f(\sin^2(x/2))$ was sampled from a mean zero Gaussian random field with covariance kernel,

$$\kappa_l(x_1, x_2) = e^{\frac{-\|x_1 - x_2\|^2}{2l^2}}, \quad l = 0.5, \tag{E.1}$$

to generate the random training and testing initial conditions. Twenty-five example initial conditions are shown in Figure E.1.

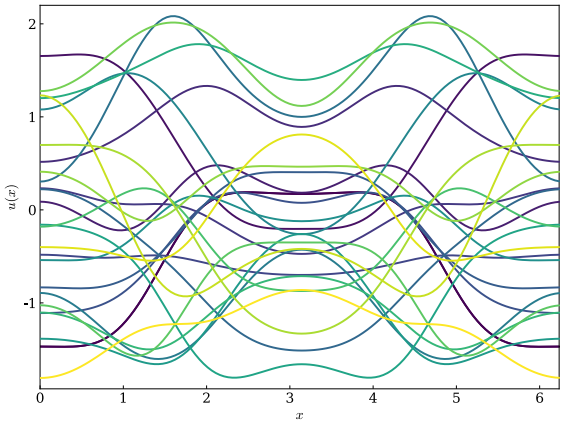


Figure E.1: Twenty-five example initial conditions sampled from the Gaussian random field.

The ground truth data for the advection, advection-diffusion, and viscous Burgers equation was generated by writing the solutions in terms of $M = 128$ Fourier modes,

$$u_{\mathrm{G}}^{M}(t,x) = \sum_{k=-M/2+1}^{M/2-1} \hat{u}_k(t) e^{ikx}. \tag{E.2}$$

The resulting systems of differential equations were solved for $t \in [0,1]$ using a Runge–Kutta–Dormand–Prince integrator with adaptive step size, relative error tolerance 10^{-10}, and absolute tolerance 10^{-14} [63]. The solution was saved at time values 10^{-3} apart for the linear and 10^{-4} for the nonlinear PDE examples. The convolution sum that results from using Equation (E.2) for solving the viscous Burgers equation was evaluated by padding the Fourier solution using the 3/2-rule for de-aliasing [11], transforming the solution to real space, and then computing the fast Fourier transform (FFT) of the product of the real space solution with itself.

The ground truth data for the Korteweg–de Vries and Kuramoto–Sivashinsky equations was generated by writing the solution in terms of $M = 512$ Fourier modes. The 3/2-rule and the pseudo-spectral transform were again utilized to evaluate the resulting convolution sums. The resulting system of differential equations for the Korteweg–de Vries equation was solved for $t \in [0,1]$ using an explicit, singly diagonal, implicit Runge–Kutta integrator with adaptive step size, relative error tolerance 10^{-8}, and absolute tolerance 10^{-12} [63]. The resulting system of differential equations for the Kuramoto–Sivashinsky equation was solved for $t \in [0,1]$ using a Crank–Nicolson integrator with adaptive step size, relative error tolerance 10^{-8}, and absolute tolerance 10^{-12} [63]. To train the DeepONet, the solution for the Korteweg–de Vries and Kuramoto–Sivashinsky equation was saved 10^{-4} time units apart and downsampled from the 512 solution locations to the 128 uniformly spaced sensor locations.

The ground truth data for the inviscid Burgers equation was generated by utilizing a MUSCL (monotonic upwind scheme for conservation laws) scheme with a second-order Roe scheme for the flux and a minmod slope limiter [58, 83]. The spatial domain was discretized

using 4096 points, and the resulting equations were solved for $t \in [0, 1]$ using a Bogacki–Shampine 3/2 integrator with adaptive step size, relative error tolerance 10^{-6}, and absolute tolerance 10^{-8} [63]. To train the DeepONet, the solution was saved 10^{-4} time units apart and downsampled from the 4096 solution locations to the 128 uniformly spaced sensor locations.

Parameter	Setting
Activation functions	Tanh
Optimizer	Adam
Error	Mean squared error
Learning rate	1×10^{-5}
Number of training epochs	50000
Number of sensors	128
Number of solution locations	100
Number of training initial conditions	500
Number of testing initial conditions	1000
Branch net depth	2
Branch net width	128
Trunk net depth	3
Trunk net width	128
Weight initialization	Glorot uniform
Bias initialization	Zero
Mini-batch size	100

Table E.1: Parameter settings for training the DeepONets.

Appendix F

APPLICATION OF BOUNDARY CONDITIONS USING THE CUSTOM-MADE BASIS FUNCTIONS AND DISCONTINUOUS GALERKIN METHODS

This appendix outlines the procedure for imposing boundary conditions in a Galerkin method when the custom-made basis functions do not individually satisfy the boundary conditions. The implementation outlined is for applying non-periodic custom-made basis functions to problems with periodic boundary conditions and follows the discontinuous Galerkin approach. In the following sections, the appropriate boundary condition treatment is presented for the one-dimensional advection (F.1), diffusion (F.2), inviscid Burgers (F.3), reduced Korteweg-de Vries (F.4), and reduced Kuramoto-Sivashinsky (F.5) equations. All the presented examples can then be accommodated by combining these treatments.

F.1 1D Advection equation

Consider the one-dimensional advection equation,

$$\frac{\partial u}{\partial t} + \alpha \frac{\partial u}{\partial x} = 0, \quad x \in [0, 2\pi], \tag{F.1}$$

where the choice of α determines the flow direction. Begin by applying the Galerkin condition,

$$\left\langle \phi_m, \frac{\partial u}{\partial t} + \alpha \frac{\partial u}{\partial x} \right\rangle = 0, \quad \text{for } 1 \leq m \leq r, \tag{F.2}$$

integrating by parts and rewriting the resulting flux term as a surface integral which yields

$$\left\langle \phi_m, \frac{\partial u}{\partial t} \right\rangle = \left\langle \alpha \frac{\partial}{\partial x} \phi_m, u \right\rangle - \int_{\partial \Omega} \hat{\mathbf{n}} \cdot \alpha u_B(s) \phi_m(s) ds, \tag{F.3}$$

where $\hat{\mathbf{n}}$ is the outward facing normal and u_B is the appropriate choice of u at the boundary. Expanding the surface integral in (F.3), utilizing the expansion in terms of the custom basis functions $u^r(t,x) = \sum_{i=1}^{r} a_i(t)\phi_i(x)$, and incorporating the periodic boundary conditions [66], yields a system ODEs for the expansion coefficients,

$$\frac{da_m(t)}{dt} = \sum_{i=1}^{r} \left[\alpha \left\langle \frac{d}{dx}\phi_m, \phi_i \right\rangle - |\alpha| \left(\phi_m(x_{\text{out}}) - \phi_m(x_{\text{in}})\right) \phi_i(x_{\text{out}}) \right] a_i(t), \tag{F.4}$$

for $1 \leq m \leq r$ and where $x_{\text{in}} = 0$ and $x_{\text{out}} = 2\pi$ for $\alpha \geq 0$ and $x_{\text{in}} = 2\pi$ and $x_{\text{out}} = 0$ for $\alpha < 0$. The initial condition for this system of equations is specified as

$$a_m(0) = \langle \phi_m, u(0,\cdot) \rangle, \quad m = 1, \ldots, r. \tag{F.5}$$

F.2 1D Diffusion equation

Consider the one-dimensional diffusion equation given by

$$\frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial x^2} = 0, \quad x \in [0, 2\pi], \tag{F.6}$$

where ν is the viscosity. Begin by rewriting (F.6) as a system of first-order equations,

$$\frac{\partial u}{\partial t} - \nu \frac{\partial q}{\partial x} = 0, \;\; q - \frac{\partial u}{\partial x} = 0, \tag{F.7}$$

and applying the Galerkin condition to the two equations,

$$\left\langle \phi_m, \frac{\partial u}{\partial t} - \nu \frac{\partial q}{\partial x} \right\rangle = 0, \quad \left\langle \phi_m, q - \frac{\partial u}{\partial x} \right\rangle = 0. \tag{F.8}$$

Integrating each equation by parts and rewriting the resulting flux term as a surface integral yields

$$\begin{aligned}\left\langle \phi_m, \frac{\partial u}{\partial t} \right\rangle &= -\nu \left\langle \frac{\partial}{\partial x}\phi_m, q \right\rangle + \nu \int_{\partial\Omega} \hat{\mathbf{n}} \cdot q_B(s)\phi_m(s)ds, \\ \langle \phi_m, q \rangle &= -\left\langle \frac{\partial}{\partial x}\phi_m, u \right\rangle + \int_{\partial\Omega} \hat{\mathbf{n}} \cdot u_B(s)\phi_m(s)ds,\end{aligned} \tag{F.9}$$

where $\hat{\mathbf{n}}$ is the outward facing normal, q_B is the appropriate choice of q at the boundary, and u_B is the appropriate choice of u at the boundary. Expanding the surface integrals, utilizing the expansions in terms of the custom basis functions $u^r(t,x) = \sum_{i=1}^r a_i(t)\phi_i(x)$, $q^r(t,x) = \sum_{j=1}^r b_j(t)\phi_j(x)$, and incorporating the periodic boundary conditions [19,66] yields

$$\frac{da_m(t)}{dt} = \nu \sum_{j=1}^{r} \left[-\left\langle \frac{d}{dx}\phi_m, \phi_j \right\rangle + (\phi_m(x_{\text{out}}) - \phi_m(x_{\text{in}}))\phi_j(x_{\text{out}}) \right] b_j(t), \tag{F.10}$$

$$b_m(t) = \sum_{i=1}^{r} \left[-\left\langle \frac{d}{dx}\phi_m, \phi_i \right\rangle + (\phi_m(x_{\text{out}}) - \phi_m(x_{\text{in}}))\phi_i(x_{\text{in}}) \right] a_i(t). \tag{F.11}$$

Combining the equations (F.10) and (F.11) yields a system of ODEs for the expansion coefficients,

$$\begin{aligned}\frac{da_m(t)}{dt} = \nu \sum_{j=1}^{r} &\left[-\left\langle \frac{d}{dx}\phi_m, \phi_j \right\rangle + (\phi_m(x_{\text{out}}) - \phi_m(x_{\text{in}}))\phi_j(x_{\text{out}}) \right] \cdot \\ &\sum_{i=1}^{r} \left[-\left\langle \frac{d}{dx}\phi_j, \phi_i \right\rangle + (\phi_j(x_{\text{out}}) - \phi_j(x_{\text{in}}))\phi_i(x_{\text{in}}) \right] a_i(t),\end{aligned} \tag{F.12}$$

for $1 \leq m \leq r$ and where $x_{\text{in}} = 0$, $x_{\text{out}} = 2\pi$. The initial condition for this system of equations is specified from the initial condition $u(0,x)$ by (F.5).

F.3 1D Inviscid Burgers equation

Consider the one-dimensional inviscid Burgers equation,

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{u^2}{2}\right) = 0, \quad x \in [0, 2\pi]. \tag{F.13}$$

Set $f(u) = u^2/2$ and begin by applying the Galerkin condition,

$$\left\langle \phi_m, \frac{\partial u}{\partial t} + \frac{\partial u}{\partial x}\left(f(u)\right) \right\rangle = 0, \tag{F.14}$$

integrating by parts and rewriting the resulting flux term as a surface integral which yields

$$\left\langle \phi_m, \frac{\partial u}{\partial t} \right\rangle = \left\langle \frac{\partial}{\partial x}\phi_m, f(u) \right\rangle - \int_{\partial\Omega} \hat{\mathbf{n}} \cdot f^* \phi_m(s) ds, \tag{F.15}$$

where $\hat{\mathbf{n}}$ is the outward facing normal and f^* represents the numerical flux term. Expanding the surface integral in (F.15) yields a system of ODEs for the expansion coefficients,

$$\frac{da_m(t)}{dt} = \left\langle \frac{\partial}{\partial x}\phi_m, f(u) \right\rangle - f^*(u(t,2\pi), u(t,0))(\phi_m(2\pi) - \phi_m(0)), \tag{F.16}$$

for $1 \leq m \leq r$ and where the numerical flux is specified as

$$f^*(u(t,2\pi), u(t,0)) = f\left(\frac{u(t,2\pi) + |u(t,2\pi)|}{2} - \frac{u(t,0) - |u(t,0)|}{2} \right). \tag{F.17}$$

Refer to [65] for additional discussion on the numerical flux. The initial condition for this system of equations is specified from the initial condition $u(0,x)$ by (F.5).

F.4 1D Reduced Korteweg–de Vries equation

Consider the reduced Korteweg–de Vries equation,

$$\frac{\partial u}{\partial t} + \delta^2 \frac{\partial^3 u}{\partial x^3} = 0, \quad x \in [0, 2\pi], \tag{F.18}$$

where δ is the strength of dispersion. Begin by rewriting (F.18) as a system of first-order equations,

$$\frac{\partial u}{\partial t} + \delta^2 \frac{\partial q}{\partial x} = 0, \;\; q - \frac{\partial p}{\partial x} = 0, \;\; p - \frac{\partial u}{\partial x} = 0, \tag{F.19}$$

and applying the Galerkin condition to the three equations,

$$\left\langle \phi_m, \frac{\partial u}{\partial t} + \delta^2 \frac{\partial q}{\partial x} \right\rangle = 0, \quad \left\langle \phi_m, q - \frac{\partial p}{\partial x} \right\rangle = 0, \quad \left\langle \phi_m, p - \frac{\partial u}{\partial x} \right\rangle = 0. \tag{F.20}$$

Integrating each equation by parts and rewriting the resulting flux term as a surface integral yields

$$\begin{aligned} \left\langle \phi_m, \frac{\partial u}{\partial t} \right\rangle &= \delta^2 \left\langle \frac{\partial}{\partial x}\phi_m, q \right\rangle - \delta^2 \int_{\partial\Omega} \hat{\mathbf{n}} \cdot q_B(s)\phi_m(s) ds, \\ \langle \phi_m, q \rangle &= -\left\langle \frac{\partial}{\partial x}\phi_m, p \right\rangle + \int_{\partial\Omega} \hat{\mathbf{n}} \cdot p_B(s)\phi_m(s) ds, \\ \langle \phi_m, p \rangle &= -\left\langle \frac{\partial}{\partial x}\phi_m, u \right\rangle + \int_{\partial\Omega} \hat{\mathbf{n}} \cdot u_B(s)\phi_m(s) ds, \end{aligned} \tag{F.21}$$

where $\hat{\mathbf{n}}$ is the outward facing normal, q_B is the appropriate choice of q at the boundary, p_B is the appropriate choice of p at the boundary, and u_B is the appropriate choice of u at the boundary. Expanding the surface integrals, utilizing the expansions in terms of the custom basis functions $u^r(x,t) = \sum_{i=1}^r a_i(t)\phi_i(x)$, $q^r(x,t) = \sum_{j=1}^r b_j(t)\phi_j(x)$, $p^r(x,t) = \sum_{l=1}^r c_l(t)\phi_l(x)$, and incorporating the periodic boundary conditions [82] yields,

$$\frac{da_m(t)}{dt} = \delta^2 \sum_{j=1}^r \left[\left\langle \frac{d}{dx}\phi_m, \phi_j \right\rangle - (\phi_m(x_{\text{out}}) - \phi_m(x_{\text{in}}))\phi_j(x_{\text{in}}) \right] b_j(t), \tag{F.22}$$

$$b_m(t) = \sum_{l=1}^r \left[-\left\langle \frac{d}{dx}\phi_m, \phi_l \right\rangle + (\phi_m(x_{\text{out}}) - \phi_m(x_{\text{in}}))\phi_l(x_{\text{in}}) \right] c_l(t), \tag{F.23}$$

$$c_m(t) = \sum_{i=1}^r \left[-\left\langle \frac{d}{dx}\phi_m, \phi_i \right\rangle + (\phi_m(x_{\text{out}}) - \phi_m(x_{\text{in}}))\phi_i(x_{\text{out}}) \right] a_i(t). \tag{F.24}$$

Combining equations (F.22), (F.23), and (F.24) yields a system of ODEs for the expansion coefficients,

$$\begin{aligned}\frac{da_m(t)}{dt} = \delta^2 \sum_{j=1}^{r} \left[\left\langle \frac{d}{dx}\phi_m, \phi_j \right\rangle - (\phi_m(x_{\text{out}}) - \phi_m(x_{\text{in}}))\phi_j(x_{\text{in}}) \right] \cdot \\ \sum_{l=1}^{r} \left[- \left\langle \frac{d}{dx}\phi_j, \phi_l \right\rangle + (\phi_j(x_{\text{out}}) - \phi_j(x_{\text{in}}))\phi_l(x_{\text{in}}) \right] \cdot \\ \sum_{i=1}^{r} \left[- \left\langle \frac{d}{dx}\phi_l, \phi_i \right\rangle + (\phi_l(x_{\text{out}}) - \phi_l(x_{\text{in}}))\phi_i(x_{\text{out}}) \right] a_i(t),\end{aligned} \tag{F.25}$$

for $1 \leq m \leq r$ and where $x_{\text{in}} = 0$, $x_{\text{out}} = 2\pi$. The initial condition for this system of equations is specified from the initial condition $u(0, x)$ by (F.5).

F.5 1D Reduced Kuramoto–Sivashinsky equation

Consider the 1-D reduced Kuramoto–Sivashinsky equation,

$$\frac{\partial u}{\partial t} + \beta \frac{\partial^4 u}{\partial x^4} = 0, \tag{F.26}$$

where β is the viscosity. Begin by rewriting (F.26) as a system of first-order equations,

$$\frac{\partial u}{\partial t} + \beta \frac{\partial q}{\partial x} = 0, \quad q - \frac{\partial p}{\partial x} = 0, \quad p - \frac{\partial w}{\partial x} = 0, \quad w - \frac{\partial u}{\partial x} = 0, \tag{F.27}$$

and applying the Galerkin condition to the four equations,

$$\left\langle \phi_m, \frac{\partial u}{\partial t} + \beta \frac{\partial q}{\partial x} \right\rangle = 0, \quad \left\langle \phi_m, q - \frac{\partial p}{\partial x} \right\rangle = 0, \quad \left\langle \phi_m, p - \frac{\partial w}{\partial x} \right\rangle = 0, \quad \left\langle \phi_m, w - \frac{\partial u}{\partial x} \right\rangle = 0. \tag{F.28}$$

Integrating each equation by parts and rewriting the resulting flux term as a surface integral yields

$$\begin{aligned}
\left\langle \phi_m, \frac{\partial u}{\partial t} \right\rangle &= \beta \left\langle \frac{\partial}{\partial x}\phi_m, q \right\rangle - \beta \int_{\partial\Omega} \hat{\mathbf{n}} \cdot q_B(s)\phi_m(s)ds, \\
\langle \phi_m, q \rangle &= -\left\langle \frac{\partial}{\partial x}\phi_m, p \right\rangle + \int_{\partial\Omega} \hat{\mathbf{n}} \cdot p_B(s)\phi_m(s)ds, \\
\langle \phi_m, p \rangle &= -\left\langle \frac{\partial}{\partial x}\phi_m, w \right\rangle + \int_{\partial\Omega} \hat{\mathbf{n}} \cdot w_B(s)\phi_m(s)ds, \\
\langle \phi_m, w \rangle &= -\left\langle \frac{\partial}{\partial x}\phi_m, u \right\rangle + \int_{\partial\Omega} \hat{\mathbf{n}} \cdot u_B(s)\phi_m(s)ds,
\end{aligned} \tag{F.29}$$

where $\hat{\mathbf{n}}$ is the outward facing normal, q_B is the appropriate choice of q at the boundary, p_B is the appropriate choice of p at the boundary, w_B is the appropriate choice of w at the boundary, and u_B is the appropriate choice of u at the boundary. Expanding the surface integrals, utilizing the expansions in terms of the custom basis functions $u^r(x,t) = \sum_{i=1}^{r} a_i(t)\phi_i(x)$, $q^r(x,t) = \sum_{j=1}^{r} b_j(t)\phi_j(x)$, $p^r(x,t) = \sum_{l=1}^{r} c_l(t)\phi_l(x)$, $w^r(x,t) = \sum_{z=1}^{r} d_z(t)\phi_z(x)$, and incorporating the periodic boundary conditions [81] yields,

$$\frac{da_m(t)}{dt} = \beta \sum_{j=1}^{r} \left[\left\langle \frac{d}{dx}\phi_m, \phi_j \right\rangle - (\phi_m(x_{\text{out}}) - \phi_m(x_{\text{in}}))\phi_j(x_{\text{in}}) \right] b_j(t), \tag{F.30}$$

$$b_m(t) = \sum_{l=1}^{r} \left[-\left\langle \frac{d}{dx}\phi_m, \phi_l \right\rangle + (\phi_m(x_{\text{out}}) - \phi_m(x_{\text{in}}))\phi_l(x_{\text{out}}) \right] c_l(t), \tag{F.31}$$

$$c_m(t) = \sum_{z=1}^{r} \left[-\left\langle \frac{d}{dx}\phi_m, \phi_z \right\rangle + (\phi_m(x_{\text{out}}) - \phi_m(x_{\text{in}}))\phi_z(x_{\text{out}}) \right] d_z(t), \tag{F.32}$$

$$d_m(t) = \sum_{i=1}^{r} \left[-\left\langle \frac{d}{dx}\phi_m, \phi_i \right\rangle + (\phi_m(x_{\text{out}}) - \phi_m(x_{\text{in}}))\phi_i(x_{\text{in}}) \right] a_i(t). \tag{F.33}$$

Combining equations (F.30), (F.31), (F.32), and (F.33) yields a system of ODEs for the expansion coefficients,

$$\begin{aligned}\frac{da_m(t)}{dt} = \beta \sum_{j=1}^{r} \left[\left\langle \frac{d}{dx}\phi_m, \phi_j \right\rangle - (\phi_m(x_{\text{out}}) - \phi_m(x_{\text{in}}))\phi_j(x_{\text{in}}) \right] \cdot \\ \sum_{l=1}^{r} \left[- \left\langle \frac{d}{dx}\phi_j, \phi_l \right\rangle + (\phi_j(x_{\text{out}}) - \phi_j(x_{\text{in}}))\phi_l(x_{\text{out}}) \right] \cdot \\ \sum_{z=1}^{r} \left[- \left\langle \frac{d}{dx}\phi_l, \phi_z \right\rangle + (\phi_l(x_{\text{out}}) - \phi_l(x_{\text{in}}))\phi_z(x_{\text{out}}) \right] \cdot \\ \sum_{i=1}^{r} \left[- \left\langle \frac{d}{dx}\phi_z, \phi_i \right\rangle + (\phi_z(x_{\text{out}}) - \phi_z(x_{\text{in}}))\phi_i(x_{\text{in}}) \right] a_i(t),\end{aligned} \tag{F.34}$$

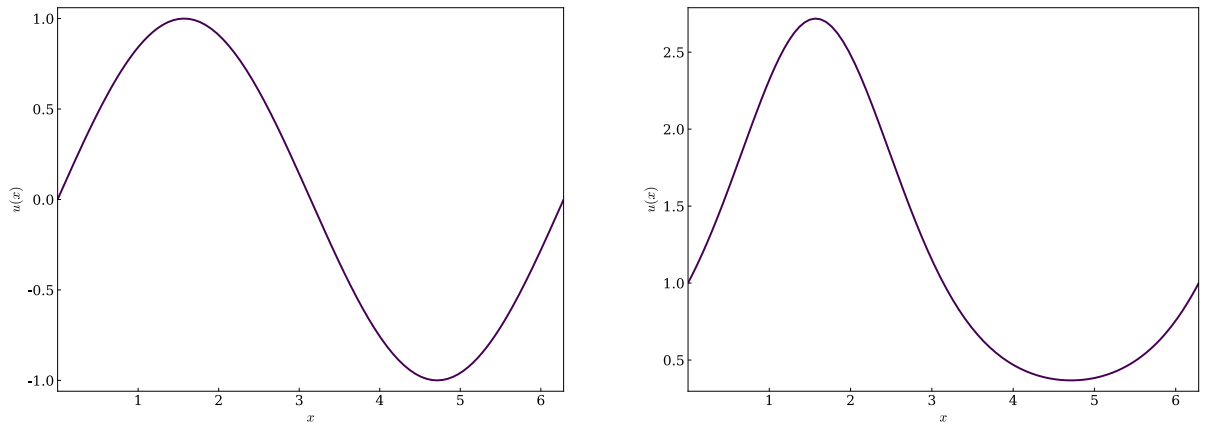
for $1 \leq m \leq r$ and where $x_{\text{in}} = 0$ and $x_{\text{out}} = 2\pi$. The initial condition for this system of equations is specified from the initial condition $u(0, x)$ by (F.5).

Appendix G

SUPPLEMENTAL MACHINE-LEARNING-BASED SPECTRAL METHODS FOR PARTIAL DIFFERENTIAL EQUATIONS RESULTS

This appendix contains additional results for the one-dimensional advection, advection-diffusion, viscous Burgers, Korteweg–de Vries, Kuramoto–Sivashinsky, and inviscid Burgers equations. Additional results for the linear PDEs are presented in Section G.1 for the advection and Section G.2 for the advection-diffusion equation. Additional results for the nonlinear PDEs are presented in Section G.3 for the viscous Burgers, Section G.4 for the Korteweg–de Vries, Section G.5 for the Kuramoto–Sivashinsky, and Section G.6 for the inviscid Burgers equation.

The two out-of-distribution initial conditions tested for each of the example PDEs are shown in Figure G.1.



(a) $u(0, x) = \sin(x)$ (b) $u(0, x) = e^{\sin(x)}$

Figure G.1: Out of distribution test initial conditions, $u(0, x) = \sin(x)$, and $u(0, x) = e^{\sin(x)}$, for each PDE.

G.1 1D Advection equation

The in-distribution initial condition is shown in Figure G.2. Spatiotemporal plots for the initial conditions $u_0(x) = \sin(x)$ and $u_0(x) = e^{\sin(x)}$ are shown in Figures G.3 and G.4, respectively. Spatiotemporal plots for the two out-of-distribution initial conditions and for the revised parameter value $\alpha = -4.0$ are shown in Figures G.5 and G.6.

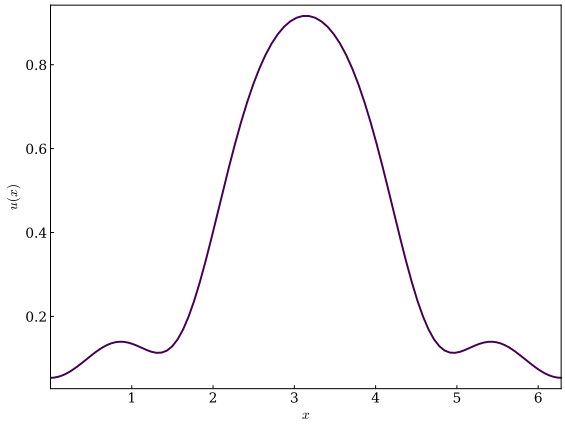
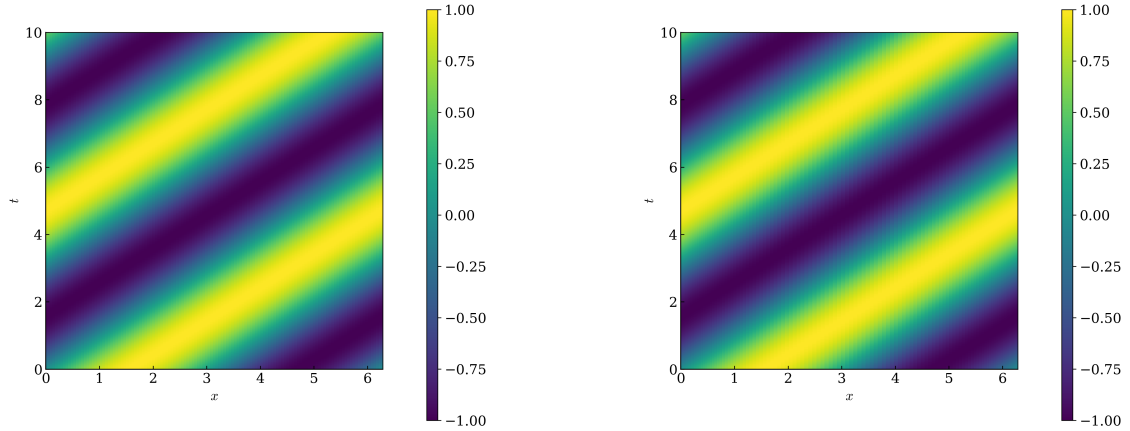


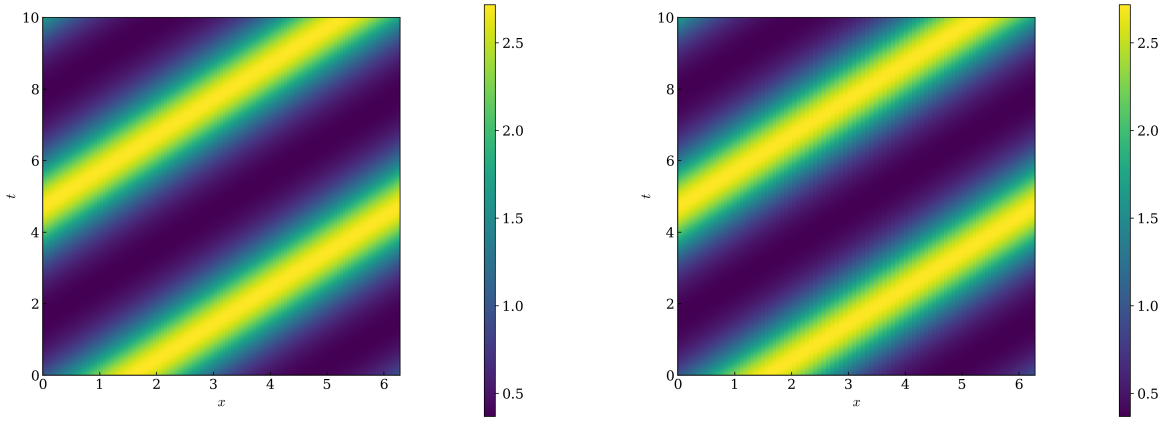
Figure G.2: Random in-distribution test initial condition for the advection equation.



(a) Fourier solution

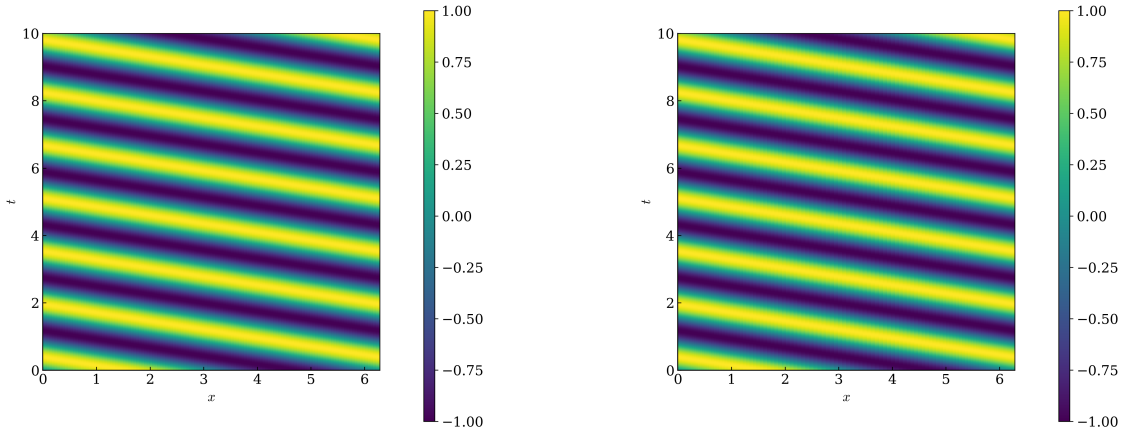
(b) Custom-made basis solution

Figure G.3: Spatiotemporal evolution of the advection equation for the temporal interval $t \in [0, 10]$ and the out-of-distribution initial condition $u_0(x) = \sin(x)$.



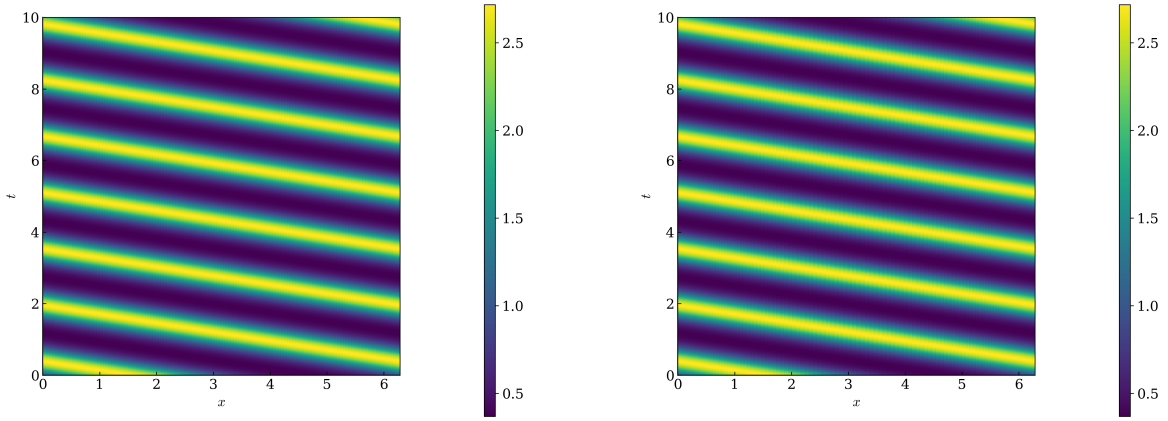
(a) Fourier solution (b) Custom-made basis solution

Figure G.4: Spatiotemporal evolution of the advection equation for the temporal interval $t \in [0, 10]$ and the out-of-distribution initial condition $u_0(x) = e^{\sin(x)}$.



(a) Fourier solution (b) Custom-made basis solution

Figure G.5: Spatiotemporal evolution of the advection equation for the temporal interval $t \in [0, 10]$, the out-of-distribution initial condition $u_0(x) = \sin(x)$, and with $\alpha = -4.0$.



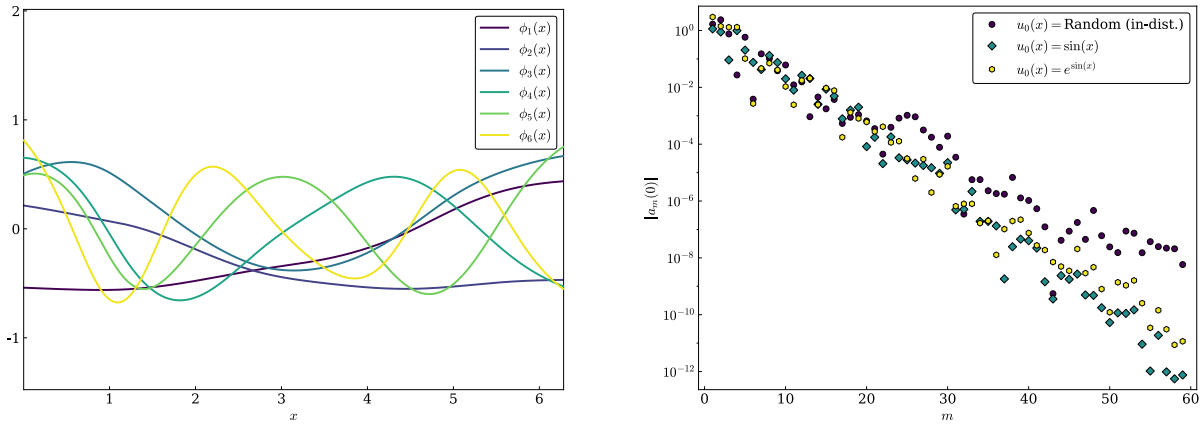
(a) Fourier solution

(b) Custom-made basis solution

Figure G.6: Spatiotemporal evolution of the advection equation for the temporal interval $t \in [0, 10]$, the out-of-distribution initial condition $u_0(x) = e^{\sin(x)}$, and with $\alpha = -4.0$.

G.2 1D Advection-diffusion equation

Plots of the custom-made basis functions and the expansion coefficients are shown in Figure G.7. The in-distribution initial condition is shown in Figure G.8. Spatiotemporal plots for the initial conditions $u_0(x) = \sin(x)$ and $u_0(x) = e^{\sin(x)}$ are shown in Figures G.9 and G.10, respectively. Spatiotemporal plots for the two out-of-distribution initial conditions and for the revised parameter values $\alpha = -4.0$, $\nu = 0.01$ are shown in Figures G.11 and G.12.



(a) Custom-made basis functions

(b) Initial condition expansion coefficients $|a_m(0)|$

Figure G.7: Custom-made basis functions (a) and corresponding initial condition expansion coefficients (b) for the advection-diffusion equation when using $r = 59$ custom-made basis functions.

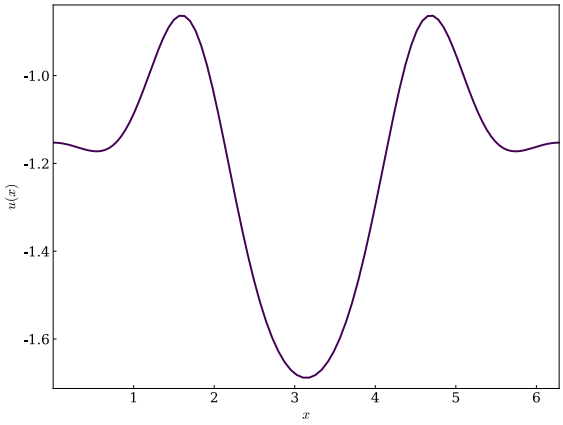
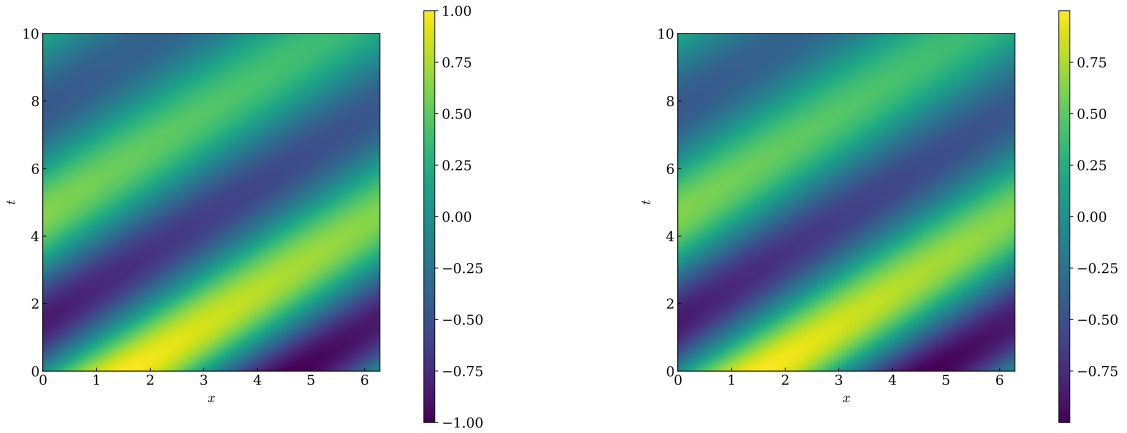
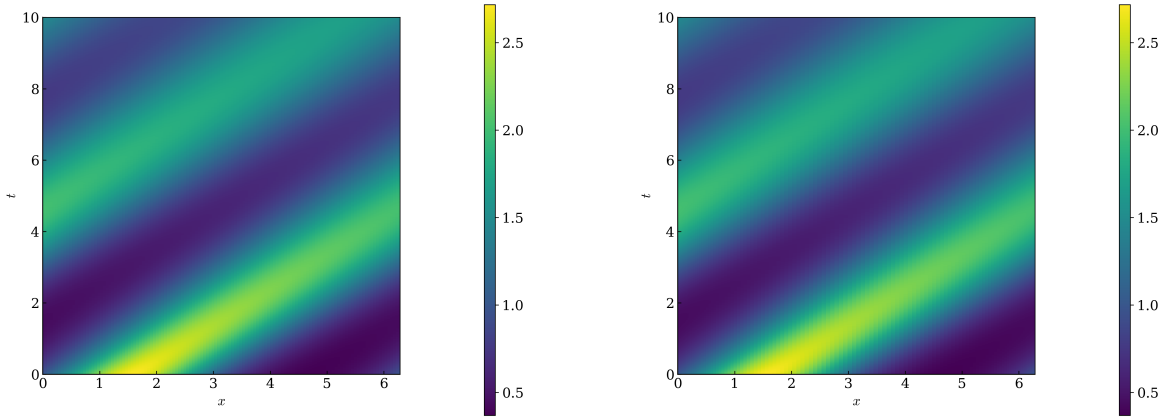


Figure G.8: Random in-distribution test initial condition for the advection-diffusion equation.



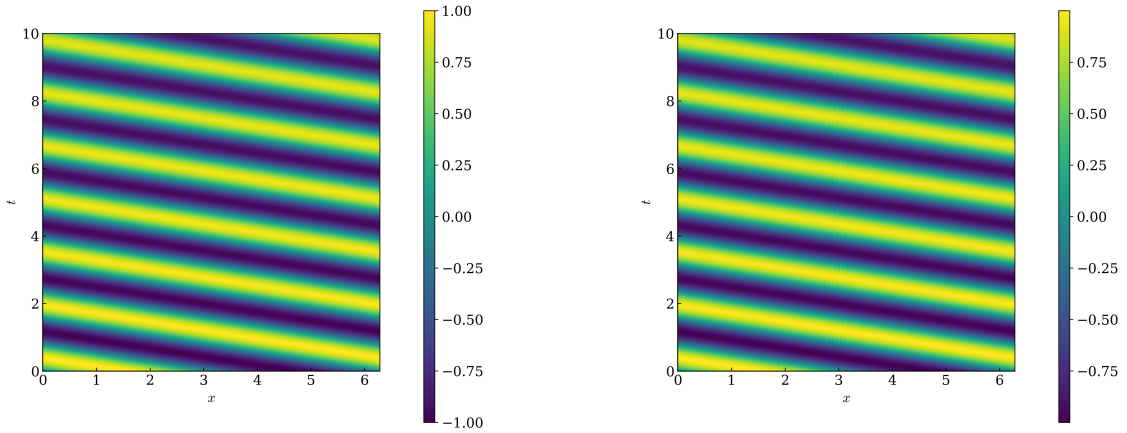
(a) Fourier solution (b) Custom-made basis solution

Figure G.9: Spatiotemporal evolution of the advection-diffusion equation for the temporal interval $t \in [0, 10]$ and the out-of-distribution initial condition $u_0(x) = \sin(x)$.



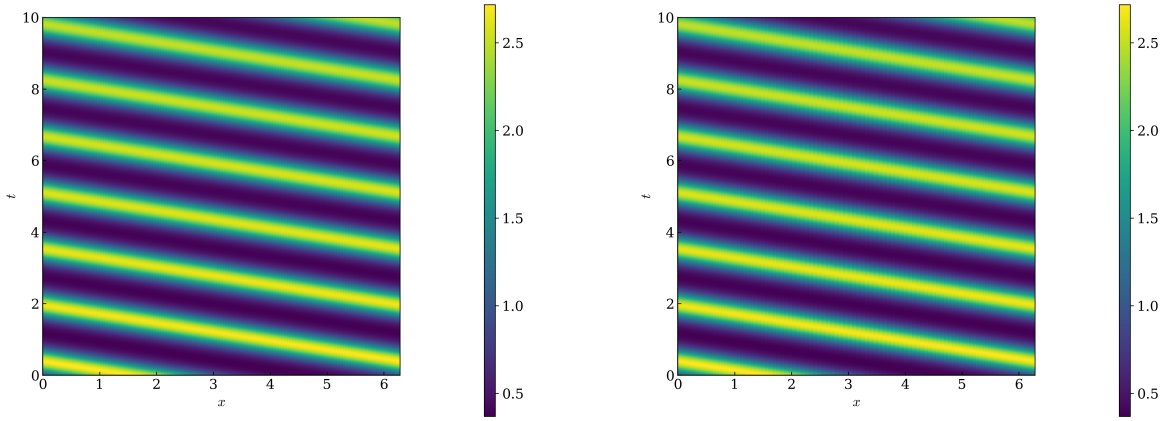
(a) Fourier solution (b) Custom-made basis solution

Figure G.10: Spatiotemporal evolution of the advection-diffusion equation for the temporal interval $t \in [0, 10]$ and the out-of-distribution initial condition $u_0(x) = e^{\sin(x)}$.



(a) Fourier solution (b) Custom-made basis solution

Figure G.11: Spatiotemporal evolution of the advection-diffusion equation for the temporal interval $t \in [0, 10]$, the out-of-distribution initial condition $u_0(x) = \sin(x)$, and with $\alpha = -4.0$, $\nu = 0.01$.

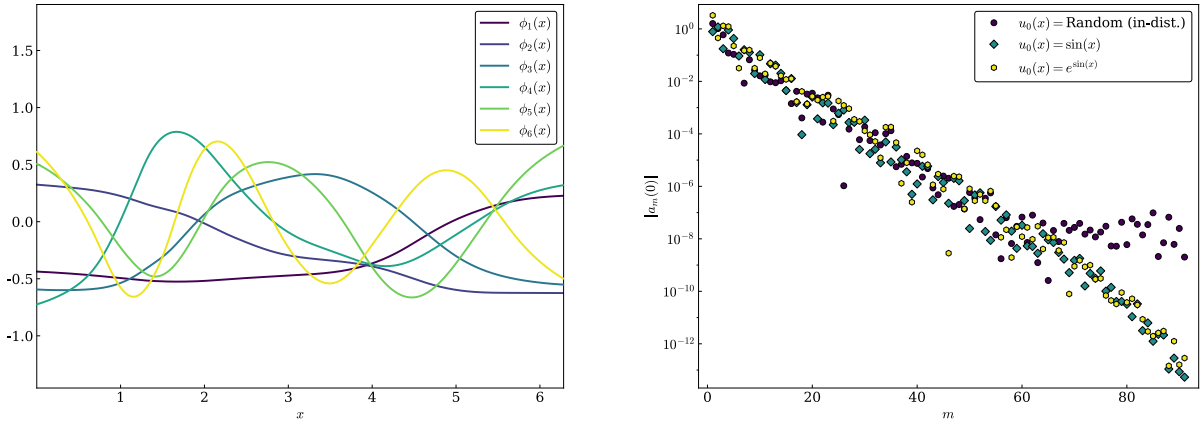


(a) Fourier solution (b) Custom-made basis solution

Figure G.12: Spatiotemporal evolution of the advection-diffusion equation for the temporal interval $t \in [0, 10]$, the out-of-distribution initial condition $u_0(x) = e^{\sin(x)}$, and with $\alpha = -4.0$, $\nu = 0.01$.

G.3 1D Viscous Burgers equation

Plots of the custom-made basis functions and the expansion coefficients are shown in Figure G.13. The in-distribution initial condition is shown in Figure G.14. Spatiotemporal plots for the initial conditions $u_0(x) = \sin(x)$ and $u_0(x) = e^{\sin(x)}$ are shown in Figures G.15 and G.16, respectively. Spatiotemporal plots for the two out-of-distribution initial conditions and for the revised parameter value $\nu = 0.01$ are shown in Figures G.17 and G.18.



(a) Custom-made basis functions

(b) Initial condition expansion coefficients $|a_m(0)|$

Figure G.13: Custom-made basis functions (a) and corresponding initial condition expansion coefficients (b) for the viscous Burgers equation when using $r = 91$ custom-made basis functions.

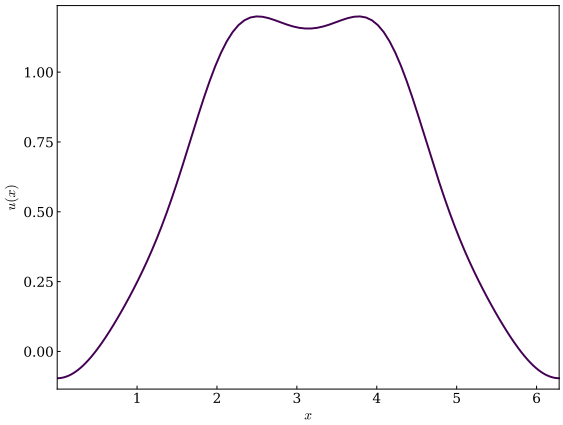
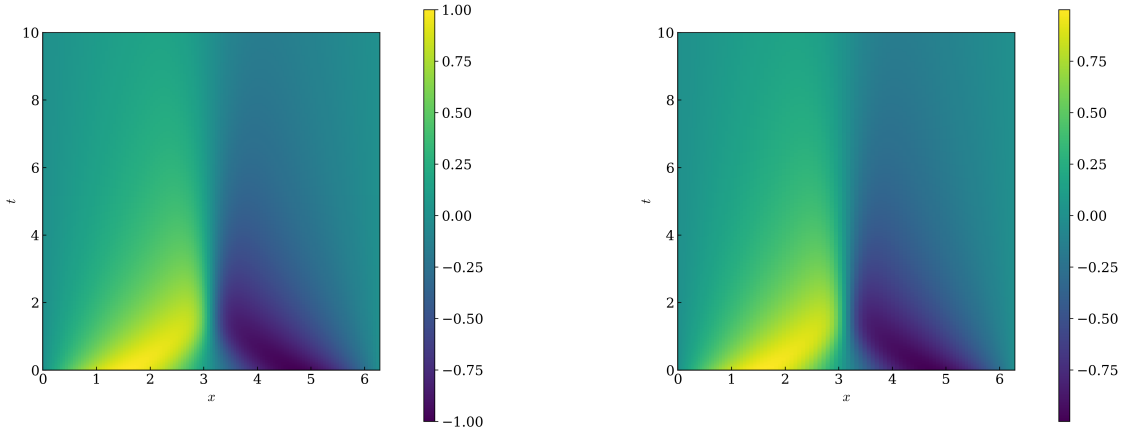


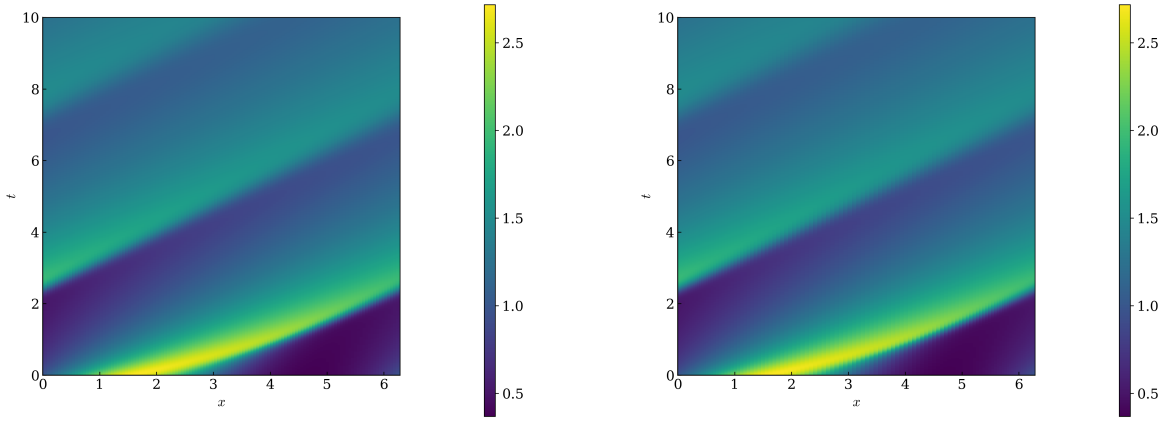
Figure G.14: Random in-distribution test initial condition for the viscous Burgers equation.



(a) Fourier solution

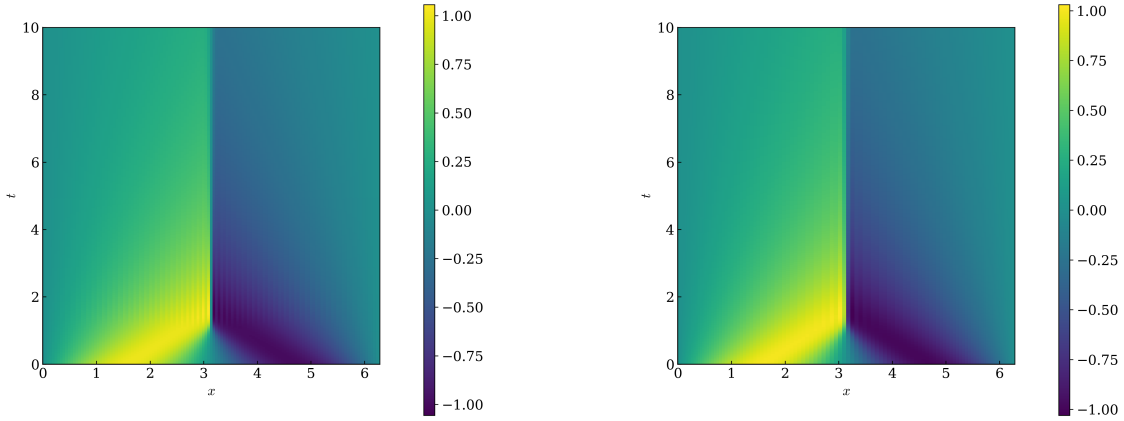
(b) Custom-made basis solution

Figure G.15: Spatiotemporal evolution of the viscous Burgers equation for the temporal interval $t \in [0, 10]$ and the out-of-distribution initial condition $u_0(x) = \sin(x)$.



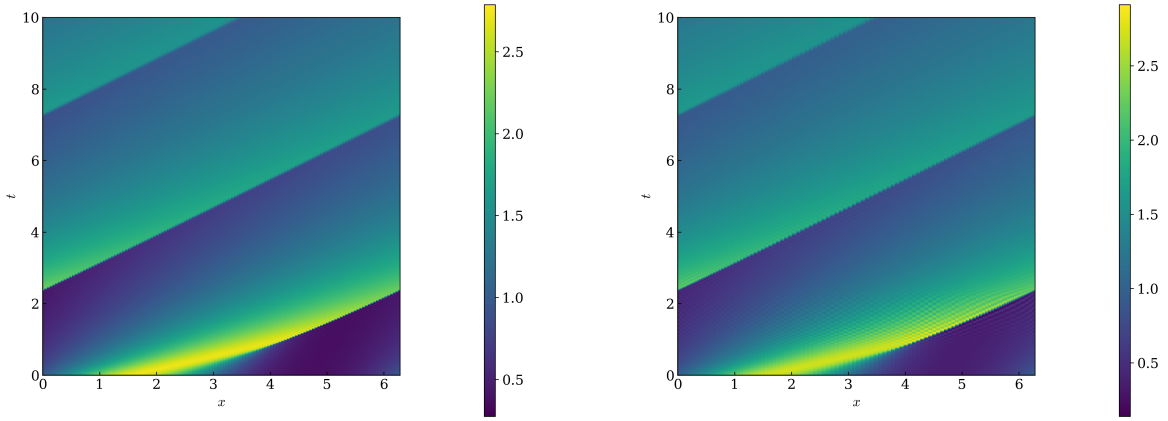
(a) Fourier solution (b) Custom-made basis solution

Figure G.16: Spatiotemporal evolution of the viscous Burgers equation for the temporal interval $t \in [0, 10]$ and the out-of-distribution initial condition $u_0(x) = e^{\sin(x)}$.



(a) Fourier solution (b) Custom-made basis solution

Figure G.17: Spatiotemporal evolution of the viscous Burgers equation for the temporal interval $t \in [0, 10]$, the out-of-distribution initial condition $u_0(x) = \sin(x)$, and with $\nu = 0.01$.



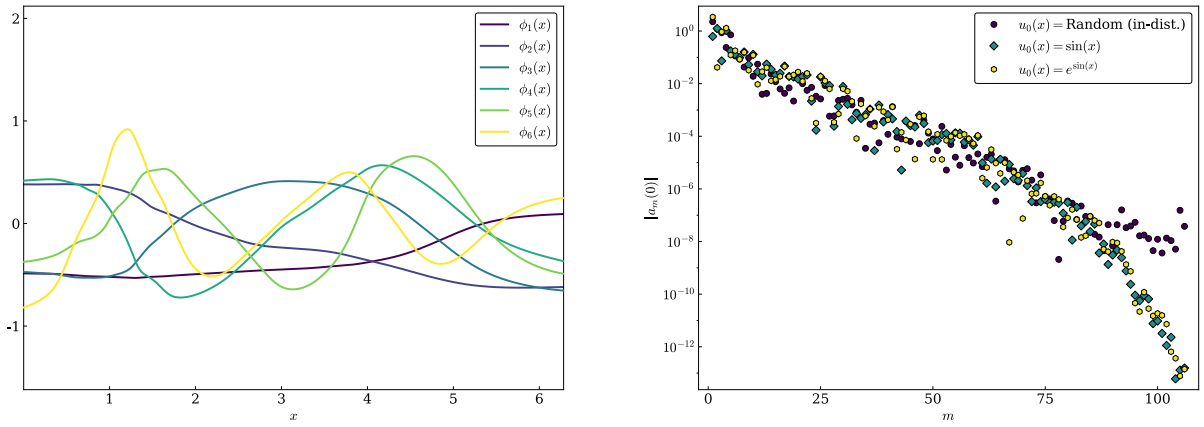
(a) Fourier solution

(b) Custom-made basis solution

Figure G.18: Spatiotemporal evolution of the viscous Burgers equation for the temporal interval $t \in [0, 10]$, the out-of-distribution initial condition $u_0(x) = e^{\sin(x)}$, and with $\nu = 0.01$.

G.4 1D Korteweg–de Vries equation

Plots of the custom-made basis functions and the expansion coefficients are shown in Figure G.19. The in-distribution initial condition is shown in Figure G.20. Spatiotemporal plots for the initial conditions $u_0(x) = \sin(x)$ and $u_0(x) = e^{\sin(x)}$ are shown in Figures G.21 and G.22, respectively. Spatiotemporal plots for the two out-of-distribution initial conditions and for the revised parameter value $\delta = \sqrt{0.005}$ are shown in Figures G.23 and G.24.



(a) Custom-made basis functions

(b) Initial condition expansion coefficients $|a_m(0)|$

Figure G.19: Custom-made basis functions (a) and corresponding initial condition expansion coefficients (b) for the Korteweg–de Vries equation when using $r = 106$ custom-made basis functions.

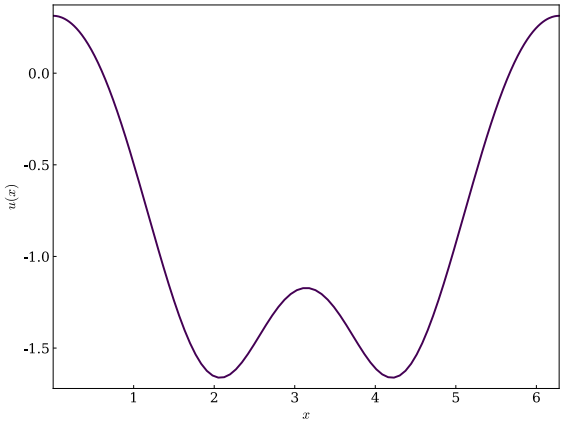
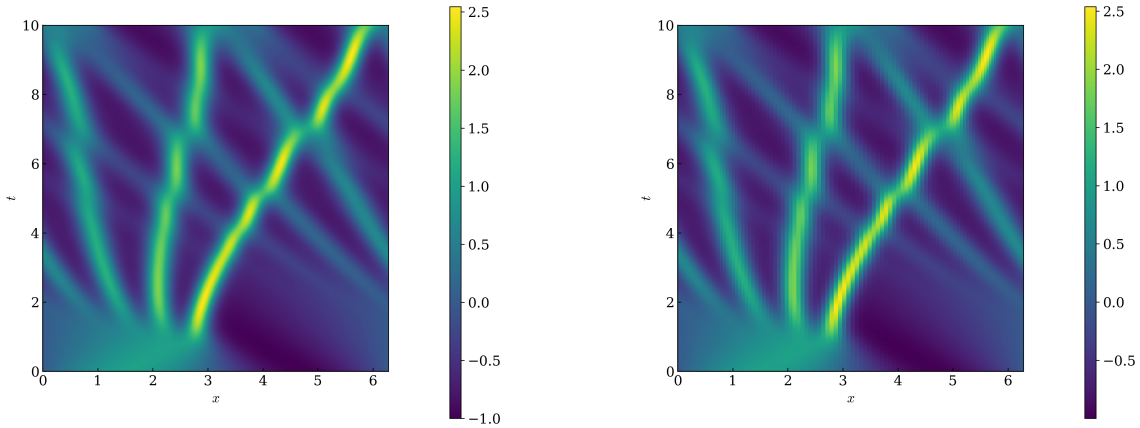
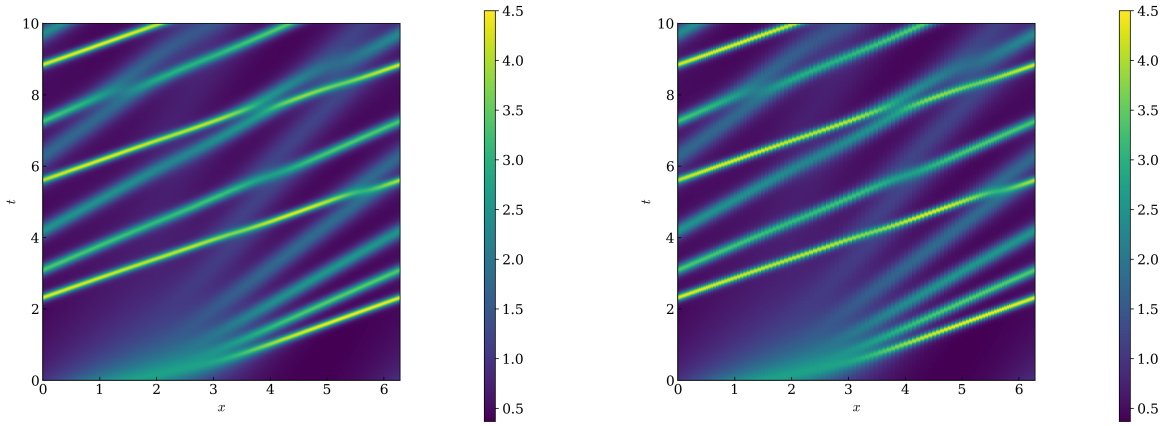


Figure G.20: Random in-distribution test initial condition for the Korteweg–de Vries equation.



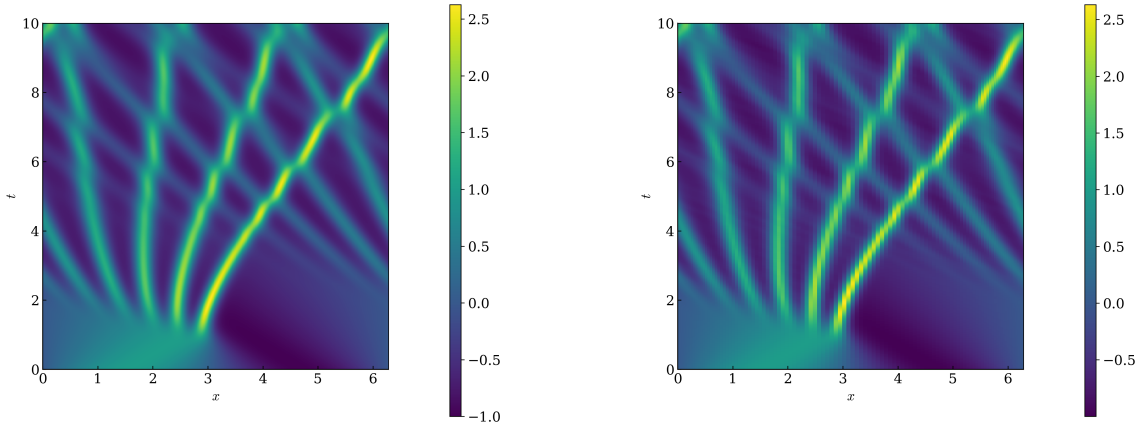
(a) Fourier solution (b) Custom-made basis solution

Figure G.21: Spatiotemporal evolution of the Korteweg–de Vries equation for the temporal interval $t \in [0, 10]$ and the out-of-distribution initial condition $u_0(x) = \sin(x)$.



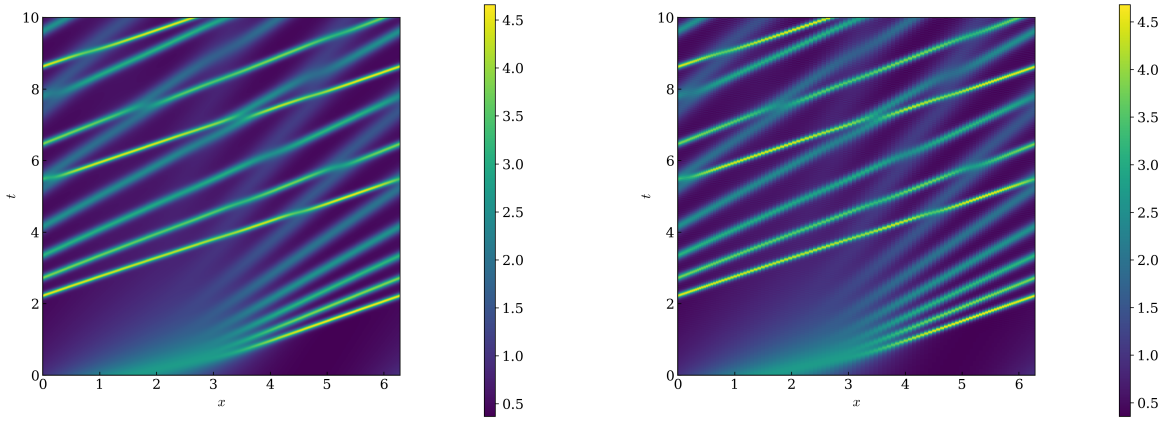
(a) Fourier solution (b) Custom-made basis solution

Figure G.22: Spatiotemporal evolution of the Korteweg–de Vries equation for the temporal interval $t \in [0, 10]$ and the out-of-distribution initial condition $u_0(x) = e^{\sin(x)}$.



(a) Fourier solution (b) Custom-made basis solution

Figure G.23: Spatiotemporal evolution of the Korteweg–de Vries equation for the temporal interval $t \in [0, 10]$, the out-of-distribution initial condition $u_0(x) = \sin(x)$, and with $\delta = \sqrt{0.005}$.

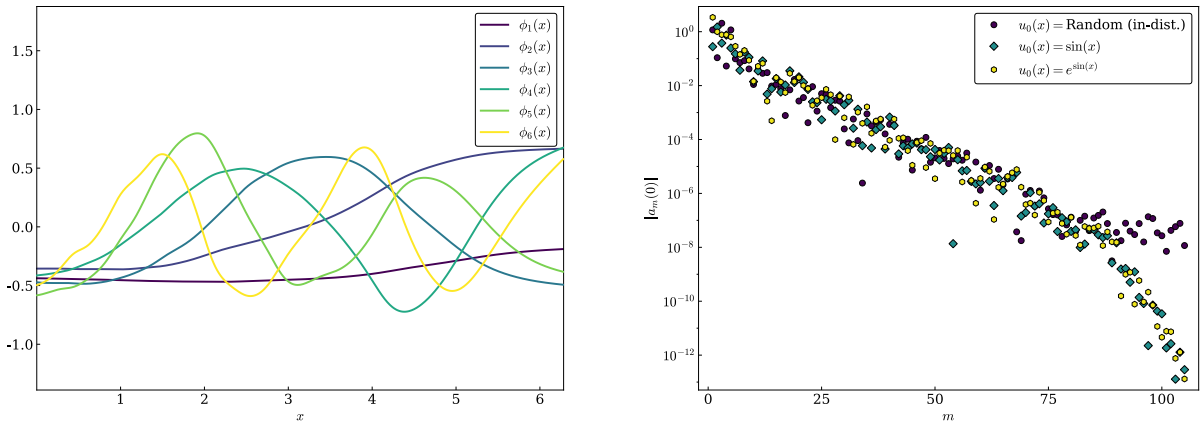


(a) Fourier solution (b) Custom-made basis solution

Figure G.24: Spatiotemporal evolution of the Korteweg–de Vries equation for the temporal interval $t \in [0, 10]$, the out-of-distribution initial condition $u_0(x) = e^{\sin(x)}$, and with $\delta = \sqrt{0.005}$.

G.5 1D Kuramoto–Sivashinsky equation

Plots of the custom-made basis functions and the expansion coefficients are shown in Figure G.25. The in-distribution initial condition is shown in Figure G.26. Spatiotemporal plots for the initial conditions $u_0(x) = \sin(x)$ and $u_0(x) = e^{\sin(x)}$ are shown in Figures G.27 and G.28, respectively. Spatiotemporal plots for the two out-of-distribution initial conditions and for the revised parameter value $\beta = 0.05$ are shown in Figures G.29 and G.30.



(a) Custom-made basis functions

(b) Initial condition expansion coefficients $|a_m(0)|$

Figure G.25: Custom-made basis functions (a) and corresponding initial condition expansion coefficients (b) for the Kuramoto–Sivashinsky equation when using $r = 105$ custom-made basis functions.

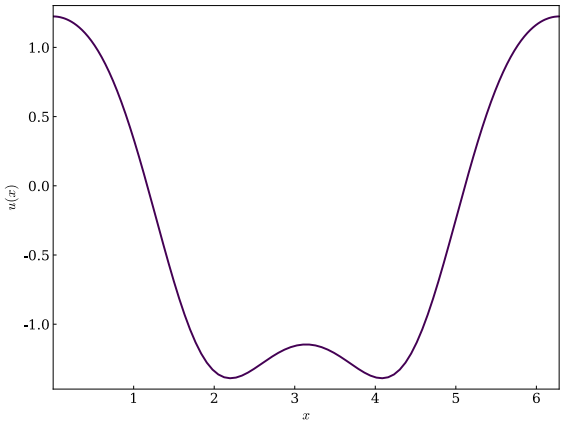
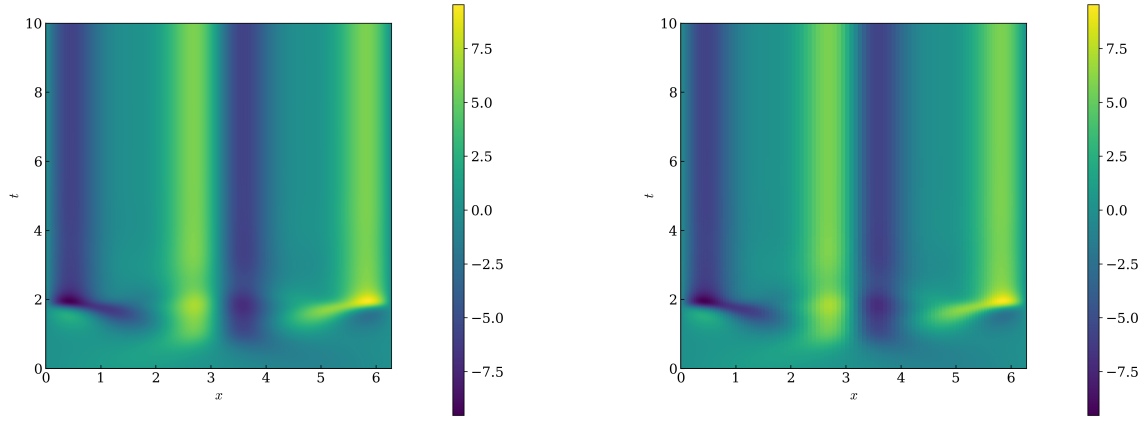


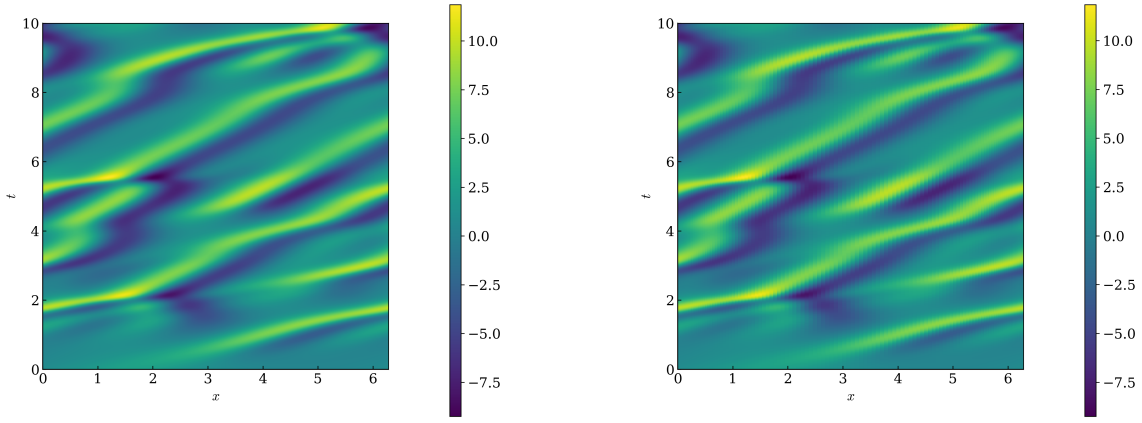
Figure G.26: Random in-distribution test initial condition for the Kuramoto–Sivashinsky equation.



(a) Fourier solution

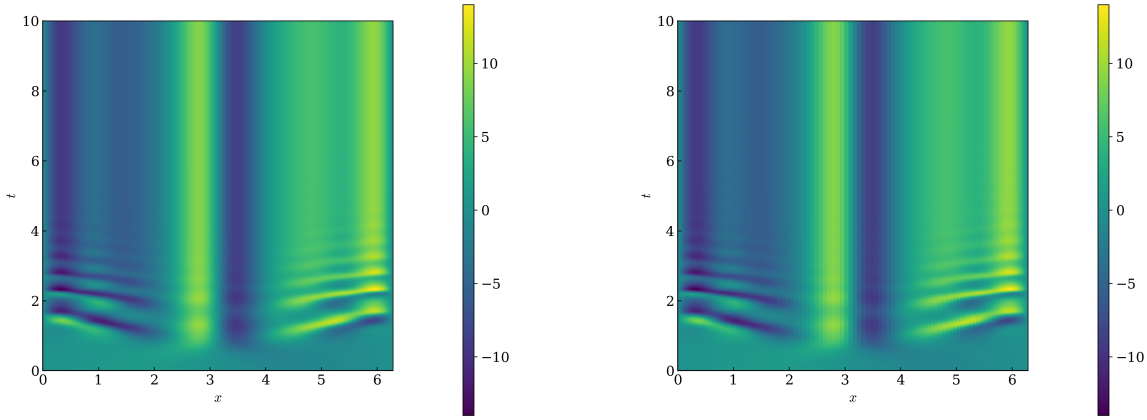
(b) Custom-made basis solution

Figure G.27: Spatiotemporal evolution of the Kuramoto–Sivashinsky equation for the temporal interval $t \in [0, 10]$ and the out-of-distribution initial condition $u_0(x) = \sin(x)$.



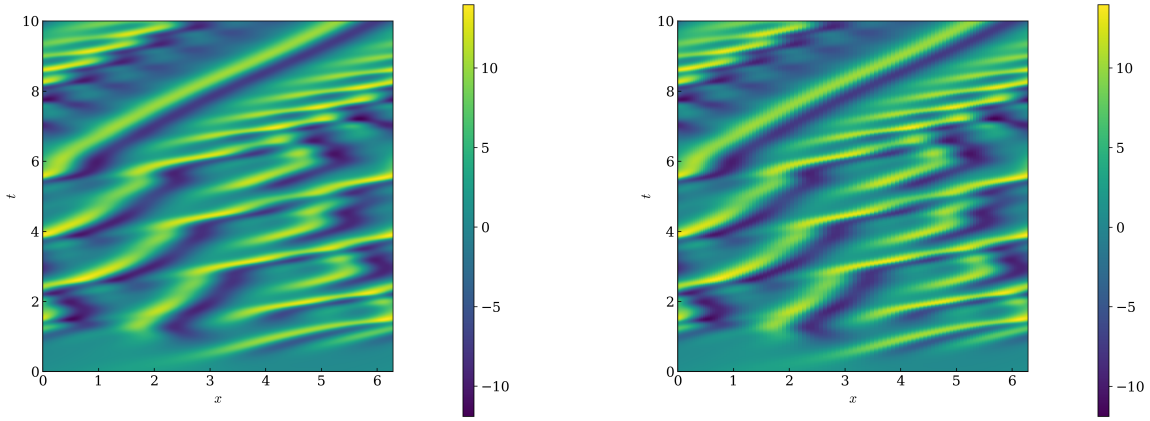
(a) Fourier solution (b) Custom-made basis solution

Figure G.28: Spatiotemporal evolution of the Kuramoto–Sivashinsky equation for the temporal interval $t \in [0, 10]$ and the out-of-distribution initial condition $u_0(x) = e^{\sin(x)}$.



(a) Fourier solution (b) Custom-made basis solution

Figure G.29: Spatiotemporal evolution of the Kuramoto–Sivashinsky equation for the temporal interval $t \in [0, 10]$, the out-of-distribution initial condition $u_0(x) = \sin(x)$, and with $\beta = 0.05$.



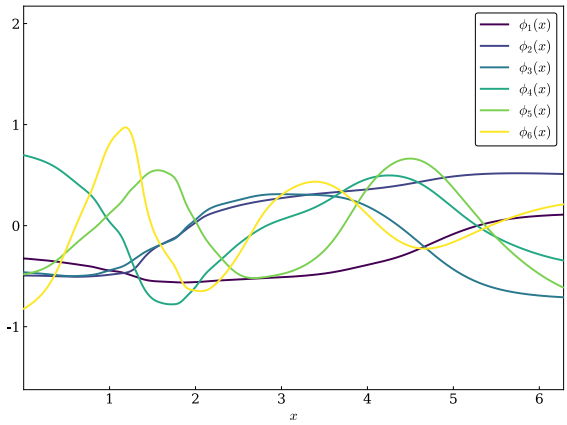
(a) Fourier solution

(b) Custom-made basis solution

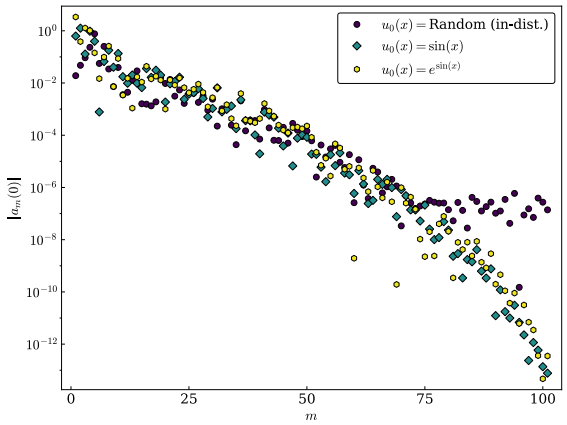
Figure G.30: Spatiotemporal evolution of the Kuramoto–Sivashinsky equation for the temporal interval $t \in [0, 10]$, the out-of-distribution initial condition $u_0(x) = e^{\sin(x)}$, and with $\beta = 0.05$.

G.6 1D Inviscid Burgers equation

Plots of the custom-made basis functions and the expansion coefficients are shown in Figure G.31. The in-distribution initial condition is shown in Figure G.32. Spatiotemporal plots for the initial conditions $u_0(x) = \sin(x)$ and $u_0(x) = e^{\sin(x)}$ are shown in Figures G.33 and G.34, respectively.



(a) Custom-made basis functions



(b) Initial condition expansion coefficients $|a_m(0)|$

Figure G.31: Custom-made basis functions (a) and corresponding initial condition expansion coefficients (b) for the inviscid Burgers equation when using $r = 101$ custom-made basis functions.

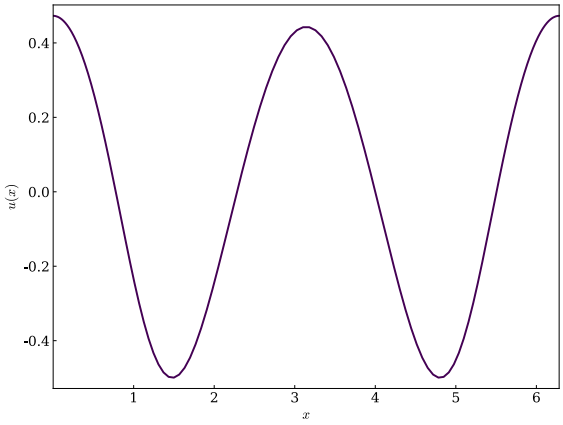
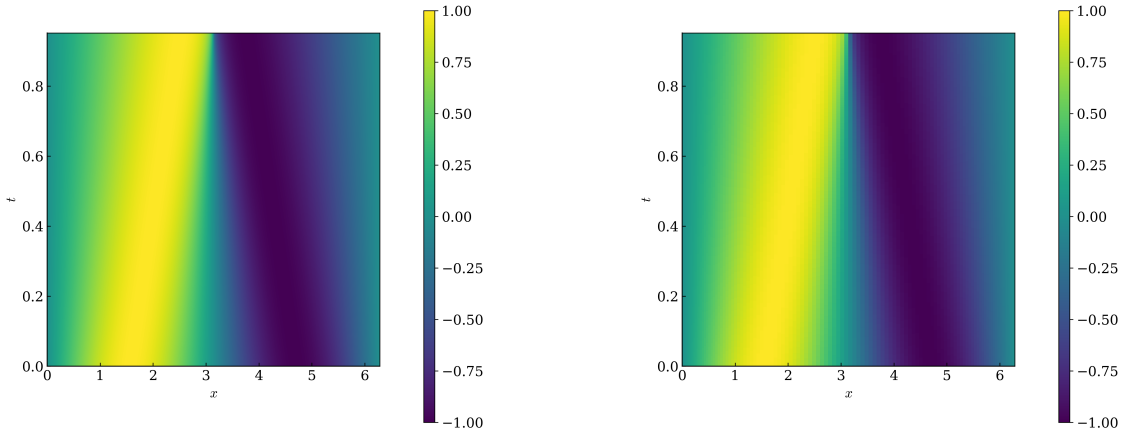
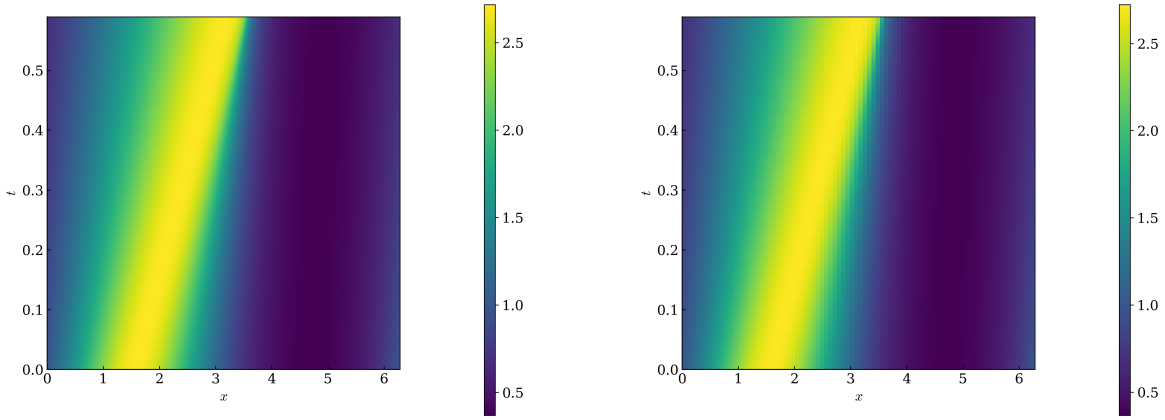


Figure G.32: Random in-distribution test initial condition for the inviscid Burgers equation.



(a) MUSCL solution (b) Custom-made basis solution

Figure G.33: Spatiotemporal evolution of the inviscid Burgers equation for the temporal interval $t \in [0, 0.9515]$ and the out-of-distribution initial condition $u_0(x) = \sin(x)$.



(a) MUSCL solution (b) Custom-made basis solution

Figure G.34: Spatiotemporal evolution of the inviscid Burgers equation for the temporal interval $t \in [0, 0.5892]$ and the out-of-distribution initial condition $u_0(x) = e^{\sin(x)}$.

Appendix H

TIME-SAMPLED CUSTOM-MADE BASIS FUNCTIONS

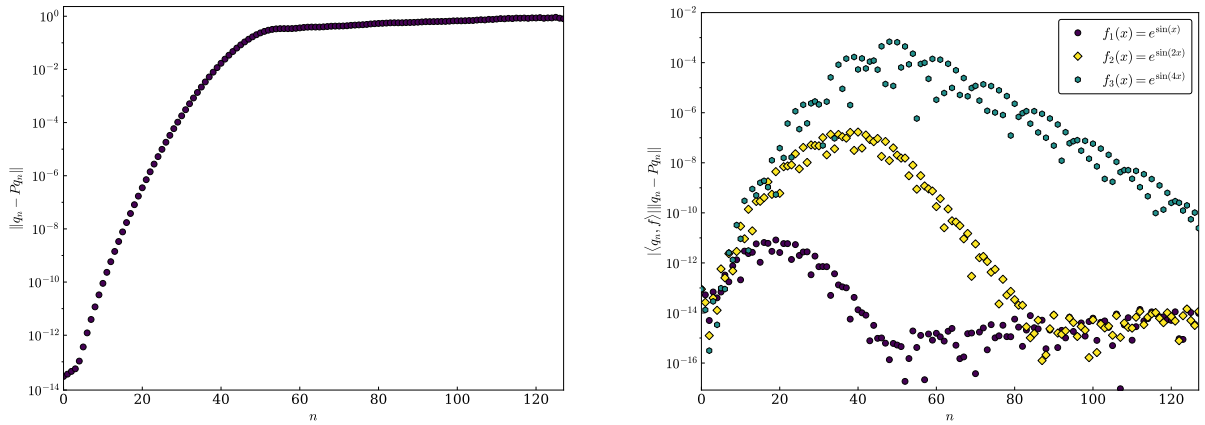
This appendix outlines the concept of time-sampling the custom-made basis functions. The approximation capabilities are evaluated in Section H.1 and numerical results for the advection equation, along with results comparing the average relative errors between the frozen-in-time and time-sampled custom-made basis functions for the advection-diffusion, viscous Burgers, Korteweg–de Vries, Kuramoto–Sivashinsky, and inviscid Burgers equations, are shown in Section H.2.

For all work presented in Chapter 3, the candidate basis functions obtained from DeepONets were evaluated at $t = 0$ as outlined in Section 3.2.1. However, the choice of $t = 0$ was an initial assumption, and there is nothing that limits the evaluation time to $t = 0$. The procedure to develop time-sampled custom-made basis functions proceeds by evaluating the candidate trunk network functions $\{\gamma_k\}_{1\leq k\leq N}$, at equally spaced values in the temporal domain to generate a larger collection of trunk network functions $\{\tau_f\}_{1\leq f\leq w}$, where $w = (1 + 1/\Delta t)N$. The matrix A is then constructed by evaluating each of the time-sampled trunk network functions at the specified quadrature nodes, such that $A_{if} = \tau_f(x_i)$. The SVD is then computed, and the Legendre polynomials are utilized to evaluate the time-sampled custom-made basis functions at locations away from the prescribed quadrature nodes.

H.1 Approximation capabilities

The approximation capabilities of the time-sampled custom-made basis functions can be evaluated by analyzing the errors made in approximating the orthonormal Legendre polynomials, as outlined in Section 3.2.2. For this analysis, the domain $\Omega = [0, 2\pi]$, is utilized, and Δt, is set to 0.05 for the time-sampling procedure. Figure H.1a shows the errors made in

approximating the Legendre polynomials using the time-sampled custom-made basis functions. Similar to what was found for the custom-made basis functions constructed from the frozen-in-time trunk network functions, the time-sampled custom-made basis functions have difficulty approximating the higher frequency functions corresponding to larger n. However, the increase in the errors of approximating functions corresponding to larger n is slower than that found for frozen-in-time custom-made basis functions (Figure 3.4). Figure H.1b shows the errors if the entire second term on the right-hand side of Equation (3.16) is considered for three functions of increasing frequency: $f_1(x) = e^{\sin(x)}$, $f_2(x) = e^{\sin(2x)}$, and $f_3(x) = e^{\sin(4x)}$. For all three functions, the peak errors and the actual errors $\|f - Pf\|$, are reduced when utilizing the time-sampled custom-made basis functions as shown in Figure H.1b and Table H.1. For these results, a trunk network trained for the advection equation $\partial u/\partial t + \partial u/\partial x = 0$, on the periodic domain $x \in [0, 2\pi]$, and for $t \in [0, 1]$, was utilized to generate the time-sampled custom-made basis. Ninety-eight time-sampled custom-made basis functions which satisfy the singular value threshold of 10^{-13} were used for the analysis.



(a) $\|q_n - Pq_n\|$ (b) $|\langle q_n, f \rangle| \|q_n - Pq_n\|$

Figure H.1: Errors in approximating the n-th Legendre polynomial using the time-sampled custom-made basis functions (a). The upper error bound, $|\langle q_n, f \rangle| \|q_n - Pq_n\|$ computed for three different functions with increasing frequency (b).

$u_0(x)$	Frozen at $t = 0$	Time-sampled
$e^{\sin(x)}$	7.3086×10^{-7}	7.5784×10^{-11}
$e^{\sin(2x)}$	1.9574×10^{-3}	1.8858×10^{-6}
$e^{\sin(4x)}$	1.4259×10^{-1}	4.3762×10^{-3}

Table H.1: The errors $\|f - Pf\|$ for three different functions with increasing frequency when using the frozen-in-time ($r = 53$) and the time-sampled ($r = 98$) set of custom-made basis functions.

H.2 Numerical results

In this section, results are presented for the one-dimensional linear advection equation. Additionally, representative results comparing the average relative errors between the frozen-in-time and time-sampled custom-made basis functions for the advection-diffusion, viscous Burgers, Korteweg–de Vries, Kuramoto–Sivashinsky, and inviscid Burgers equations are presented. A singular value threshold of 10^{-13} was specified for all examples, and a 128-node Gauss–Legendre quadrature grid coupled with Gauss–Legendre quadrature was utilized for the calculation of inner products. Differentiation was performed using automatic differentiation, and the same numerical integrators and tolerances as outlined in Section 3.4 were utilized.

Figure H.2 shows the decaying singular value spectrum for all six of the test PDEs. The decay rate is slower than that found for the custom-made basis functions constructed using the frozen-in-time trunk network functions but still indicates a hierarchical structure for the time-sampled custom-made basis functions. Figure H.3 presents the time-sampled custom-made basis functions and the expansion coefficients for the advection equation. Figure H.4 shows the evolution of the relative error for the three test initial conditions when using $r = 98$ and $r = 53$ time-sampled custom-made basis functions to expand the solution of the advection equation. Using $r = 98$ time-sampled custom-made basis functions results in an increase in accuracy for the in-distribution initial condition and the out-of-distribution initial

condition $u_0(x) = e^{\sin(x)}$, but a decrease in accuracy for the out-of-distribution initial condition $u_0(x) = \sin(x)$ compared to the frozen-in-time custom-made basis functions. Using the same number of basis functions as found for the frozen-in-time custom-made basis functions, $r = 53$, results in a slight increase in accuracy for the in-distribution initial condition but an increase in the average relative error for the out-of-distribution initial conditions.

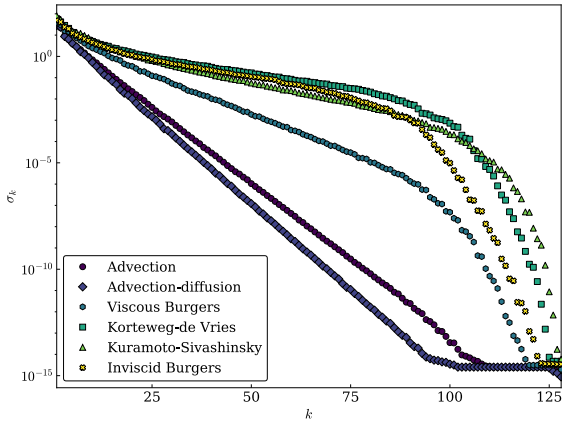
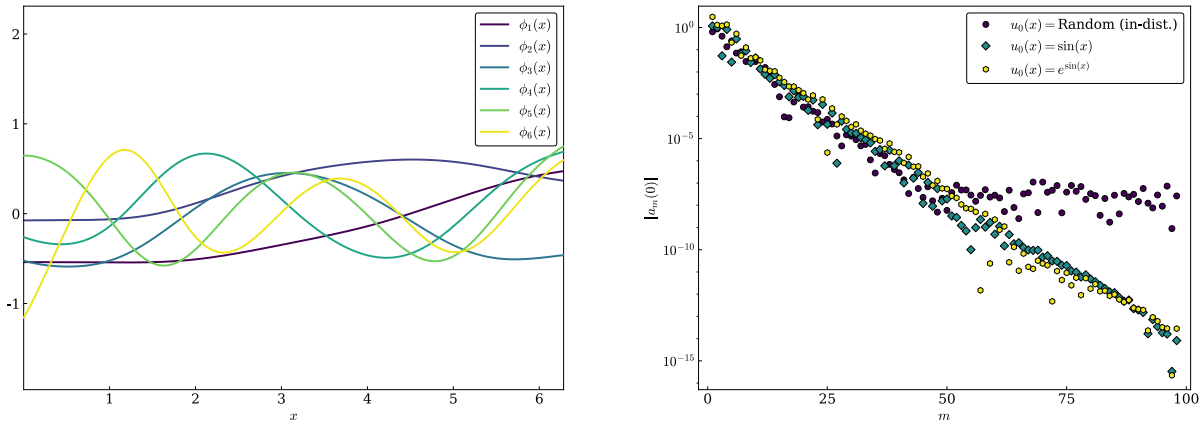


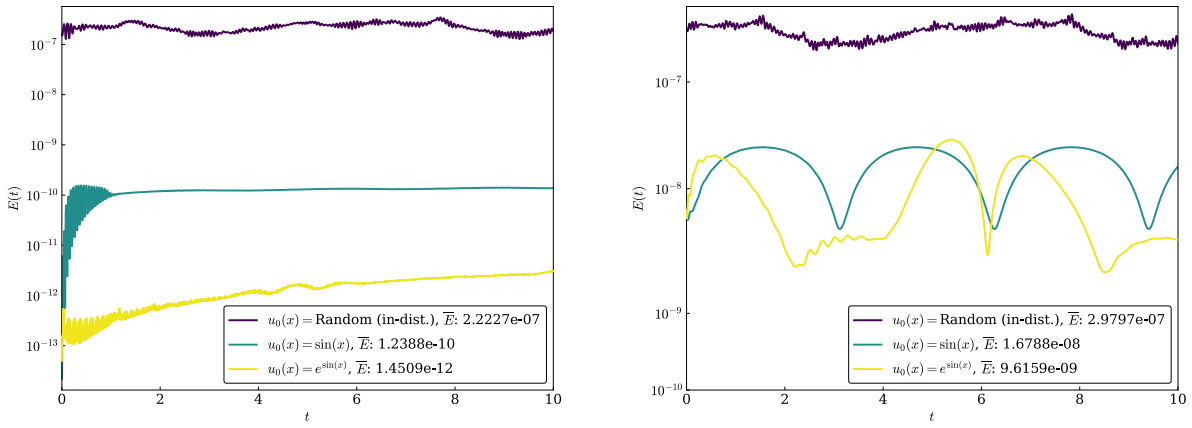
Figure H.2: Singular value spectrum of the time-sampled custom-made basis functions for the advection, advection-diffusion, viscous Burgers, Korteweg–de Vries, Kuramoto–Sivashinsky, and inviscid Burgers equations.



(a) Time-sampled custom-made basis functions

(b) Initial condition expansion coefficients $|a_m(0)|$

Figure H.3: Time-sampled custom-made basis functions (a) and corresponding initial condition expansion coefficients (b) for the advection equation when using $r = 98$ time-sampled custom-made basis functions.



(a) $r = 98$

(b) $r = 53$

Figure H.4: Relative error evolution of the advection equation for the temporal interval $t \in [0, 10]$ and the three test initial conditions when utilizing $r = 98$ (a) and $r = 53$ (b) time-sampled custom-made basis functions.

Representative results comparing the average relative errors between the time-sampled and frozen-in-time custom-made basis functions for the advection-diffusion, viscous Burgers,

Korteweg–de Vries, Kuramoto–Sivashinsky, and inviscid Burgers equations are shown in Tables H.2, H.3, H.4, H.5, and H.6, respectively. For the advection-diffusion and Korteweg–de Vries equations, the use of the time-sampled custom-made basis functions leads to an improvement in the average error for all three test initial conditions when compared to the frozen-in-time custom-made basis functions. The usage of time-sampled custom-made basis functions for the viscous Burgers equation results in an improvement in the average relative error for the out-of-distribution initial condition $u_0(x) = e^{\sin(x)}$, but a decrease in accuracy for the in-distribution initial condition and the out-of-distribution initial condition $u_0(x) = \sin(x)$. For the Kuramoto–Sivashinsky equation, the use of time-sampled custom-made basis functions leads to an increase in the average relative error for all three initial conditions. The use of time-sampled custom-made basis functions for the inviscid Burgers equation leads to a small increase in the average relative error for the in-distribution initial condition. Usage of the time-sampled custom-made basis functions results in a slight decrease in the average relative error for the two out-of-distribution initial conditions. To facilitate a fair comparison for the inviscid Burgers equation, the solution end times for the time-sampled custom-made basis functions were specified to match those found for the frozen-in-time custom-made basis functions.

$u_0(x)$	Frozen at $t = 0$	Time-sampled
Random	7.7971×10^{-11}	7.0628×10^{-11}
$\sin(x)$	7.8798×10^{-13}	2.2073×10^{-13}
$e^{\sin(x)}$	8.6461×10^{-13}	1.4576×10^{-13}

Table H.2: Average relative errors for the advection-diffusion equation using the frozen-in-time ($r = 59$) and the time-sampled ($r = 87$) set of custom-made basis functions. Average error is computed for $t \in [0, 10]$.

$u_0(x)$	Frozen at $t = 0$	Time-sampled
Random	1.4225×10^{-8}	1.8992×10^{-6}
$\sin(x)$	9.7880×10^{-8}	1.0013×10^{-7}
$e^{\sin(x)}$	1.4337×10^{-6}	1.4078×10^{-8}

Table H.3: Average relative errors for the viscous Burgers equation using the frozen-in-time ($r = 91$) and the time-sampled ($r = 115$) set of custom-made basis functions. Average error is computed for $t \in [0, 10]$.

$u_0(x)$	Frozen at $t = 0$	Time-sampled
Random	1.8670×10^{-5}	4.2074×10^{-6}
$\sin(x)$	8.1135×10^{-5}	6.8911×10^{-5}
$e^{\sin(x)}$	1.5309×10^{-4}	1.3923×10^{-4}

Table H.4: Average relative errors for the Korteweg–de Vries equation using the frozen-in-time ($r = 106$) and the time-sampled ($r = 124$) set of custom-made basis functions. Average error is computed for $t \in [0, 10]$.

$u_0(x)$	Frozen at $t = 0$	Time-sampled
Random	7.6522×10^{-4}	8.2453×10^{-4}
$\sin(x)$	6.1557×10^{-5}	2.1650×10^{-3}
$e^{\sin(x)}$	4.9445×10^{-4}	5.8131×10^{-4}

Table H.5: Average relative errors for the Kuramoto–Sivashinsky equation using the frozen-in-time ($r = 105$) and the time-sampled ($r = 126$) set of custom-made basis functions. Average error is computed for $t \in [0, 10]$.

$u_0(x)$	Frozen at $t = 0$	Time-sampled
Random	8.9672×10^{-4}	8.9848×10^{-4}
$\sin(x)$	5.5510×10^{-4}	5.2832×10^{-4}
$e^{\sin(x)}$	1.0534×10^{-4}	9.8833×10^{-5}

Table H.6: Average relative errors for the inviscid Burgers equation using the frozen-in-time ($r = 101$) and the time-sampled ($r = 120$) set of custom-made basis functions. The evolution end time for the time-sampled custom-made basis functions was specified to match that found for the frozen-in-time custom-made basis functions.

Appendix I

ENFORCING BOUNDARY CONDITIONS DURING TRAINING THROUGH A FEATURE EXPANSION

This appendix contains results for the one-dimensional advection, advection-diffusion, and viscous Burgers equations when the custom-made basis functions are forced to individually satisfy the boundary conditions so that a Galerkin, as opposed to a discontinuous Galerkin procedure, can be utilized. To force the custom-made basis functions to individually satisfy the boundary conditions, a feature expansion [47] utilizing two Fourier basis functions ($x \mapsto \{\sin(x), \cos(x)\}$), was applied to the spatial component of the trunk network input. A feature expansion consisting of two Fourier basis functions was selected so that periodic boundary conditions could be enforced, while alternative types of boundary conditions may be enforced through the use of hard constraints. Section I.1 presents the method for constructing a periodic custom-made basis from a DeepONet utilizing a feature expansion, while Section I.2 provides results for the linear advection (Section I.2.1), advection-diffusion (Section I.2.2) equations and the nonlinear viscous Burgers equation (Section I.2.3).

I.1 Construction of periodic custom-made basis functions

The construction of the periodic custom-made basis functions begins with the frozen-in-time trunk network functions and follows a similar approach as outlined in Section 3.2.1, but with the trapezoid rule ($\{(x_j, w_j)\}_{0\leq j\leq M-1}$) utilized to obtain the values of $\{\phi_l\}_{1\leq l\leq p}$ at the quadrature nodes. To evaluate $\{\phi_l\}$ at locations away from the quadrature grid, expansions based on the real trigonometric polynomials and B-Splines were considered. Using the real trigonometric polynomials, $\{\phi_l\}$ can be evaluated at points away from the quadrature grid

through

$$\tilde{\phi}_l = \frac{1}{2\pi}\sum_{j=0}^{M-1}\phi_l(x_j)w_j + \sum_{k=1}^{M/2-1}\left[\left(\frac{1}{\pi}\sum_{j=0}^{M-1}\phi_l(x_j)\cos(kx_j)w_j\right)\cos(kx) + \left(\frac{1}{\pi}\sum_{j=0}^{M-1}\phi_l(x_j)\sin(kx_j)w_j\right)\sin(kx)\right], \quad \text{for } 1 \le l \le p, \tag{I.1}$$

where p is the number of trunk network functions evaluated at $t = 0$. The expansion based on trigonometric polynomials coupled with a singular value threshold of 10^{-10} is sufficient when considering linear PDEs or when utilizing a small number of custom-made basis functions; however, nonlinear PDEs, or the case of r large, requires a larger number of quadrature points than specified for the standard construction outlined in Section 3.2.1.

An expansion based on an n-degree B-Spline basis [27] eliminates the need for a larger number of quadrature points and accommodates the usage of the singular threshold 10^{-13}. To construct an expansion based on B-Splines for the spatial domain $\Omega = [0, 2\pi]$, and periodic boundary conditions, the spatial domain was discretized according to the trapezoid rule (with the addition of the endpoint 2π) and padded to include $6n$ additional quadrature nodes on both ends of the interval $[0, 2\pi]$. From the padded set of quadrature nodes defining the interval $[a, b]$, a knot-sequence of length $k + 1$ is given by $\mathbf{t} = [a = t_0 \le \cdots \le t_k = b]$ where the endpoints appear $n + 1$ times and determine $t_0, \ldots, t_n$ and $t_{k-n}, \ldots, t_k$. From the defined knot sequence, the set of B-Spline functions can be obtained recursively,

$$N_i^0(x) = \begin{cases} 1, & t_i \le x < t_{i+1} \\ 0, & \text{otherwise} \end{cases} \tag{I.2a}$$

$$N_i^h(x) = \frac{(x - t_i)N_i^{h-1}(x)}{t_{i+h} - t_i} + \frac{(t_{i+h+1} - x)N_{i+1}^{h-1}(x)}{t_{i+h+1} - t_{i+1}}, \tag{I.2b}$$

where $h = 1, \ldots, n$ and for $i = 0, \ldots, k - n - 1$. The B-Spline expansion is then written as

$$\tilde{\phi}_l = \sum_{i=0}^{k-n-1} c_i N_i^n(x), \tag{I.3}$$

where the expansion is valid for Ω and has $\mathcal{C}^{n-1}$ continuity at the knots. To determine the expansion coefficients (c_i's), the linear system

$$\sum_{i=0}^{k-n-1} c_i N_i^n(x_g) = \phi_l(x_g), \quad \text{for } 0 \leq g \leq k - n - 1, \tag{I.4}$$

is solved, where x_g are the quadrature nodes of the padded domain and the periodicity of the custom-made basis vectors is utilized to obtain $\phi_l(x_g)$.

I.2 Numerical results

In this section, results are presented for three one-dimensional PDEs: the advection, advection-diffusion, and viscous Burgers equations. Similar to the results presented for the discontinuous Galerkin approach, the initial conditions were sampled from a mean zero Gaussian random field, and the ground truth data for the advection, advection-diffusion, and viscous Burgers equations was generated by writing the solution in terms of a 128-mode Fourier expansion. The resulting systems of differential equations were solved for $t \in [0, 1]$ using a Runge–Kutta–Dormand–Prince integrator with adaptive step size, relative error tolerance 10^{-10}, and absolute tolerance 10^{-14} [63]. The solutions were saved at time values of 10^{-3} for the linear PDEs and 10^{-4} apart for the nonlinear PDE. Refer to Table E.1 in Appendix E for the parameter settings used to train the DeepONets. To indicate the effects of random initialization of the neural networks, the mean testing error and corresponding standard deviation based on three training runs are presented in Table I.1 for each PDE.

The periodic custom-made basis functions were constructed as outlined in Section I.1 using B-splines with $n = 17$. The r number of basis functions used to solve the PDE was specified to be the set of functions with a corresponding singular value larger than 10^{-13}.

PDE	Mean	Standard deviation
Advection	3.36×10^{-5}	3.36×10^{-6}
Advection-diffusion	6.43×10^{-5}	1.24×10^{-5}
Viscous Burgers	2.04×10^{-3}	6.92×10^{-5}

Table I.1: DeepONet mean testing errors and standard deviation when utilizing a feature expansion for the advection, advection-diffusion, and viscous Burgers equations based on three training runs each.

All examples were solved on a 128-node equally spaced grid, the trapezoid rule was utilized for the calculation of inner products, and differentiation of the custom-made basis functions was performed using automatic differentiation. Additionally, the quadratic nonlinear terms were computed in modal space, with all necessary triple product integrals computed in advance. For all PDEs, the same adaptive step size numerical integrator and tolerances used to generate the ground truth data for training the DeepONets were utilized to evolve the periodic custom-made basis function solutions. The errors presented were calculated using Equation (3.22) and Equation (3.23).

Figure I.1 shows the decaying singular value spectrums for the three test PDEs. The decay rate is slower than that found for the non-periodic custom-made basis functions but still highlights the hierarchical structure of the periodic custom-made basis functions. Similar to the non-periodic custom-made basis functions presented in Section 3.4, the variation in the decay rates between each of the spectrums indicates that the trunk net space of functions is linked to the complexity of the dynamics that each of the PDEs exhibit.

I.2.1 Application 1: 1D Advection equation

Consider again the one-dimensional linear advection equation given by Equation (3.24), with $\alpha = 1.0$. To obtain a system of ODEs in terms of the expansion coefficients, the procedure outlined in Section 3.3 is followed. However, due to the fact the periodic custom-made basis functions each individually satisfy the boundary conditions, additional decomposition is not

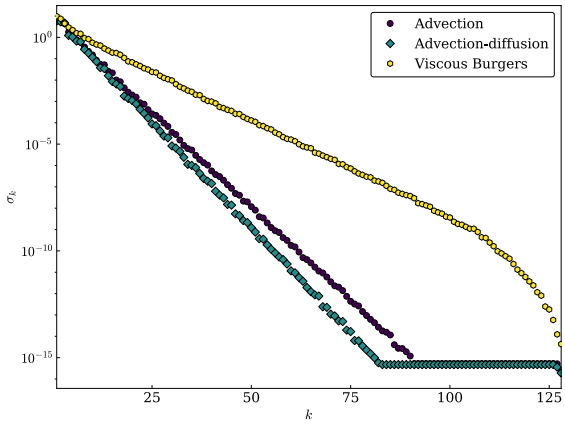
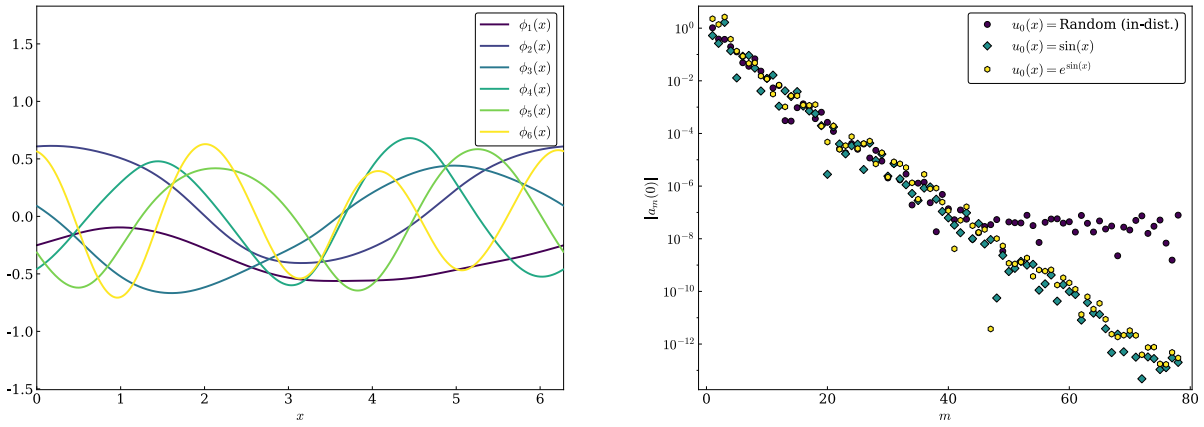


Figure I.1: Singular value spectrum of the periodic custom-made basis functions for the advection, advection-diffusion, viscous Burgers equations.

required, and Equation (3.20) may be used directly. Utilizing this approach and dropping the tildes on $\{\tilde{\phi}_l\}$ results in the following system of ODEs for the advection equation,

$$\frac{da_m(t)}{dt} = -\alpha \sum_{l=1}^{r} a_l(t) \left\langle \phi_m, \frac{d}{dx}\phi_l \right\rangle, \tag{I.5}$$

for $1 \leq m \leq r$. The hierarchical structure of the $r = 78$ periodic custom-made basis functions is further highlighted by the first several functions (Figure I.2a) and the decay of the expansion coefficients of the initial condition (Figure I.2b). The evolution of the relative errors for the three initial conditions (refer to Appendix E for plots of the individual initial conditions) are shown in Figure I.3. For all three initial conditions, good agreement is found between the Fourier solution and the periodic custom-made basis function solution for the entire temporal interval. Furthermore, for all three initial conditions, an improvement in the average relative error is found when compared to the $r = 53$ non-periodic custom-made basis function solution (see Figure 3.8).



(a) Periodic custom-made basis functions

(b) Initial condition expansion coefficients $|a_m(0)|$

Figure I.2: Custom-made basis functions (a) and corresponding initial condition expansion coefficients (b) for the advection equation when using $r = 78$ periodic custom-made basis functions.

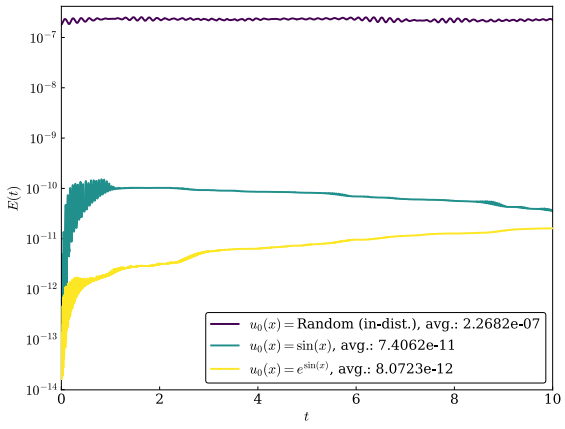


Figure I.3: Relative error evolution of the advection equation for the temporal interval $t \in [0, 10]$ and the three test initial conditions when using the periodic custom-made basis functions.

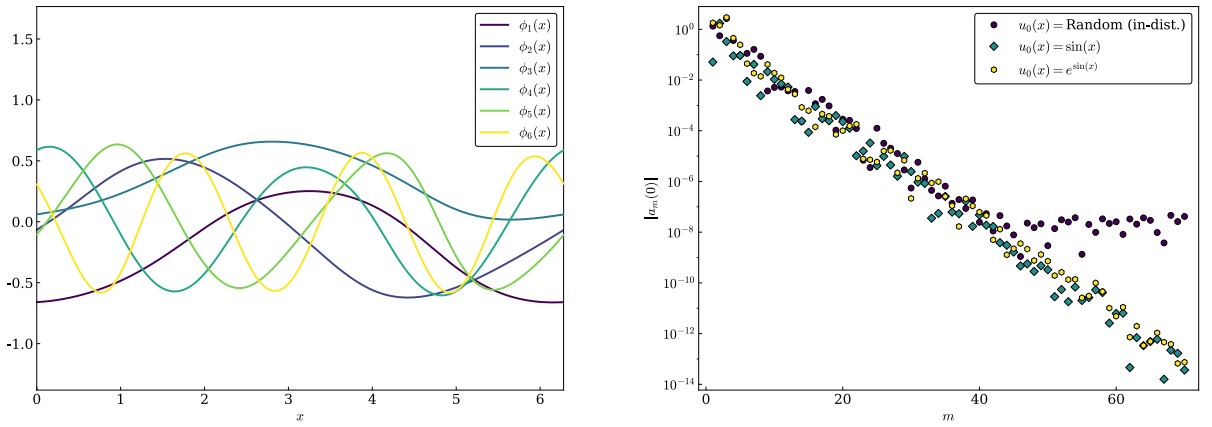
I.2.2 Application 2: 1D Advection-diffusion equation

Next, consider the one-dimensional linear advection-diffusion equation given by Equation (3.25), with $\alpha = 1.0$ and $\nu = 0.1$. Following the Galerkin approach, the system of ODEs for

the expansion coefficients is given by

$$\frac{da_m(t)}{dt} = -\alpha \sum_{l=1}^{r} a_l(t) \left\langle \phi_m, \frac{d}{dx}\phi_l \right\rangle + \nu \sum_{l=1}^{r} a_l(t) \left\langle \phi_m, \frac{d^2}{dx^2}\phi_l \right\rangle, \tag{I.6}$$

for $1 \leq m \leq r$ and where $r = 70$. The first six periodic custom-made basis functions identified for the advection-diffusion equation are shown in Figure I.4a, while the expansion coefficients of the initial condition are shown in Figure I.4b. Figure I.5 shows the evolution of the relative errors for the three initial conditions and shows strong agreement is again obtained with the Fourier solution. An increase in accuracy is found for the in-distribution initial when compared to the $r = 59$ non-periodic custom-made basis function solution (see Figure 3.12), but a decrease in accuracy is found for the two out-of-distribution initial conditions.



(a) Periodic custom-made basis functions

(b) Initial condition expansion coefficients $|a_m(0)|$

Figure I.4: Custom-made basis functions (a) and corresponding initial condition expansion coefficients (b) for the advection-diffusion equation when using $r = 70$ periodic custom-made basis functions.

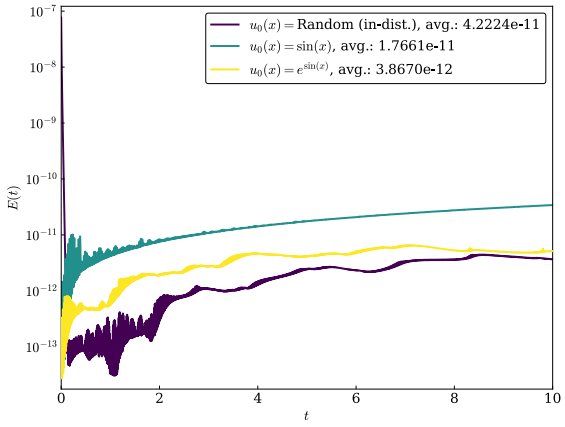


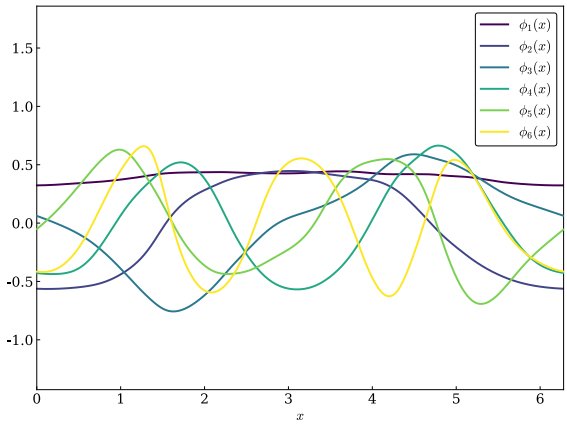
Figure I.5: Relative error evolution of the advection-diffusion equation for the temporal interval $t \in [0, 10]$ and the three test initial conditions when using the periodic custom-made basis functions.

I.2.3 Application 3: 1D Viscous Burgers equation

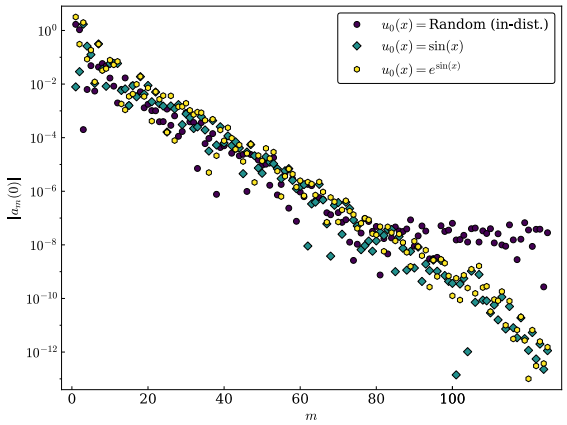
Next, consider the one-dimensional nonlinear viscous Burgers equation given by Equation (3.26), with $\nu = 0.1$. Following the Galerkin approach, the system of ODEs for the expansion coefficients is given by,

$$\frac{da_m(t)}{dt} = -\sum_{l=1}^{r}\sum_{h=1}^{r} a_l(t)a_h(t)\left\langle \phi_m, \phi_l \frac{d}{dx}\phi_h \right\rangle + \nu \sum_{l=1}^{r} a_l(t)\left\langle \phi_m, \frac{d^2}{dx^2}\phi_l \right\rangle, \tag{I.7}$$

for $1 \leq m \leq r$ and where $r = 125$. The first six periodic custom-made basis functions identified for the viscous Burgers equation and the expansion coefficients of the initial condition are shown in Figure I.6. The evolution of the relative errors for the three initial conditions are shown in Figure I.7. Strong agreement is found with the Fourier solution, and an improvement in the accuracy is found for all three initial conditions when compared to the $r = 91$ non-periodic custom-made basis function solution (see Figure 3.16).



(a) Periodic custom-made basis functions



(b) Initial condition expansion coefficients $|a_m(0)|$

Figure I.6: Custom-made basis functions (a) and corresponding initial condition expansion coefficients (b) for the viscous Burgers equation when using $r = 125$ periodic custom-made basis functions.

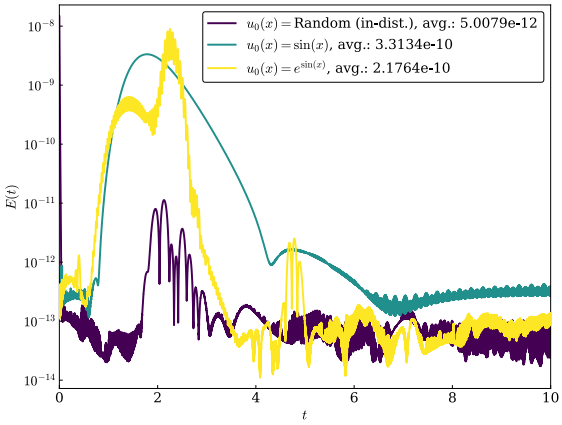


Figure I.7: Relative error evolution of the viscous Burgers equation for the temporal interval $t \in [0, 10]$ and the three test initial conditions when using the periodic custom-made basis functions.

Appendix J

WALL-CLOCK TIMES FOR THE CUSTOM-MADE BASIS FUNCTION SOLUTIONS

This appendix contains wall-clock timings for the custom-made basis function solutions along with time comparisons to Fourier methods. The mean wall-clock times for each of the six example PDEs are shown in Tables J.1, J.2, J.3, J.4, J.5, and J.6. The time presented for the advection, advection-diffusion, viscous Burgers, Korteweg–de Vries, and Kuramoto–Sivashinsky equations is the time required to numerically integrate the r or M size system of ordinary differential equations corresponding to the expansion or modal coefficients for $t \in [0, 10]$. For the Fourier solution, conjugacy was not exploited, and all modal coefficients (positive and negative) were evolved. The mean wall-clock time presented for the inviscid Burgers equation is the time required to numerically integrate the r size system of ordinary differential equations corresponding to the expansion coefficients or the MUSCL solution for the stated spatial discretization for 0.8296 seconds (as is explained in Section 3.4.6, after that time the spectral method based on custom-made basis functions becomes increasingly inaccurate). The in-distribution initial condition was used to initialize the models, the mean was calculated based on three runs (post initial compilation run), and all tests were conducted on an Intel Core i9-9900K CPU. All presented results are for the parameter values for which DeepONets were trained, e.g., $\alpha = 1.0$, $\nu = 0.1$.

The errors presented for the Fourier solutions are based on the same 128-node Gauss–Legendre quadrature grid used for the custom-made basis function solutions. For the nonlinear models, custom-made basis function results are shown for two different treatments of the nonlinear term; the triple product integral, which computes the nonlinear term in the expansion coefficient space, and a pseudo-spectral transform, which transforms the expansion

coefficient solution to real space, performs the required multiplication, and then returns the solution to the expansion coefficient space. The transformation from expansion coefficient to real space is achieved by directly calculating the series expansion, while the transformation from real to expansion coefficient space is achieved by directly calculating the necessary inner products. Refer to Appendix K for additional details.

Solution method	Mean wall-clock time (s)	Average relative error
Custom-made basis function solution ($r = 53$)	0.007680	3.0254×10^{-7}
Fourier solution ($M = 54$)	0.021379	3.6084×10^{-7}
Fourier reference solution ($M = 128$)	0.040358	–

Table J.1: Mean wall-clock times in seconds and average relative errors for the advection equation computed using the custom-made basis function and Fourier solution methods. The average errors were computed for $t \in [0, 10]$.

Solution method	Mean wall-clock time (s)	Average relative error
Custom-made basis function solution ($r = 59$)	0.293744	7.7971×10^{-11}
Fourier solution ($M = 60$)	0.035615	2.6889×10^{-8}
Fourier reference solution ($M = 128$)	0.120678	–

Table J.2: Mean wall-clock times in seconds and average relative errors for the advection-diffusion equation computed using the custom-made basis function and Fourier solution methods. The average errors were computed for $t \in [0, 10]$.

Solution method	Mean wall-clock time (s)	Average relative error
Custom-made basis function solution ($r = 91$) Calculated using triple product integral	230.190098	1.4225×10^{-8}
Custom-made basis function solution ($r = 91$) Calculated using pseudo-spectral transform	24.947826	1.4225×10^{-8}
Fourier solution ($M = 92$)	4.035121	9.7757×10^{-11}
Fourier reference solution ($M = 128$)	5.343368	–

Table J.3: Mean wall-clock times in seconds and average relative errors for the viscous Burgers equation computed using the custom-made basis function and Fourier solution methods. The average errors were computed for $t \in [0, 10]$.

Solution method	Mean wall-clock time (s)	Average relative error
Custom-made basis function solution ($r = 106$) Calculated using triple product integral	108.112182	1.8670×10^{-5}
Custom-made basis function solution ($r = 106$) Calculated using pseudo-spectral transform	7.249629	1.8670×10^{-5}
Fourier solution ($M = 106$)	39.179805	8.0910×10^{-7}
Fourier reference solution ($M = 512$)	235.160790	–

Table J.4: Mean wall-clock times in seconds and average relative errors for the Korteweg–de Vries equation computed using the custom-made basis function and Fourier solution methods. The average errors were computed for $t \in [0, 10]$.

Solution method	Mean wall-clock time (s)	Average relative error
Custom-made basis function solution ($r = 105$) Calculated using triple product integral	3640.652389	7.6522×10^{-4}
Custom-made basis function solution ($r = 105$) Calculated using pseudo-spectral transform	304.400205	7.6634×10^{-4}
Fourier solution ($M = 106$)	83.722244	3.3127×10^{-4}
Fourier reference solution ($M = 512$)	391.701459	–

Table J.5: Mean wall-clock times in seconds and average relative errors for the Kuramoto–Sivashinsky equation computed using the custom-made basis function and Fourier solution methods. The average errors were computed for $t \in [0, 10]$.

Solution method	Mean wall-clock time (s)	Average relative error
Custom-made basis function solution ($r = 101$) Calculated using triple product integral	1.869812	8.9672×10^{-4}
Custom-made basis function solution ($r = 101$) Calculated using pseudo-spectral transform	0.204423	8.5049×10^{-4}
MUSCL solution Discretized using spatial grid of 102 points	0.005417	5.3397×10^{-3}
MUSCL reference solution Discretized using a spatial grid of 4096 points	1.217919	–

Table J.6: Mean wall-clock times in seconds and average relative errors for the inviscid Burgers equation computed using the custom-made basis function and MUSCL solution methods. The average errors were computed for $t \in [0, 0.8296]$.

Appendix K

PSEUDO-SPECTRAL CUSTOM-MADE BASIS TRANSFORM

This appendix presents the results of utilizing a pseudo-spectral custom-made basis function transform for computing the nonlinear term of the viscous Burgers equation with $\nu = 0.1$. The pseudo-spectral transform is computed by utilizing the expansion coefficients and corresponding custom-made basis functions to return the solution to real space, computing the nonlinear term (in real space), and then expanding the solution in terms of the custom-made basis functions prior to advancing to the next time step. The transformation from expansion coefficient to real space is achieved by directly calculating the series expansion, while the transformation from real to expansion coefficient space is achieved by directly calculating the necessary inner products. Padding the solution using the 3/2-rule for de-aliasing [11] was also considered, and the extra custom-made basis functions required to pad the pseudo-spectral transform necessitated the usage of $r = 60$ custom-made basis functions for the results presented in this appendix, rather than the $r = 91$ custom-made basis functions presented in Section 3.4.3 and Appendix G.3. The value of $r = 60$ was chosen so that the remaining custom-made basis functions utilized for de-aliasing ($k = 61, 62, \ldots, 91$) exceeded the singular value threshold of 10^{-13}.

The evolution of the relative error for the three test initial conditions when computing the nonlinear term in modal space using the triple product integral—similar to what was done in Section 3.4.3 and Appendix G.3—are shown in Figure K.1. The evolution of the relative error for the three test initial conditions when utilizing the pseudo-spectral transform, without padding by the 3/2-rule, are shown in Figure K.2a, while the results utilizing the pseudo-spectral transform padded by the 3/2-rule are shown in Figure K.2b. The average relative errors are comparable between the triple product integral results, the pseudo-spectral

transform results with no padding, and the pseudo-spectral transform results padded by the 3/2-rule. Similar results were found when using all the custom-made basis functions exceeding the singular value threshold of 10^{-13} and the pseudo-spectral transform for all nonlinear example PDEs. Refer to the average relative errors presented in Tables J.3, J.4, J.5, and J.6 in Appendix J.

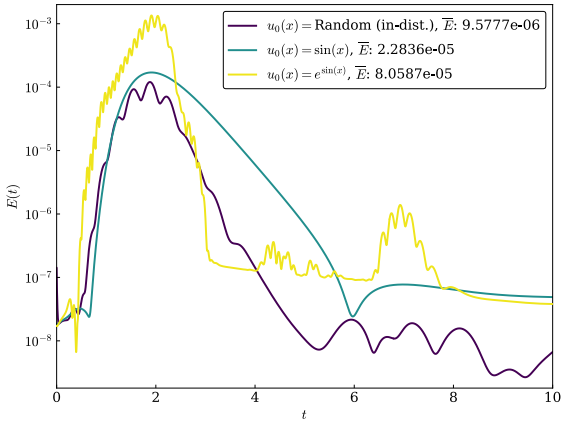
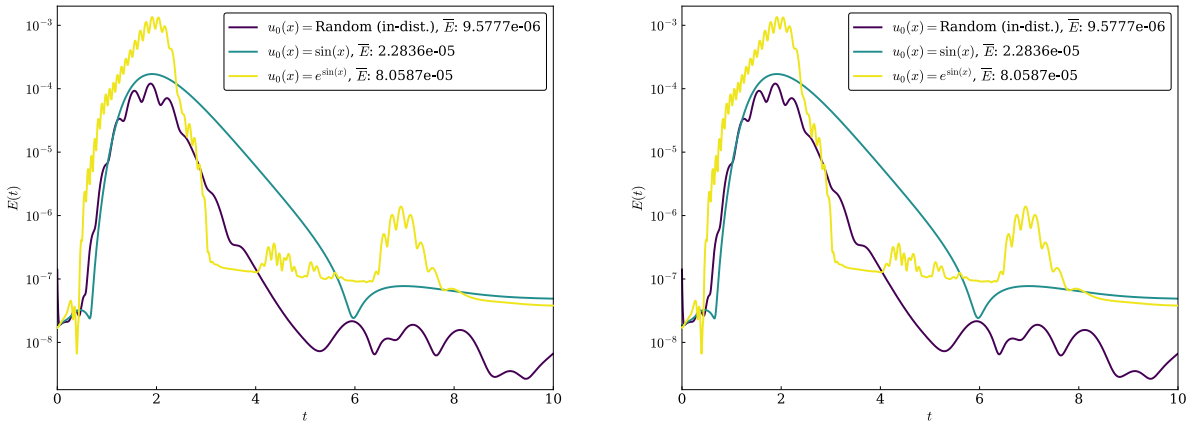


Figure K.1: Relative error evolution of the viscous Burgers equation for the temporal interval $t \in [0, 10]$ for the three test initial conditions when using $r = 60$ custom-made basis functions and computing the nonlinear term in modal space using the triple product integral.



(a) No solution padding

(b) Padded using 3/2-rule

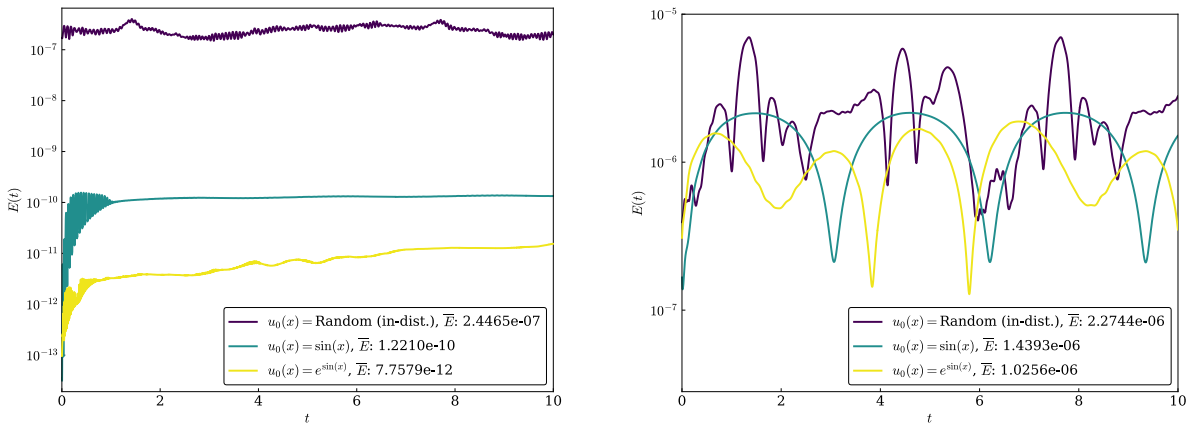
Figure K.2: Relative error evolution of the viscous Burgers equation for the temporal interval $t \in [0, 10]$ for the three test initial conditions when using $r = 60$ custom-made basis functions and computing the nonlinear term using the pseudo-spectral transform.

Appendix L

USING THE CUSTOM-MADE BASIS FUNCTIONS FROM ONE PDE TO EXPAND THE SOLUTION OF ANOTHER PDE

This appendix contains results obtained when using the custom-made basis functions identified for a PDE that exhibits more complex dynamics to expand the solution of an alternative PDE that exhibits less complex dynamics. Two test cases were considered: using the custom-made basis functions identified for the viscous Burgers equation to expand the solution of the advection equation and using the custom-made basis functions identified for the Korteweg–de Vries equation to expand the solution of the viscous Burgers equation.

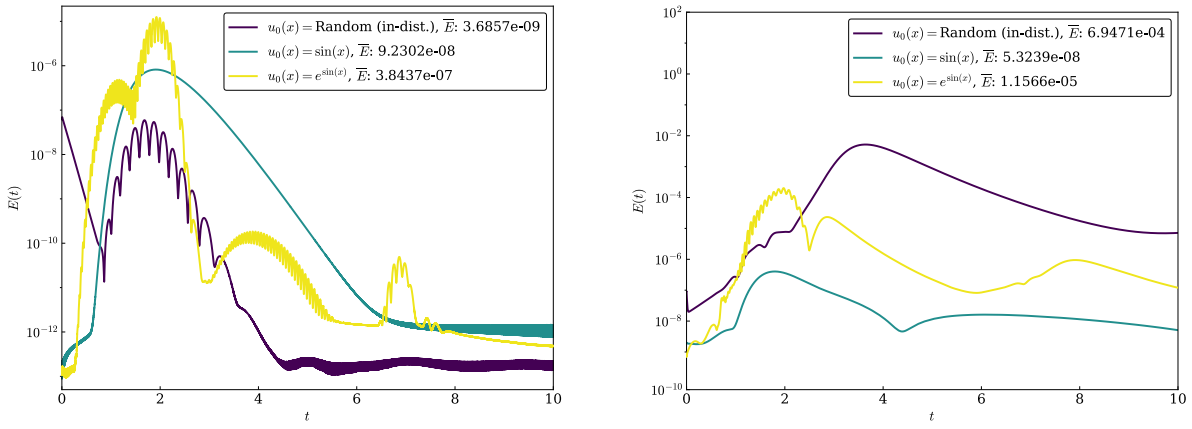
Figure L.1a shows the evolution of the relative error for the three test initial conditions when using the $r = 91$ custom-made basis functions identified for the viscous Burgers equation ($\nu = 0.1$) to expand the solution of the advection equation. Figure L.1b shows the evolution of the relative errors when using $r = 53$ of the custom-made basis identified for the viscous Burgers equation. The value of $r = 53$ matches the number of basis functions identified for the advection equation. Refer to Figure 3.8 for results when using the $r = 53$ custom-made basis functions identified specifically for the advection equation. Figure L.1a shows that when utilizing all $r = 91$ custom-made basis functions identified for the viscous Burgers equation, the average errors are of a similar order as those obtained when using the $r = 53$ custom-made basis functions (Figure 3.8) found for the advection equation, but there is a slight improvement in the accuracy for the in-distribution initial condition, and a slight decrease in accuracy for the two out-of-distribution initial conditions. When using only $r = 53$ of the custom-made basis functions identified for the viscous Burgers equations (Figure L.1b), a decrease in accuracy is found when compared to using the $r = 53$ custom-made basis functions identified advection equation.



(a) $r = 91$ (b) $r = 53$

Figure L.1: Relative error evolution of the advection equation for the temporal interval $t \in [0, 10]$ and the three test initial conditions when utilizing $r = 91$ (a) and $r = 53$ (b) custom-made basis functions identified for the viscous Burgers equation to expand the solution.

Figure L.2a shows the evolution of the relative error for the three test initial conditions when using the $r = 105$ custom-made basis functions identified for the Korteweg–de Vries equation ($\delta = 0.1$) to expand the solution of the viscous Burgers equation with $\nu = 0.1$. Figure L.2b shows the evolution of the relative errors when using $r = 91$ of the custom-made basis identified for the Korteweg–de Vries equation. The value of $r = 91$ matches the number of basis functions identified for the viscous Burgers equation. Refer to Figure 3.16 for results when using the $r = 91$ custom-made basis functions identified specifically for the viscous Burgers equation. When using $r = 105$ custom-made basis identified for the Korteweg–de Vries equation results in an increase in accuracy for all three initial conditions (Figure L.2a) when compared to the $r = 91$ custom-made basis functions identified for the viscous Burgers equation (Figure 3.16). When utilizing $r = 91$ custom-made basis functions identified for the Korteweg–de Vries equation, a decrease in accuracy is found when compared to using the $r = 91$ custom-made basis functions identified for the viscous Burgers equations.



(a) $r = 105$ (b) $r = 91$

Figure L.2: Relative error evolution of the viscous Burgers equation for the temporal interval $t \in [0, 10]$, the three test initial conditions, and $\nu = 0.1$ when utilizing $r = 105$ (a) and $r = 91$ (b) custom-made basis functions identified for the Korteweg–de Vries equation to expand the solution.

Appendix M

DEVELOPMENT OF A DEEPONET CUSTOM-MADE BASIS INVERSE TRANSFORM

This appendix outlines the development of a DeepONet custom-made basis function inverse transform along with numerical results for the viscous Burgers equation. The inverse transform takes the expansion coefficients as inputs and returns the corresponding function value at a specified spatial location. For the inverse transform DeepONet, the branch layers include bias, the output layer of the branch network did not apply an activation function, and the network width was constant for all layers.

M.1 Generation of ground truth data and DeepONet training parameters

The training and testing solution data for the viscous Burgers equation ($\nu = 0.1$) solved for $t \in [0, 1]$, same as for the DeepONets outlined in Appendix E, was also utilized as the starting point for generating the ground truth data for training the DeepONet inverse transform. Rather than using this data directly, the solution space was randomly sampled temporally for each of the training and testing initial conditions to generate a set of solution snapshots. For each of these solution snapshots, the inner product was then computed to obtain the expansion coefficients corresponding to the custom-made basis functions. Feedforward neural networks were employed for all branch and trunk networks. The expansion coefficients were specified as the input to the branch network. The input to the trunk network were spatial locations from the solution snapshot $u(x)$. Twenty-five example solution snapshots are shown in Figure M.1. Ninety-one custom-made basis functions were utilized to generate the expansion coefficients required for training the DeepONet. The network parameters specified for the DeepONet inverse transform are presented in Table M.1. The mean testing

error and corresponding standard deviation based on three training runs are presented in Table M.2.

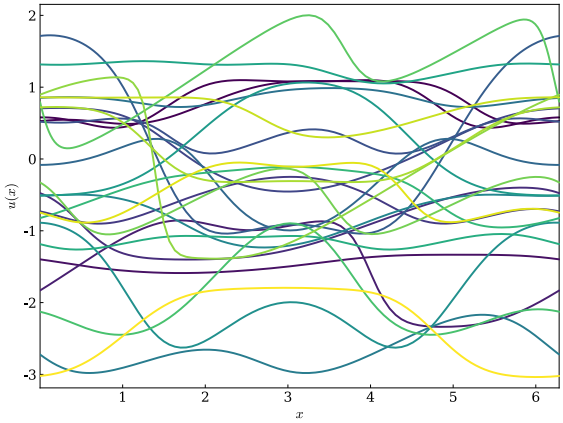


Figure M.1: Twenty-five example input functions randomly sampled from the solution of the viscous Burgers equation for $t \in [0, 1]$ and initialized from the Gaussian random field outlined in Appendix E.

M.2 Numerical results

This section presents results for the three test initial conditions when using $r = 91$ custom-made basis functions to evolve the viscous Burgers equation and with the DeepONet inverse transform utilized for computing the solution through a pseudo-spectral approach. The forward transform was computed by directly calculating the necessary inner products. Refer to Figure 3.16 for reference results for the viscous Burgers equation when using $r = 91$ custom-made basis functions and the triple product integral to compute the nonlinear term in modal space.

Spatiotemporal plots for the in-distribution initial condition are shown in Figure M.2, the out-of-distribution initial condition $u_0(x) = \sin(x)$, in Figure M.3, and the out-of-distribution initial condition $u_0(x) = e^{\sin(x)}$, in Figure M.4. The evolution of the relative errors for the three initial conditions are shown in Figure M.5. The results show that the custom-made basis function solution is able to qualitatively capture the solution characteristics, but with

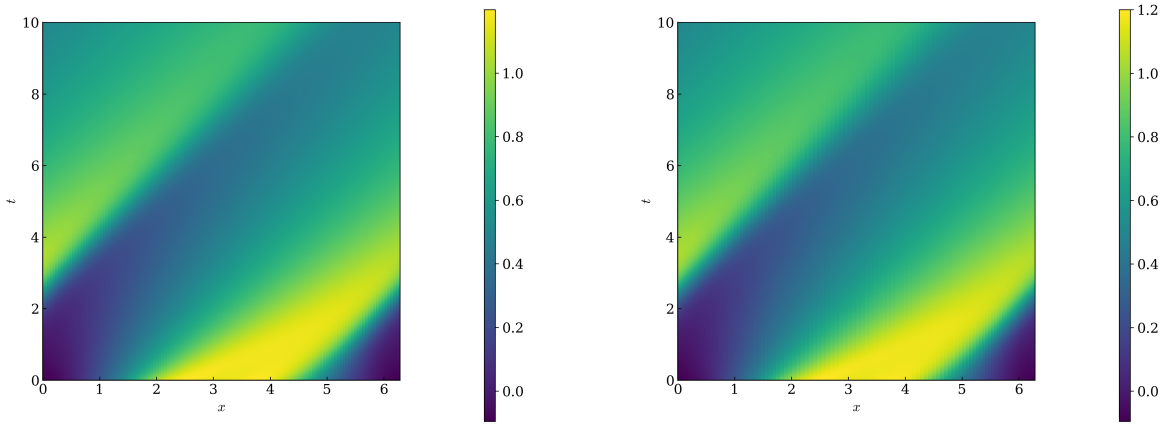
Parameter	Setting
Activation functions	ReLU
Optimizer	Adam
Error	Mean squared error
Learning rate	1×10^{-4}
Number of training epochs	10000
Number of sensors	91
Number of solution locations	40
Number of training input coefficient sets	18×500
Number of testing input coefficient sets	18×1000
Branch net depth	2
Branch net width	100
Trunk net depth	3
Trunk net width	100
Weight initialization	Glorot uniform
Bias initialization	Zero
Mini-batch size	40

Table M.1: Parameter settings for training the custom-made basis function DeepONet inverse transform.

Transform	Mean	Standard deviation
Inverse	8.08×10^{-5}	1.51×10^{-5}

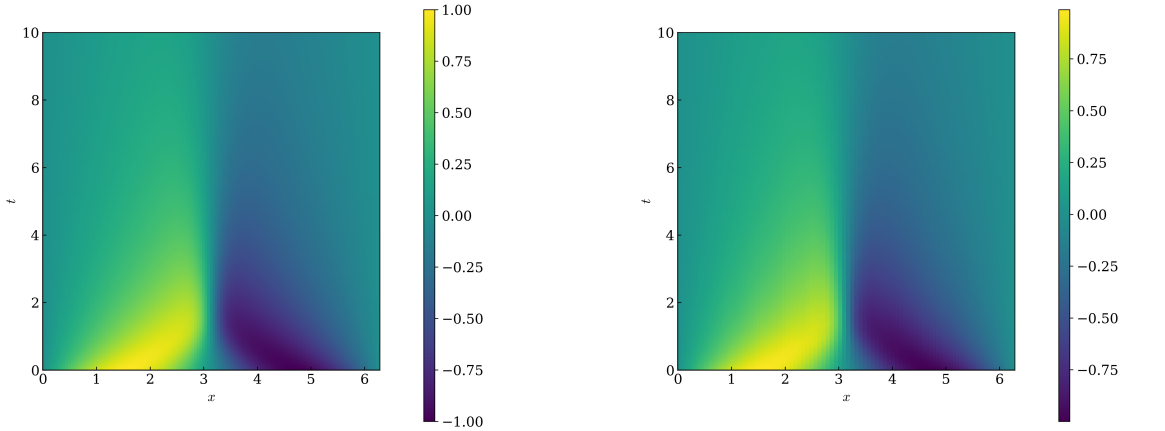
Table M.2: DeepONet mean testing errors and standard deviation for the DeepONet inverse transform based on three training runs each.

a lower average relative error than found when using the triple product integral (refer to Figure 3.16). Despite the decrease in accuracy, these results are comparable to Fourier operator methods for $\nu = 0.1$ [44].



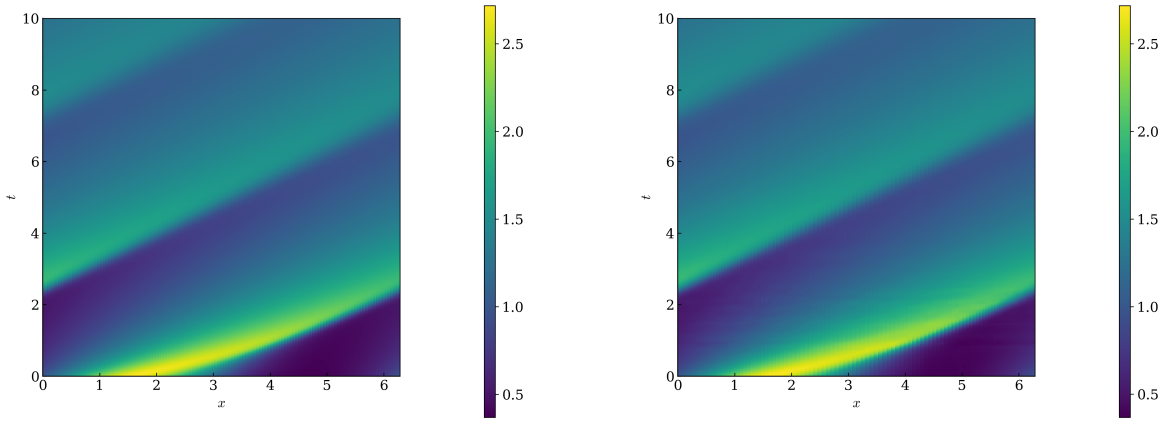
(a) Fourier solution (b) Custom-made basis solution

Figure M.2: Spatiotemporal evolution of the viscous Burgers equation for the temporal interval $t \in [0, 10]$ and the random in-distribution initial condition when using the DeepONet inverse transform with a pseudo-spectral approach.



(a) Fourier solution (b) Custom-made basis solution

Figure M.3: Spatiotemporal evolution of the viscous Burgers equation for the temporal interval $t \in [0, 10]$ and the out-of-distribution initial condition $u_0(x) = \sin(x)$, when using the DeepONet inverse transform with a pseudo-spectral approach.



(a) Fourier solution (b) Custom-made basis solution

Figure M.4: Spatiotemporal evolution of the viscous Burgers equation for the temporal interval $t \in [0, 10]$ and the out-of-distribution initial condition $u_0(x) = e^{\sin(x)}$, when using the DeepONet inverse transform with a pseudo-spectral approach.

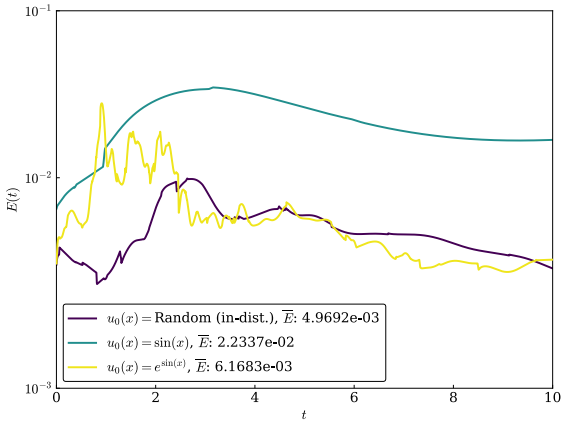


Figure M.5: Relative error evolution of the viscous Burgers equation for the temporal interval $t \in [0, 10]$ and the three test initial conditions when using the DeepONet inverse transform with a pseudo-spectral approach.