Data-driven inference of symmetry-equivariant models of natural phenomena

Daniel Gurevich

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
by the Program in
Applied and Computational Mathematics
Advisers: Professor Clarence W. Rowley,
Professor Charles L. Fefferman

September 2025

Abstract

This dissertation provides an in-depth description of the SPIDER (Sparse Physics-Informed Discovery of Empirical Relations) framework for data-driven inference of symmetry-equivariant models of spatiotemporally extended physical systems. The homogeneity and isotropy of physical space require mathematical models of natural phenomena to be equivariant with respect to symmetries including translations, rotations, and reflections. Unstructured data-driven model inference leads to an intractable optimization problem due to the high dimensionality of both the spatiotemporal data and the model search space. In contrast, SPIDER exploits tensor representations of the symmetry group, alongside other physical constraints such as local or short-range interactions, to describe systems via sets of tensor-valued partial differential or integro-differential equations, which can parsimoniously and interpretably parametrize a wide range of models.

While prior work touches on necessary ingredients for the successful application of SPIDER to complex real-world problems, such as weak formulation of differential equations and sparse regression, this manuscript addresses essential gaps that have precluded more widespread adoption. Namely, it (1) formalizes an algorithm for automatically generating and manipulating libraries of symmetry-equivariant tensors, (2) describes how to synthesize all logically independent equations in the search space, and (3) generalizes SPIDER to discover mean-field models of many-body systems composed of discrete particles or agents. These innovations are implemented as part of a software package, `PySPIDER`, and illustrated using two examples representing an incompressible fluid flow in the continuum limit and a compressible fluid consisting of discrete interacting particles.

Acknowledgements

The work presented in this thesis would not have been possible without the support of many friends and colleagues. I am incredibly grateful to my parents for their unwavering support throughout my academic journey. My advisors, Prof. Clarence Rowley and Prof. Charles Fefferman, have shaped me as a researcher and educator through their guidance, wisdom, and encouragement.

I would particularly like to thank Akash Gaonkar, who has been like a third advisor to me for the majority of my graduate studies. Akash has consistently and selflessly shared his curiosity, patience, and expertise with me ever since we met, and I could not be more grateful for his mentorship and friendship. This work could not have happened without his support.

I would like to thank my other committee members, Prof. Zachary Kincaid and Prof. Howard Stone, for their extensive feedback on my work; Prof. Daniel Sussman for his collaboration and providing data that I used extensively in testing discrete SPIDER; and Prof. Roberto Car, Prof. Ryan Cory-Wright, Prof. Santanu Dey, Prof. Liza Rebrova, Dr. Pablo Piaggi, and Dr. Pinchen Xie for many helpful discussions.

My collaborators at Georgia Tech have made indispensable contributions to the approach that would ultimately become known as SPIDER. I thank Prof. Roman Grigoriev, Dr. Matthew Golden, and Dr. Patrick Reinbold for their extensive help in developing conceptual machinery; Carlos de Oliveira e Silva Filho for testing and contributing to the `PySPIDER` codebase; and Matteo Ugliotti and Brandon Choi for extensive alpha testing of `PySPIDER` v2.

Additionally, I am grateful for the funding sources that made my research possible: the Air Force Office of Scientific Research under grant FA9550-19-1-0005, the National Science Foundation Graduate Research Fellowship Program, and Princeton University.

To the gremlins.

Contents

Abstract 3

Acknowledgements 4

List of Tables 8

List of Figures 10

1 Introduction **11**

1.1 Symbolic and Sparse Regression 13

1.2 Applying SPIDER to a Toy Example 18

1.3 Contributions of This Thesis 20

2 Symmetry-Equivariant Tensor Libraries **24**

2.1 Review of Group Representations 24

2.2 Notation and Libraries for Continuous Data 27

2.3 Symmetry Breaking and Boundary Conditions 37

2.4 Mean-Field Models for Many-Body Systems 40

3 Automatic Generation and Manipulation of Symbolic Libraries **52**

3.1 Formal Grammars and Semantics 52

3.2 Library Generation 57

3.3 Term Evaluation 63

3.3.1 Weight Functions and Integration by Parts 66

3.3.2 Splitting among Irreducible Representations 69

3.3.3 Evaluating Boundary Conditions 71
3.4 Equation Synthesis 72

4 Robust Sparse Regression **79**
4.1 Quantifying Solution Performance 79
4.2 Nondimensionalization and Normalization 83
4.3 Iterative Heuristic Algorithms for Regression 86
4.4 Certifiably Optimal Mixed-Integer Regression 93
4.5 Initializing Regression 97

5 Illustrative Applications of SPIDER **104**
5.1 Continuous: Incompressible Fluid Flow 104
5.2 Discrete: Compressible Fluid Flow 113

6 Conclusion **126**
6.1 Future Work 127

A PySPIDER code overview **130**

B Supplementary Figures **133**

Bibliography **138**

List of Tables

2.1 Sizes of unsplit hydrodynamic libraries with various combinations of rank and complexity threshold. 33

2.2 Sizes of representative complexity-4 unsplit discrete libraries for various combinations of rank and maximum number of primes per term. . . . 50

2.3 Sizes of representative complexity-5 unsplit discrete libraries for various combinations of rank and maximum number of primes per term. . . . 50

4.1 Ratio of the Frobenius norm and operator norm, $\|G\|_F/\|G\|_{op}$, for the library matrices G used in Chapter 5. 81

4.2 Characteristic scale of the mean m and variation σ for observables in the examples of Chapter 5. 85

4.3 Performance metrics for various initialization strategies used with greedy regression. 101

4.4 Performance metrics for truncated inverse iteration (TII) used with varying numbers of passes of greedy regression. 102

5.1 List of all fundamental equations identified by sparse regression for JAX-CFD dataset using optimal SPIDER hyperparameters. 108

5.2 List of coefficient relative errors for multi-term equations in Table 5.1. 108

5.3 List of all fundamental equations identified by SPIDER for the discrete compressible fluid simulation. 118

5.4 Hybrid residuals r_h of factorized relations after eliminating the factor of $1 - \rho$ and refitting coefficients. 122

List of Figures

1.1 Schematic of the SPIDER algorithm. 17

3.1 Extended Backus-Naur form (EBNF) specification for the continuous SPIDER language. 54

3.2 A sequence of term rewrites transforming the term $\partial_\alpha \partial_\beta u_\alpha \cdot u_\beta$ to its canonical form. 59

3.3 A directed graph showing some of the equations implied by the incompressibility condition $\nabla \cdot \boldsymbol{u} = 0$. 74

5.1 Snapshots of the vorticity field $\omega = \partial_x u_y - \partial_y u_x$ for the JAX-CFD incompressible fluid flow dataset at times $t =$ (a) 0, (b) 1.5, (c) 3, (d) 4.5. 106

5.2 Residual r_F as a function of sparsity k for the best multi-term equations identified by SPIDER for the JAX-CFD dataset. 110

5.3 Snapshots of coarse-grained density ρ (left column) and x component of momentum $\rho[v_x]$ (right column) at time $t = 0.125$ for $h =$ (a)-(b) 0.01, (c)-(d) 0.02, (e)-(f) 0.05. 115

5.4 Residual r_F as a function of sparsity k for the fundamental equations identified by SPIDER for the discrete compressible fluid dataset. . . . 120

B.1 Snapshots of the particle velocity data for the discrete compressible fluid simulation at times $t =$ (a) 0, (b) 0.2, (c) 0.4, (d) 0.6. 134

Chapter 1

Introduction

The symbolic language of equations has been indispensable to scientific discovery over the centuries. Although Johannes Kepler's formulation of the laws of planetary motion in the early 17^{th} century still expressed equations in the verbose, plain language favored by the ancient Greeks, the convention of using single letters to abstractly and compactly denote parameters and variables introduced by François Viète in the late 16^{th} century [63] was soon adopted universally by scientists for formalizing their mathematical models. The invention of differential and integral calculus by Isaac Newton and Gottfried Leibniz in the 17th century was also fundamental to the modern paradigm of quantitative science, as it enabled the use of differential equations to model physical concepts such as time evolution and spatial variation. Today, the same language of differential equations is widely used to provide interpretable and often quantitatively accurate models for complex systems across science and engineering. This is particularly true in physics, where central modeling roles are played by partial differential equations (PDEs) such as Maxwell's equations in electromagnetism, Boltzmann's equation in nonequilibrium statistical mechanics, the Navier-Stokes equation in fluid dynamics, Schrödinger's equation in quantum mechanics, and Einstein's field equations in general relativity. Differential equations also serve as models of processes

in biology and chemistry (reaction-diffusion equations, models of neural and cardiac excitation), epidemiology, population dynamics, financial markets, and so on.

At the same time, mathematical modeling over the last 350 years has often been a time-consuming trial-and-error affair, requiring steady accumulation of domain expertise, iterative testing of theoretical approximations through experiment, and large amounts of ingenuity. Quantitatively validating models is also made more difficult by the sensitive dependence of many models on their parameters, creating the need for careful fine-tuning. As a result, many complex experimental systems in fields such as astrophysics have been studied extensively and documented through large and rich datasets but still lack a confirmed quantitative description.

The challenge of automating scientific discovery has seen active research efforts since the 1960s. Early artificial intelligence (AI) work was largely symbolic in nature, introducing expert systems based on heuristic search and a set of human-specified reasoning rules, such as DENDRAL [43], a special-purpose program for identifying unknown organic molecules from their mass spectra, and BACON [68], which generated and tested candidate empirical laws describing labeled objects in an "artificial two-dimensional universe" while synthesizing new, potentially descriptive attributes. In the past two decades, the prevailing research direction in AI has instead shifted to black-box models based on deep neural networks (NNs). However, when fitting complex spatiotemporally extended physical systems that currently lack rigorous human-derived mathematical models, NN-based data-driven learning has encountered substantial challenges in learning models that are accurate, generalizable, or interpretable. In these cases, poorly structured model inference tends to lead to an intractable optimization problem due to the high dimensionality of both the spatiotemporal data and the model search space, and it is instead necessary to introduce strong inductive biases that constrain the search space. Meanwhile, such inductive

biases are already present in the symbolic structure of models that are based on the familiar, easily interpretable form of differential equations.

1.1 Symbolic and Sparse Regression

Research on synthesizing equation-based models from data can be categorized primarily into two distinct approaches with the (often confusingly) similar names of symbolic regression [15, 102, 35] and sparse regression [23, 127, 126, 99, 101, 82, 60, 105, 106] (the latter having precursors in Refs. [93] and [30]). Symbolic regression aims to identify general nonlinear equations that are (approximately) satisfied by the data:

$$\mathcal{F}(\boldsymbol{x}) = 0 \quad \forall \boldsymbol{x} \in X, \tag{1.1}$$

where $\mathcal{F}$ can be *any* syntactically valid expression involving the data $\boldsymbol{x}$ within the input dataset X as well as constants, user-specified operators such as addition, multiplication, exponentiation, derivatives, and other functions. In practice, parsimony of the model is enforced by searching among short expressions with only a few elements, although this search problem is NP-hard [118] and challenging in practice. Symbolic regression approaches using evolutionary programming to generate candidate equations, such as Eureqa [15, 102, 35] and PySR [29], have been moderately successful in learning both algebraic and differential equations from scientific datasets. In particular, symbolic regression methods based on gene-expression programming [45] were able to discover a range of common two-dimensional PDEs [117] as well as coarse-grained PDE models of fluid flows from molecular dynamics simulations in [125, 76]. Symbolic regression using deep NNs [80, 100] has also gained traction; in particular, transformer-based architectures are commonly used [14] and recently identified a range of PDE models in physics [72, 108]. Unfortunately, symbolic regression is not as well-suited to learning the compact relations involving tensor-valued variables that are

universal in physics, since constraining the expressions $\mathcal{F}$ to be symmetry-equivariant is challenging.

In contrast, sparse regression introduces an additional inductive bias in the form of the identified equations. Sparse regression searches for equations that are linear combinations of a few shorter nonlinear expressions:

$$\sum_{k \in S} c_k \boldsymbol{f}_k(\boldsymbol{x}) = 0 \quad \forall \boldsymbol{x} \in X, \tag{1.2}$$

where the list (known as a *dictionary* or *library*) of all candidate sub-expressions (generally called *terms*) $\boldsymbol{f}_k$ is by construction small enough to explicitly enumerate, and the support set S has small cardinality. Indeed, many physical systems evolve under the influence of a moderate number of decoupled effects or feature dynamics that can be approximated by a truncated series; as a result, most human-discovered differential equations have this sum-of-terms structure. Unlike symbolic regression, where fitting the constants in an equation is particularly challenging unless they are constrained to be integers or simple fractions, sparse regression can learn the term coefficients c_k as well as the support S using more efficient constrained linear regression algorithms. (Note that the bulk of sparse regression literature assumes an inhomogeneous formulation where one of the terms f_k is moved to the right-hand side of (1.2). However, it is generally impossible to assume the presence of any specific term in the equation, whereas this issue is easily sidestepped by confining all terms to the left-hand side, with only small modifications to the regression algorithm.)

Both symbolic and sparse regression could more broadly be considered to fall under the umbrella of the computer science subfield of *program synthesis*, which replaces the language of calculus with the more general language of computer programs and exchanges the task of producing equality for constructing a program that satisfies a formal specification, that is, a set of logical assertions that must hold for its outputs

[55]. Moreover, there exists a compelling analogy between the problem of equation learning considered in this dissertation and the problem of *invariant synthesis* in computer science. Invariants are assertions that continuously hold throughout a phase of program execution. Although invariants are often identified by hand and verified through formal proofs, this approach is similarly laborious for programs beyond a modest level of complexity, leading to recent work on data-driven identification of invariants based on concrete examples of states occurring during program execution [38, 86, 124]. Although many techniques from program synthesis could likely be directly applied to the present topic, we will generally not explore these analogies further than borrowing a few basic definitions and techniques from programming language theory.

The general characterization of sparse regression above omits several concrete details:

1. which terms are to be included in a library,

2. how these terms are actually evaluated using the data, and

3. how the coefficients and support are learned.

In fact, these choices have proven to be crucial to the practical success of sparse regression in modeling real-world experimental systems, which involve noisy measurements and unknown governing equations. Sparse regression was popularized as part of the Sparse Identification of Nonlinear Dynamics (SINDy) method [20], which is hindered by suboptimal choices on all three fronts. Specifically, SINDy provides no guidance on how to form a library for PDE equations and typically struggles to include all important terms without an explosion in the library size; terms involving derivatives are evaluated pointwise by finite differences, causing extreme sensitivity to noise; and term selection is performed by a simple thresholding procedure that struggles to distinguish between important terms with small coefficients (which is

sensitive to arbitrary term rescaling such as change of units) and ones that are *physically* irrelevant. In addition, SINDy uses an inhomogeneous formulation of (1.2) where the equation is constrained to involve a specific time derivative term. Later work has corrected some of these shortcomings: WSINDy [83, 82] decreases the impact of noise by evaluating terms in weak form (i.e., integrating them with respect to a test function), a technique originally introduced in Ref. [57]. SINDY-PI [62] uses a more general regression formulation that can discover relations other than evolution equations. Other weaknesses of the SINDy method, which derive from the fact that it is an almost purely data-driven method that uses limited domain knowledge, persist even in more recently introduced versions of the method.

In contrast, the SPIDER (Sparse Physics-Informed Discovery of Empirical Relations) framework [57, 97, 98, 61, 52, 50, 56, 51] takes advantage of essential domain knowledge to prescribe answers to all three of these questions. The first key assumption is that the physical laws governing the system are equivariant with respect to some continuous symmetry group (though this need not be true of the *geometry* of the system or symmetry-breaking effects such as exogenous forcing—I will further discuss handling of symmetry-breaking in Section 2.3). Indeed, almost all of physics is built on this same assumption. For instance, physical space is generally homogeneous and isotropic, and so the equations of physics are correspondingly equivariant to translations and rotations. This assumption is reflected in SPIDER's use of tensor-valued data and equations, which not only enforces the physically necessary conditions for a valid model but also starkly reduces the number of terms needed to express the same laws [52]. SPIDER also generally assumes homogeneity of space, which is reflected in the absence of the space and time coordinates in the library in explicit form. The second assumption is the principle of locality, which states that all interactions in the system are local (or, at worst, short-ranged), which allows the use of partial differential equations as the fundamental form of the model. Under these assumptions, terms

should be formed from tensor products of the observed fields and the differential operators acting on them, each transforming as tensors with respect to the symmetry group of the system, and libraries in turn consist of all terms contained within an irreducible representation of the symmetry group. Evaluation of terms should occur in weak form over extended spatiotemporal domains to minimize sensitivity to noise. Finally, coefficients should be identified using a regression algorithm that takes into account the physically relevant scales of the system. Figure 1.1 summarizes the SPIDER algorithm in schematic form. For an excellent self-contained introduction to SPIDER, the reader may refer to Ref. [52].

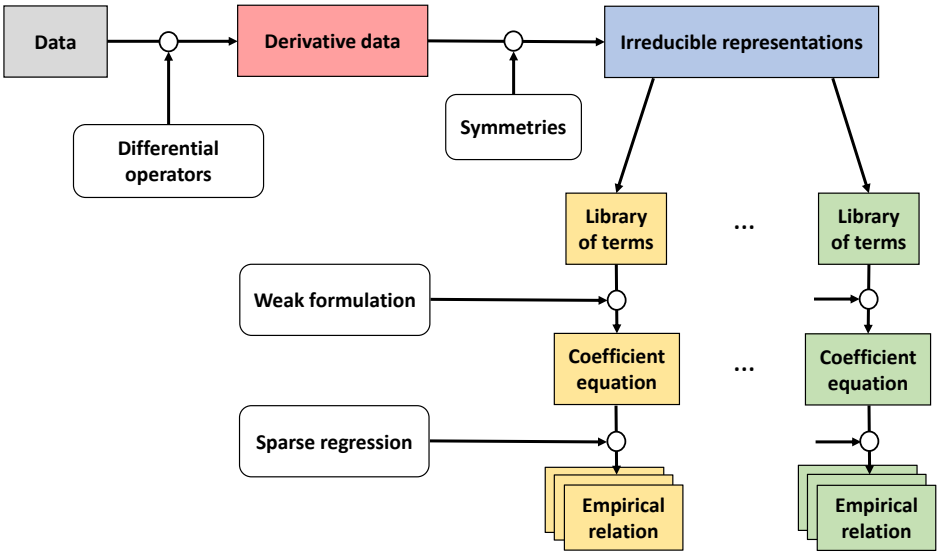


Figure 1.1: **A schematic of the SPIDER algorithm from Ref. [50] (used with author's permission).** The data and differential operators transforming as tensors are combined to generate libraries of tensor-valued terms corresponding to the irreducible representations of the symmetry group of the system. For each library, all of its terms are evaluated in weak form on many slices of the dataset to form a matrix equation $G\boldsymbol{c} = 0$. Multiple sparse and normalized approximate solutions $\boldsymbol{c}$ to the equation are generated, producing a list of parsimonious empirical relations.

1.2 Applying SPIDER to a Toy Example

As an example of this procedure, let us examine the toy problem of learning from data the inviscid Burgers' equation in three spatial dimensions,

$$\partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = 0 \tag{1.3}$$

This is a vector-valued PDE, so we would search for such a relation in a larger library of candidate vector terms, say

$$\mathcal{L} = \{\underbrace{\boldsymbol{u}}_{\boldsymbol{f}_1}, \underbrace{\partial_t \boldsymbol{u}}_{\boldsymbol{f}_2}, \underbrace{\boldsymbol{u}(\nabla \cdot \boldsymbol{u})}_{\boldsymbol{f}_3}, \underbrace{(\boldsymbol{u} \cdot \nabla)\boldsymbol{u}}_{\boldsymbol{f}_4}, \underbrace{\boldsymbol{u} \cdot (\nabla \boldsymbol{u})}_{\boldsymbol{f}_5}\} \tag{1.4}$$

(Of course, real libraries are usually much more flexible, including anywhere between 10 to thousands of terms.) In principle, each of these terms $\boldsymbol{f}_i \in \mathcal{L}$ could immediately be evaluated pointwise using the available data, resulting in a high-dimensional vector with one element per grid point of the data, but as mentioned above, such an approach would amplify noise. Instead, we evaluate the terms using carefully chosen linear maps. We define a set of hypercubic spatiotemporal domains

$$\Omega_i = \{(x, y, z, t) \mid \underline{x}_i \leq x \leq \overline{x}_i, \underline{y}_i \leq y \leq \overline{y}_i, \underline{z}_i \leq z \leq \overline{z}_i, \underline{t}_i \leq t \leq \overline{t}_i\} \tag{1.5}$$

and *weight* (or test) *functions* $\boldsymbol{w}_j$ that are each compactly supported on Ω_j and have their first few spatial and temporal derivatives vanishing on the boundary of Ω_j. (In this example, at a minimum, the weight and its first derivative must vanish, matching the highest derivative order in the library.) Then, we may define a natural inner product between terms and weights by

$$\langle \boldsymbol{f}_i, \boldsymbol{w}_j \rangle = \int_{\Omega_j} (\boldsymbol{f}_i(x, y, z, t) \cdot \boldsymbol{w}_j(x, y, z, t))\, d\boldsymbol{x}^3\, dt \tag{1.6}$$

This construction usually allows for integration by parts to shift derivatives from the term, which is evaluated using the noisy data, onto the weight function, which has been specified in closed form. For instance, note that

$$\begin{aligned}\langle \boldsymbol{u}\cdot(\nabla \boldsymbol{u}), \boldsymbol{w}_j\rangle &= \frac{1}{2}\langle \nabla(\boldsymbol{u}\cdot\boldsymbol{u}), \boldsymbol{w}_j\rangle = \frac{1}{2}\int_{\Omega_j} \nabla(\boldsymbol{u}\cdot\boldsymbol{u})\cdot \boldsymbol{w}_j \, d\boldsymbol{x}^3\, dt \\ &= -\frac{1}{2}\int_{\Omega_j} (\boldsymbol{u}\cdot\boldsymbol{u})(\nabla\cdot \boldsymbol{w}_j)\, d^3\boldsymbol{x}\, dt\end{aligned} \tag{1.7}$$

since the boundary terms involving $\boldsymbol{w}_j$ vanish. By evaluating all of the inner products between the five terms in the library and $n > 5$ weight functions, we transform (1.2) into an overdetermined matrix equation

$$G\boldsymbol{c} = 0 \tag{1.8}$$

where

$$G = \begin{bmatrix} \langle \boldsymbol{f}_1, \boldsymbol{w}_1\rangle & \langle \boldsymbol{f}_2, \boldsymbol{w}_1\rangle & \dots & \langle \boldsymbol{f}_5, \boldsymbol{w}_1\rangle \\ \langle \boldsymbol{f}_1, \boldsymbol{w}_2\rangle & \langle \boldsymbol{f}_2, \boldsymbol{w}_2\rangle & \dots & \langle \boldsymbol{f}_5, \boldsymbol{w}_2\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \boldsymbol{f}_1, \boldsymbol{w}_n\rangle & \langle \boldsymbol{f}_2, \boldsymbol{w}_n\rangle & \dots & \langle \boldsymbol{f}_5, \boldsymbol{w}_n\rangle \end{bmatrix} \tag{1.9}$$

If the noise in the data is not too large, then at least one sparse normalized vector $\boldsymbol{c}$ achieving a small residual $\|G\boldsymbol{c}\|$ will be close to the true coefficient vector $\boldsymbol{c}^* = \left[0, 1/\sqrt{2}, 0, 1/\sqrt{2}, 0\right]^T$, allowing us to discover that Burgers' equation indeed describes the data. (This choice of normalization is arbitrary, but it is most convenient to consider $\|\boldsymbol{c}\| = 1$.)

It is worth comparing the compact PDE model and library used in SPIDER to the much more unwieldy representations that would be used in a standard approach that does not take advantage of the symmetry of the problem, such as SINDy. Rather than learning a single simple equation (1.3), such a method would instead need to

identify three substantially more complex relations independently of each other:

$$\partial_t u_x + u_x \partial_x u_x + u_y \partial_y u_x + u_z \partial_z u_x = 0 \tag{1.10}$$

$$\partial_t u_y + u_x \partial_x u_y + u_y \partial_y u_y + u_z \partial_z u_y = 0 \tag{1.11}$$

$$\partial_t u_z + u_x \partial_x u_z + u_y \partial_y u_z + u_z \partial_z u_z = 0 \tag{1.12}$$

The corresponding library, containing arbitrary combinations of the three components of $\boldsymbol{u}$ and three distinct spatial derivative operators, would contain a much larger number of terms as well.

1.3 Contributions of This Thesis

SPIDER has already achieved impressive results in practice. It was used to rediscover a full set of equations describing simulations of a turbulent incompressible fluid flow [56], turbulent thermal convection [119], and magnetohydrodynamic turbulence [51]. Additionally, it has discovered novel models describing experimental systems with quasi-two-dimensional fluid flow [98] and active nematic turbulence [50].

Nevertheless, there remain critical challenges that have prevented the widespread adoption of SPIDER as an automatic model discovery tool, which this dissertation aims to resolve. First, the existing descriptions of SPIDER—indeed, all prior sparse regression literature—do not furnish a principled and automatic procedure for constructing or manipulating terms and libraries. Not only does library generation involve arbitrary choices on what exactly to include that vary by user, but more problematically, the entire burden of enumerating all distinct library terms and encoding the rules for evaluating them is placed on the user, requiring laborious and error-prone mathematical and programming work that, at a minimum, necessitates a high degree of familiarity with tensor calculus and the library specification. Second, although prior

work suggests sparse regression algorithms for identifying individual equations, there is no existing algorithm that is capable of systematically discovering *all* approximate relations contained within a given library. The present work will explain how these processes can be almost entirely automated using systematic symbolic algorithms, as has been done in the open-source SPIDER implementation `PySPIDER`. For the second task, it is even possible to algorithmically differentiate between fundamental equations and equations that are trivially implied by a simpler, already discovered relation. The backbone of these algorithms is a basic special-purpose computer algebra system for manipulating tensor calculus expressions.

The other focus of this dissertation is on extending SPIDER to learning coarse-grained descriptions of many-body systems (MBS) involving large numbers of interacting particles or agents. Standard examples of such systems include molecular dynamics [46], which is typically studied by simulating anywhere from 10^3 to over 10^9 [95] point particles or molecules, and active matter, such as bird flocks [4, 25], biological cell migration [115], and non-living motile colloids [19, 18]. (To avoid cumbersome or confusing language, we will generally anchor our discussion on one representative example, where the MBS is a gas of particles.) Traditionally, MBS have been examined through analytical approximations, such as mean-field theory, which average over the unmodeled microscopic degrees of freedom relating to the individual particles. In the case of thermodynamic systems described by the Boltzmann equation, considering the interrelated hierarchy of equations describing the evolution of moments of the velocity leads to an approximate continuum hydrodynamic description. For instance, the Chapman-Enskog theory [26] relies on an asymptotic expansion modeling small fluctuations in systems near thermal equilibrium. Such theories are limited by their reliance on prior knowledge of the form of the microscopic interactions between particles and make predictions that are accurate only under specific assumptions (such as small deviations from equilibrium, short collision times, high

number density, etc.). Another approach stemming from active matter instead starts by hypothesizing a simple hydrodynamic PDE model featuring a set of terms that are constrained by symmetry considerations [112, 113]; the coefficients of these terms are fitted to approximate the observed qualitative macroscopic behavior. Although this approach was originally not conceived as a data-driven method, it has clear parallels with regression.

More recently, there has been a growing body of research aiming to learn physics-informed models of MBS from data. In Ref. [17], the Boltzmann equation is taken as a starting point, with the collision operator modeled non-parametrically. In contrast, the majority of hybrid methods rely on symbolic or sparse regression, which has been applied to modeling microscopic dynamics with Langevin equations [16, 24, 84], learning Boltzmann-like evolution equations for the probability density function [3], and learning hydrodynamic PDEs [2, 77, 125, 109, 76]. The initial efforts in learning coarse-grained PDEs for MBS have been limited in much the same ways as earlier sparse regression efforts for continuous systems: they lack a principled method of library generation and fail to fully take advantage of the natural tensor structure of both the observed data and final learned PDEs. In contrast, the direct generalization of SPIDER for MBS that are presented in this dissertation extends continuous SPIDER's solutions to these problems. Although the particle data are fundamentally discrete and thus not immediately differentiable, we use a kernel convolution-based approach to convert particle data into smooth continuous fields, resulting in a model consisting of integro-differential equations which can be learned using the same exact methods.

This manuscript is organized as follows. Chapter 2 formalizes the structure of the symmetry-equivariant tensor libraries to be used in SPIDER, including a symbolic language representing a particular class of mean-field models. Chapter 3 describes symbolic algorithms for generating, evaluating, and performing logical deduction on

these libraries. Chapter 4 discusses various ingredients in a regression algorithm that allows for more robust discovery of all physically meaningful equations. Finally, Chapter 5 illustrates SPIDER as applied to two simple simulated systems, representing a continuous incompressible fluid flow and a compressible fluid modeled using discrete interacting atoms.

Chapter 2

Symmetry-Equivariant Tensor Libraries

First, Section 2.1 will review the definitions from the representation theory of groups that are essential to SPIDER. In Sections 2.2 and 2.3, we will then establish the systematic specification of the libraries used in SPIDER for problems where the data are fundamentally continuous. Section 2.4 will focus on extending this construction to the case of many-body systems, where the measurements are localized to discrete particles, requiring the use of distinct but related *discrete libraries*.

2.1 Review of Group Representations

In order to formulate symmetry-equivariant equations describing a physical system of interest, it is important to understand how its symmetry group will act on each term appearing in these equations and which spaces of terms remain invariant under this action. Rather than considering the elements of the group in abstract, it is generally simpler to treat them as linear operators acting on the terms. This notion is formalized by group representation theory. In particular, as we will see in Section 2.2, in SPIDER, we are careful to choose symbolic libraries that correspond with a set of

irreducible representations, for a particular symmetry group (e.g., rotations in three dimensions): this allows us to decompose the equation search space into its smallest components that respect the symmetries of the system. In this section, we summarize the relevant concepts from representation theory [116].

Let G and V be a group and a vector space, respectively.

Definition 2.1.1 (Group Representation)**.** A group of linear transformations $U(G)$ on V is said to be a *representation of G on V* if there exists a group homomorphism $U : G \to U(G)$.

For instance, the group of $n \times n$ rotation matrices is the standard example of a representation of the symmetry group of rotations $\mathrm{SO}(n)$ on the vector space $V = \mathbb{R}^n$. For a given rotation $r \in \mathrm{SO}(n)$ and any vector $v \in V$, $U(r)$ is the matrix with entries ensuring that action by left matrix multiplication

$$r \circ v = U(r)v \tag{2.1}$$

generates the corresponding rotated copy of the vector v. Similarly, the orthogonal matrices form a representation of the orthogonal group $\mathrm{O}(n)$ on V: multiplying a vector by an orthogonal matrix leads to an orthogonal transformation of the vector.

The action of rotations is commonly considered not only on the space of vectors, but also scalars ($V = \mathbb{R}$), rank-2 tensors ($V = \mathbb{R}^{n \times n}$), and so on. Note that U cannot be defined in isolation from V, so it makes sense to think of representations as tuples (U, V). We are also interested in the effect of U on subspaces of V:

Definition 2.1.2 (Invariant Subspace)**.** A subspace W of V is said to be an *invariant subspace* of V with respect to $U(G)$ if $U(g)W \subseteq W$ for all $g \in G$.

Definition 2.1.3 (Irreducible Representation)**.** A representation $U(G)$ on V is called *irreducible* if the only subspaces of V that are invariant with respect to $U(G)$ are V and the trivial subspace $\{0\}$. Otherwise, it is called *reducible*.

Returning to our illustrative example of $\mathrm{O}(n)$, while the representation of orthogonal transformations as orthogonal matrices is irreducible on the space of n-dimensional vectors, $\mathbb{R}^n$, this is no longer true for the space of second-rank tensors $V = \mathbb{R}^{n\times n}$. To see this, let us define the action of g on second-rank tensors $A \in V$ as the following linear transformation on V:

$$g \circ A = R(g)AR(g)^{-1}, \tag{2.2}$$

where R is the aforementioned representation of $\mathrm{O}(n)$ on $\mathbb{R}^n$. This is no longer an irreducible representation on the space of second-rank tensors: for example, the subspace $W \subset V$ of antisymmetric $n \times n$ matrices is invariant under all such transformations.

We ultimately aim to decompose reducible representations into direct sums of representations that *are* irreducible. The following definition is particularly useful:

Definition 2.1.4 (Subrepresentations)**.** A subrepresentation of a representation $U(G)$ on V is a representation $U'(G)$ on a subspace W of V such that $U'(g)$ and the restriction of $U(g)$ to W are equal for all $g \in G$.

What we would like to identify is the non-trivial irreducible subrepresentations of our original reducible representation on the whole space V.

For the purposes of SPIDER, the exact details of what are the representations of the symmetry group matter less than the irreducible *subrepresentations* and, even more so, the involved invariant subspaces of V. As in Ref. [52], **we will abuse terminology and refer interchangeably to an irreducible subrepresentation (U') and the invariant subspace on which it is defined (W)**. By extension, we will generally refer to the "nth rank irreducible representations" of the symmetry group, implicitly meaning the subspaces of the space of nth rank tensors that are invariant under these irreducible subrepresentations.

The importance of the above discussion is easily summarized. For many of the symmetry groups under which the physical systems of interest may be invariant (such as O(n), SO(n), the group of Lorentz transformations, etc.), the irreducible representations of a given rank

1. are the smallest nontrivial subspaces invariant under the action of the symmetry group of the system, i.e., library terms remain in the irreducible representation under any symmetry transformation, and

2. have direct sum equal to the entire vector space, as the Lie group in question possesses the *complete reducibility property* [58].

For example, the second-rank irreducible representations of O(n) are [116]

1. the subspace of matrices proportional to the identity matrix I_n,

2. the subspace of antisymmetric $n \times n$ matrices,

3. and the subspace of trace-free symmetric $n \times n$ matrices.

Moreover, an arbitrary $n \times n$ matrix can be written as the sum of matrices from each of the three subspaces. We will make extensive use of these irreducible representations in the next section, separately learning second-rank equations in each of the subspaces.

2.2 Notation and Libraries for Continuous Data

Before explaining how SPIDER's libraries for continuous data are constructed, let us establish a consistent set of notations and conventions for clarity. To maximize succinctness of presentation, unless otherwise noted, we restrict ourselves to the most common type of problem, namely learning equations for spatiotemporal systems with a Euclidean, non-relativistic geometry and a global symmetry group

$\mathrm{E}(n) = \mathrm{T}(n) \rtimes \mathrm{O}(n)$: that is, full translational and orthogonal (rotation and/or reflection) symmetry. However, note that the conceptual machinery of SPIDER is able to handle other geometries and symmetry groups, generally requiring only straightforward modifications to the algorithm, as explained later in this section. This manuscript generally adheres to the notation of Ricci calculus [110]. Greek subscripts $\alpha, \beta, \gamma, \dots$ (or rarely the Latin subscripts $i, j, k, \dots$) denote abstract tensor indices, whereas Latin subscripts x, y, z correspond to concrete spatial indices of a tensor (referring to a specific entry in the coordinate representation used to express the input data). However, all superscripts will generally refer to exponentiation, since, for the symmetry group $\mathrm{E}(n)$, it is sufficient to consider fully covariant tensors. This is because the spaces of tensors with contravariant indices are isomorphic to the fully covariant tensor spaces, subject to multiplying the partly contravariant tensor by an appropriate number of copies of the metric tensor $g_{\alpha\beta}$ [52]. We use the Einstein summation notation where summation over a repeated index of a tensor is implicit, for instance:

$$A_{\alpha\alpha} = A_{xx} + A_{yy} \, (+ A_{zz} + \dots) \tag{2.3}$$

We call the act of assigning the same value to two distinct abstract indices in a tensor (creating a summation over this new repeated index) *contraction*. This notation allows for equally transparent and unambiguous representations for common simple tensor operations such as $\boldsymbol{u} \cdot \boldsymbol{v} = u_\alpha v_\alpha$ or $(\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = u_\beta \partial_\beta u_\alpha$ as well as complex ones without a clear standard notation, such as $A_{\alpha\beta}\partial_\beta v_\alpha$. Finally, we use the index set notation $[n] = \{1, 2, \dots, n\}$.

The *dataset* is a set of functions (which we will generally refer to as *observables*) with domain $\Omega = X \times [0, T] \subset \mathbb{R}^n \times \mathbb{R}$ and a codomain of some vector space V, which represent measurements of relevant physical quantities on the spatial domain X at various position-time pairs $(\boldsymbol{x}, t)$. The resolution of the data is limited in reality, so the observables can be sampled at only a finite number of points, generally assumed to

be a uniformly spaced grid, which is both convenient and allows for better numerical convergence [57, 83]. (In principle, SPIDER can even learn from data samples from multiple realizations of the same experiment, at different times or even on different domains – this is a consequence of the global symmetry assumption that is rarely exploited in practice.) Specifically, V will be the space of tensors of a given rank or an invariant subspace with respect to the symmetry group. For example, in the case of scalar- or vector-valued observables, like pressure or velocity, respectively, we have $V = \mathbb{R}$ or $\mathbb{R}^n$. Observables that are second-rank tensors (for example, a strain-rate tensor) may be taken from the space $V = \mathbb{R}^{n\times n}$ of all such tensors, or they may already belong to one of the second-rank irreducible representations of the rotational symmetry group $\mathrm{O}(n)$: namely, the set of antisymmetric or the set of symmetric trace-free tensors. (Note that there is no such splitting possible for ranks 0 and 1, since they have no non-trivial irreducible representations.) Finally, the symmetry-equivariant differential operators are simply the time derivative ∂_t and the gradient, which can be treated as a rank-1 covariant operator and that we will notate as ∇ or ∂_α. While the notions of covariant derivative ∇ and partial derivative ∂_α may not coincide in non-Euclidean geometry, these are completely equivalent for Euclidean problems and differ only by a factor of the metric tensor $g_{\alpha\beta}$ otherwise. Repeated applications of the gradient k times are notated by ∇^k, increase the rank of the derivand tensor by k, and should not be confused with the Laplacian, which we can write as $\partial_\alpha\partial_\alpha$, and so on.

We impose stricter symmetry restrictions on the libraries $\mathcal{L}$ and consequently the library terms f that constitute them: not only are the terms tensors, but it is most advantageous to choose a set of libraries in bijection with the set of irreducible representations. This is because, as we saw in Section 2.1, the irreducible representations of a given rank comprise the finest splitting of the space of tensors of that rank into a direct sum of invariant subspaces under the symmetry group. Hence, splitting all

tensors among the irreducible representations gives rise to the greatest number of potential independent relations while still fully describing the relations between arbitrary tensors. Although it is possible to consider data and equations of rank higher than 2 by dealing with higher-rank irreducible representations [116], the vast majority of physical theories can be formulated using tensors of rank up to 2 only and so these will not be examined in the present work.

Although we will not attempt to fully formalize the algebraic structure of the libraries, it turns out that the starting point for constructing the libraries can be expressed using the most elementary notions from differential algebra [66]. Given a set of observables $Y = \{y_1, \ldots, y_m\}$ and the (commuting) differentials ∂_t and ∇, the ring of differential polynomials in Y with real coefficients and derivations $\Delta = \{\partial_t, \nabla\}$ is notated by $\mathbb{R}\{Y\}_\Delta$ and consists of the set of polynomials in the variables

$$\left\{ \partial_t^i \nabla^j y_k \,\middle|\, i, j \in \mathbb{Z}_{\geq 0},\, k \in [m] \right\}. \tag{2.4}$$

We refer to the monomials $\partial_t^i \nabla^j y_k$ as *primes* since they cannot be expressed as products of smaller monomials (indeed, they are primes of the polynomial ring according to the algebraic definition of a prime). For instance, the only primes used in constructing the toy library (1.4) are $\boldsymbol{u}$, $\partial_t \boldsymbol{u}$, and $\nabla \boldsymbol{u}$.

Now, consider an arbitrary expression formed by only elementary operations: namely, some combination of the observables, the differential operators, addition, and multiplication. By the product rule of differentiation and the distributivity of multiplication, any of these expressions may be written as a sum of products of primes. Thus, any linear combination of such expressions can be rewritten as a linear combination of products of primes. A product of primes is a full-rank tensor in the sense that no abstract index is repeated, and lower-rank tensors may be formed by successively applying contractions to this tensor. This suggests a natural procedure

for constructing tensors that may appear in a library: we will call any contraction formed from a finite product of primes a term. Terms are exactly the monomials in the differential polynomial ring $\mathbb{R}\{Y\}_\Delta$. (Note: the product of no primes, 1, is generally not included in the scalar library, although it is not explicitly forbidden by symmetry considerations.) As an example, the terms in the toy library (1.4) are each contracted products of up to two primes and could also be written as follows:

$$\mathcal{L} = \{u_\alpha, \partial_t u_\alpha, u_\alpha \cdot \partial_\beta u_\beta, u_\beta \cdot \partial_\beta u_\alpha, u_\beta \cdot \partial_\alpha u_\beta\}. \tag{2.5}$$

Before considering the issue of splitting into irreducible representations, let us see how to construct a complete unsplit library (which is already sufficient for ranks 0 and 1). The terms should be not only tensors of the appropriate rank but also not too complex. To formalize this concept, we define the *complexity* $\mathcal{C}[f]$ of a term f as follows. We take a scalar or vector observable to have unit complexity, and a higher-rank observable, a complexity equal to its rank. Next, the complexity of a prime is the sum of the order of its derivative and the observable:

$$\mathcal{C}\left[\partial_t^i \nabla^j y\right] = i + j + \mathcal{C}[y]. \tag{2.6}$$

To define the complexity of a full term, we take complexity to be additive for products of primes:

$$\mathcal{C}\left[\prod_{i=1}^{n} p_i\right] = \sum_{i=1}^{n} \mathcal{C}[p_i]. \tag{2.7}$$

Finally, we define complexity as unchanged by contractions. Fixing some positive integer C_{max} as the complexity threshold, we form the unsplit library of rank r by simply enumerating all terms of rank r with complexity up to C_{max}. This means that any equation expressible from the unsplit library will be a contraction of $P = 0$ for some differential polynomial $P \in \mathbb{R}\{Y\}_\Delta$, although conversely most differential

polynomials do not correspond to valid equations since they are not required to all be tensors of the same rank.

As an illustrative example, consider a hydrodynamic dataset with observables of pressure p (rank 0) and velocity $\boldsymbol{u}$ (rank 1). The rank 0, 1, and 2 unsplit libraries with complexity threshold 3 are as follows:

$$\begin{aligned}\mathcal{L}_0 = \{&p, \partial_\alpha^2 p, \partial_t p, \partial_t^2 p, p^2, p \cdot \partial_t p, p^3, \partial_\alpha u_\alpha, \partial_t \partial_\alpha u_\alpha, \\ &\partial_\alpha p \cdot u_\alpha, p \cdot \partial_\alpha u_\alpha, u_\alpha \cdot u_\alpha, u_\alpha \cdot \partial_t u_\alpha, p \cdot u_\alpha \cdot u_\alpha\}\end{aligned} \tag{2.8}$$

$$\begin{aligned}\mathcal{L}_1 = \{&\partial_\alpha p, \partial_t \partial_\alpha p, p \cdot \partial_\alpha p, u_\alpha, \partial_\alpha \partial_\beta u_\beta, \partial_\beta^2 u_\alpha, \partial_t u_\alpha, \\ &\partial_t^2 u_\alpha, p \cdot u_\alpha, \partial_t p \cdot u_\alpha, p \cdot \partial_t u_\alpha, p^2 \cdot u_\alpha, \\ &u_\alpha \cdot \partial_\beta u_\beta, u_\beta \cdot \partial_\beta u_\alpha, u_\beta \cdot \partial_\alpha u_\beta, u_\alpha \cdot u_\beta \cdot u_\beta\}\end{aligned} \tag{2.9}$$

$$\begin{aligned}\mathcal{L}_2 = \{&\partial_\alpha u_\beta, \partial_t \partial_\alpha u_\beta, \partial_\alpha p \cdot u_\beta, p \cdot \partial_\alpha u_\beta, \\ &u_\alpha \cdot \partial_t u_\beta, \partial_\alpha \partial_\beta p, u_\alpha \cdot u_\beta, p \cdot u_\alpha \cdot u_\beta\}\end{aligned} \tag{2.10}$$

In particular, (2.10) displays an important nuance: when the library contains a term $t_{\alpha\beta}$, its transpose $t_{\beta\alpha}$ does not also appear. Instead, *all* relations connecting tensors and their versions with permuted free indices are formed by projecting onto the irreducible representations.

These libraries are sufficient to express the governing equations of an incompressible fluid flow: the incompressibility condition

$$\partial_\alpha u_\alpha = 0 \tag{2.11}$$

which includes just a single complexity-2 term, and the Navier-Stokes equation

$$\partial_t u_\alpha + u_\beta \cdot \partial_\beta u_\alpha + \partial_\alpha p - \nu \partial_\beta^2 u_\alpha = 0 \tag{2.12}$$

which uses terms of complexity no higher than 3. Thus, the compact libraries (2.8) and (2.9), respectively containing 14 and 16 terms, are sufficient to represent a full set of equations describing the incompressible fluid. Additional well-known identities also hold for such a flow, such as the pressure-Poisson equation

$$\partial_\alpha^2 p + \partial_\alpha(u_\beta \partial_\beta u_\alpha) = 0 \tag{2.13}$$

and the energy equation

$$\partial_t(u_\alpha u_\alpha) + \partial_\alpha(u_\alpha u_\beta u_\beta) + u_\alpha \partial_\alpha p - \nu \partial_\alpha^2(u_\beta u_\beta) + \nu(\partial_\alpha u_\beta \cdot \partial_\alpha u_\beta) = 0, \tag{2.14}$$

each of which requires the complexity-4 library to represent. However, these are not algebraically independent from equations (2.11) and (2.12): in fact, after expanding and applying (2.11), they simplify to the divergence of (2.12) and the product of (2.12) with u_α, respectively. In Chapter 3, we will see how to identify *all* such non-fundamental relations in addition to the simpler fundamental equations that imply them.

Table 2.1: Sizes of unsplit hydrodynamic libraries with various combinations of rank and complexity threshold.

Rank	$\mathcal{C} \leq 3$	$\mathcal{C} \leq 4$	$\mathcal{C} \leq 5$
0	14	43	114
1	16	51	170
2	8	38	132

Table 2.1 lists the numbers of terms in the scalar, vector, and rank-2 unsplit libraries for complexity thresholds ranging from 3 to 5. To put the modest size of these libraries in perspective, consider the brute-force libraries used in SINDy, which do not take symmetry considerations into account. In Ref. [99], the (physically equivalent) equation describing the time evolution of the vorticity $\omega = \nabla \times \boldsymbol{u}$ was identified using a brute-force library including $\partial_t \omega$ and "polynomial terms of vorticity

and all velocity components up to second degree, multiplied by derivatives of the vorticity up to second order" – in three dimensions, such a library would contain 101 terms, more than a factor of six larger than our more physically comprehensive complexity-3 vector library.

Occasionally, it may be useful to further constrain the libraries used in SPIDER, for instance, by restricting the number of primes (in other words, copies of observables in the product) or the number of derivatives in the term. However, such constraints are generally useful only if there is specific reason to believe certain classes of terms are unphysical, as they have little effect on the size of the library.

In order to split the second-rank library into irreducible representations, recall that a second-rank term $E_{\alpha\beta}$ is a sum of three projected parts, each contained in a separate second-rank irreducible representation of $\mathrm{O}(n)$: its trace times the identity matrix (i.e. the Kronecker delta)

$$e = \mathrm{tr}(E)\delta_{\alpha\beta} = E_{\gamma\gamma}\delta_{\alpha\beta}, \tag{2.15}$$

its antisymmetric part

$$W_{\alpha\beta} = [\mathrm{AS}(E)]_{\alpha\beta} = \frac{1}{2}(E_{\alpha\beta} - E_{\beta\alpha}), \tag{2.16}$$

and its symmetric trace-free part

$$A_{\alpha\beta} = [\mathrm{STF}(E)]_{\alpha\beta} = \frac{1}{2}(E_{\alpha\beta} + E_{\beta\alpha}) - \frac{1}{n}e. \tag{2.17}$$

Since the trace results in a set of terms proportional to the scalar (rank 0) library, only the antisymmetric and symmetric trace-free parts merit independent interest. (This is also the reason that terms proportional to the Kronecker delta $\delta_{\alpha\beta}$ need not be included in the second-rank library.) It is nearly sufficient to directly apply

the projection operators AS and STF to each term in the rank-2 library, except some terms, such as $u_\alpha u_\beta$ or $\partial_\alpha \partial_\beta p$, are always symmetric. (If the dataset includes a second-rank observable which is explicitly defined as antisymmetric, it is also possible for a library term to be identically antisymmetric.) These terms may be dropped immediately from the library corresponding to the opposite irreducible representation, since their projection onto that irreducible representation is identically zero. For instance, the second-rank antisymmetric library of complexity 3 for the illustrative example above consists of the antisymmetric parts of

$$\mathcal{L}_{2,\mathrm{AS}} = \{\partial_\alpha u_\beta, \partial_t \partial_\alpha u_\beta, \partial_\alpha p \cdot u_\beta, p \cdot \partial_\alpha u_\beta, u_\alpha \cdot \partial_t u_\beta\} \tag{2.18}$$

and the second-rank symmetric trace-free library comprises the STF projections of every term in $\mathcal{L}_2$ in (2.10).

Finally, we consider modifications to the library construction that allow some of the assumptions stated at the beginning of this section to be relaxed. First, take the case of a general metric tensor $g_{\alpha\beta}$ rather than a Euclidean geometry. As noted in [52], the metric does not enter into the library construction at all and the only necessary changes are implicit: contractions, including the trace, include a factor of $g_{\alpha\beta}$ to account for the fact that we have defined all indices to be covariant, observables may need to be made covariant by multiplying by $g_{\alpha\beta}$, and ∂_α is now the covariant derivative in the geometry of the system. A related point is that, in fact, SPIDER does not require a *global* symmetry of the system: the existence of a local symmetry is sufficient for the definition of symmetry-equivariant tensors, although the coordinate bases may be defined only locally in that case. For instance, this remark shows that SPIDER can be applied directly in the case of the periodic spatial domains often used in numerical simulations. Nevertheless, the assumption of a globally Euclidean

geometry simplifies other constructions in the SPIDER algorithm, such as defining integration domains (see Section 3.3).

Systems that lack a reflection symmetry and have the rotational symmetry group $\mathrm{SO}(n)$ in place of $\mathrm{O}(n)$ have a similar library construction, except pseudotensors (which change sign under an orthogonal transformation with $\det = -1$) are now allowed to be present among the observables, and the list of observables is augmented with a special orientation form: the Levi-Civita symbol $\epsilon_{\alpha\beta\ldots}$ with n indices, which is also a pseudotensor. This allows the representation of pseudotensors of any rank from regular tensors alone, including the most familiar case of pseudovectors, such as the cross product

$$\boldsymbol{u} \times \boldsymbol{v} = \epsilon_{\alpha\beta\gamma} u_\alpha v_\beta \tag{2.19}$$

in three dimensions. However, the Levi-Civita symbol can only appear at most once per library term because of the identity

$$\epsilon_{i_1 i_2 \ldots i_n} \epsilon_{j_1 j_2 \ldots j_n} = \begin{vmatrix} \delta_{i_1 j_1} & \delta_{i_1 j_2} & \ldots & \delta_{i_1 j_n} \\ \delta_{i_2 j_1} & \delta_{i_2 j_2} & \ldots & \delta_{i_2 j_n} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{i_n j_1} & \delta_{i_n j_2} & \ldots & \delta_{i_n j_n} \end{vmatrix}, \tag{2.20}$$

which allows any term with multiple Levi-Civita symbols to be expressed as a sum of contractions of existing terms.

Finally, more complex symmetry groups, such as the Lorentz group of transformations of Minkowski spacetime that represents the symmetry of relativistic systems with negligible gravity [92], admit similar library constructions that are outside the scope of this dissertation. An interested reader may again refer to Ref. [52] for a basic treatment of this topic.

2.3 Symmetry Breaking and Boundary Conditions

A common conceptual confusion surrounding the SPIDER framework is the following: which systems possess the necessary symmetries for SPIDER to be employed? To name a couple of examples where the applicability of SPIDER is less obvious, the shape of the domain may not be rotationally symmetric; the system may be affected by an external force such as gravity or an applied magnetic field pointing in a certain direction, or even a force with varying direction at different locations that does not obey any global symmetry; or the physical state of the system may involve essential anisotropies, such as the alignment of nematic molecules in liquid crystals [37, 69] or stresses induced by force chains in granular media [94].

In fact, none of these cases of (global) symmetry breaking violate the (local) symmetry assumptions made by SPIDER: as previously noted in Section 2.2, SPIDER requires only local symmetry, which is reflected in the fact that the bulk equations governing a system do not depend on its spatial configuration. The other instances above involve asymmetry in the *data*, not physical laws. Namely, a known external forcing can be input as another observable; similarly, data describing the anisotropies at each point in the domain, e.g., the nematic orientation or stress tensor, are essential to a faithful description of such anisotropic systems. (Such data often most naturally admit a representation as second-rank tensors, in which case SPIDER's ability to model second-rank equations is essential.)

If direct measurements of these symmetry-breaking effects are inaccessible, the apparent equations describing the system's dynamics may not be symmetric; however, this is inherently a symptom of working with incomplete data, not the symmetry-breaking itself. The presence of unknown effects presents a substantial challenge for all interpretable model discovery, except in special cases where they can be eliminated through a specific averaging procedure [97, 98].

In contrast, a case where symmetry breaking cannot be ignored in library construction is the discovery of boundary conditions rather than bulk equations. Near the boundary of the system, the rotational symmetry is partially broken: the problem is only invariant with respect to rotations about the normal vector $\boldsymbol{n}$, rather than all spatial rotations. Thus, the symmetry group describing boundary conditions is $\mathrm{O}(n-1) \times \{1\} \simeq \mathrm{O}(n-1)$. Nevertheless, SPIDER can still be used to identify the boundary conditions. The set of observables that transform as vectors near the boundary must be augmented by $\boldsymbol{n}$ (which satisfies the identities $n_\alpha n_\alpha = 1$ and $\partial_t n_\alpha = \partial_\beta n_\alpha = 0$), and then the resulting libraries of rank 1 or higher must be split via application of the normal and tangential projection operators $P_\perp$ and $P_\parallel$, which satisfy

$$P_\perp(y_\alpha) = n_\alpha n_\beta y_\beta \tag{2.21}$$

and

$$P_\parallel(y_\alpha) = (\delta_{\alpha\beta} - n_\alpha n_\beta) y_\beta = y_\alpha - P_\perp(y_\alpha). \tag{2.22}$$

Thus, the vector library would simply be projected onto its normal and tangential components, which correspond to the scalar and vector irreducible representations of the reduced symmetry group $O(n-1)$.

For illustration, consider again the hydrodynamic problem considered in Section 2.2. The complexity-2 scalar library is

$$\mathcal{L}_{0,\boldsymbol{n}} = \{1, p, \partial_t p, p^2, \partial_\alpha u_\alpha, n_\alpha \cdot u_\alpha, u_\alpha \cdot u_\alpha\} \tag{2.23}$$

(where the 1 term arises through evaluating $n_\alpha n_\alpha$). In turn, the complexity-2 vector library prior to splitting is given by

$$\mathcal{L}_{1,\boldsymbol{n}} = \{n_\alpha, \partial_\alpha p, n_\alpha p, u_\alpha, \partial_t u_\alpha, p u_\alpha\} \tag{2.24}$$

Then, the normal vector library is

$$\mathcal{L}_\perp = \{n_\alpha, n_\alpha(n_\beta \cdot \partial_\beta p), n_\alpha p, n_\alpha(n_\beta \cdot u_\beta), n_\alpha(n_\beta \cdot \partial_t u_\beta), n_\alpha(p n_\beta \cdot u_\beta)\} \tag{2.25}$$

and the tangential vector library is

$$\mathcal{L}_\parallel = \{P_\parallel \partial_\alpha p, P_\parallel u_\alpha, P_\parallel \partial_t u_\alpha, P_\parallel (p \cdot u_\alpha)\}. \tag{2.26}$$

Observe that the normal vector library simply consists of scalars of complexity no greater than 3 multiplied by a factor of n_α, so it encodes no further information compared to the complexity-3 scalar library. Similarly, the second-rank library consists of three parts: these correspond to normal projections along both free indices, one normal projection and one tangential projections, and tangential projections along both indices. Respectively, these can be identified with scalars, $(n-1)$-dimensional vectors, and $[(n-1)\times(n-1)]$-dimensional second-rank tensors. However, just as the normal vector library for boundary conditions is symbolically equivalent to the scalar library, the first two parts are equivalent to the scalar and tangential vector libraries. Only the last projection is independent of the lower-rank boundary condition libraries! **In general, it is sufficient to consider only the tangential boundary condition libraries of each rank.** As in the previous section, we may further split the second-rank library into antisymmetric and symmetric trace-free parts.

Due to the introduction of the normal vector $\boldsymbol{n}$, the boundary condition libraries are larger than their direct counterparts for bulk equations. Moreover, they are more difficult to evaluate in a numerically well-conditioned way. While the bulk equations are evaluated in weak form via integrating over $(n+1)$-dimensional spatiotemporal subdomains, boundary conditions must include data only near the boundary and thus require essentially n-dimensional subdomains that are extremely narrow in the normal direction (see Section 3.3 for a detailed discussion of library evaluation). Thus,

although almost all derivatives in bulk equations can be eliminated using integration by parts, derivatives along the normal direction in the boundary conditions generally cannot be.

Fortunately, boundary conditions are usually expressible as simpler relations compared to bulk equations, in terms of both complexity and the highest derivative order that needs to be evaluated. This is for two reasons. First, the most common types of boundary conditions have extremely simple representations in the boundary condition libraries. These include Dirichlet boundary conditions $u_\alpha = 0$ or their normal/tangential projections, Neumann boundary conditions $n_\alpha \partial_\alpha u_\beta = 0$, as well as Cauchy and Robin boundary conditions, which are intersections and linear combinations of Dirichlet- and Neumann-type boundary conditions, respectively. Second, even for more complex boundary conditions, the bulk equations can be discovered independently and then serve as identities which are used to further simplify the form of the boundary conditions. For instance, if evolution equations for the observables are already known, then time derivatives no longer need to be included in the boundary condition libraries as they can be eliminated via the bulk relations.

2.4 Mean-Field Models for Many-Body Systems

Earlier in this chapter, we considered models for systems described by continuous fields, such as fluid flows with a continuum approximation. We now consider constructing mean-field models for systems that are fundamentally discrete in nature, such as the motion of interacting atoms in a gas or flocks of birds. We refer to such systems as many-body systems (MBS). Let us start by reviewing existing systematic first-principles approaches for discovering mean-field models of MBS that are grounded in kinetic theory [104]. The most common way to derive hydrodynamic equations for MBS without internal degrees of freedom is by considering moments of

the single-particle probability distribution function $f(\boldsymbol{x}, \boldsymbol{v}, t)$, described by the Boltzmann equation:

$$\frac{\partial f}{\partial t} + \boldsymbol{v} \cdot \frac{\partial f}{\partial \boldsymbol{x}} + \frac{\boldsymbol{F}}{m} \cdot \frac{\partial f}{\partial \boldsymbol{v}} = \hat{C}[f], \tag{2.27}$$

where the third term represents an external forcing F on the particles, which have mass m. The collision term $\hat{C}(f)$ is an integral operator quadratic in f, which causes this equation to be difficult to analyze. As shown by Chapman and Enskog [26], this operator simplifies considerably when the system is in thermodynamic equilibrium, in which case the velocity dependence of f is given by the Maxwell distribution. Chapman-Enskog theory expresses the solution to the Boltzmann equation as a perturbative expansion in the Knudsen number Kn, which is the ratio of the molecular mean-free path and a characteristic length scale of the system. This expansion leads to a closed system of equations for a small number of velocity moments of f:

$$\int f \, d\boldsymbol{v}, \quad \int f \boldsymbol{v} \, d\boldsymbol{v}, \quad \int f |\boldsymbol{v}|^2 \, d\boldsymbol{v}. \tag{2.28}$$

To first order in Kn, this recovers the Navier-Stokes equation. Retaining higher orders leads to corrections such as the Burnett equations [21]; however, the Chapman-Enskog expansion is asymptotic and diverges for larger values of Kn corresponding to MBS of low density, so these higher order corrections rarely lead to a more accurate description.

For systems that deviate from thermodynamic equilibrium, the assumptions of Chapman-Enskog theory no longer hold, and a more general approach is necessary. A notable example is the moment method introduced by Grad [54], which considers a more comprehensive set of moments, which include higher-rank tensors. Among other things, this expanded set of moments is necessary in order to allow for nonzero heat flux. Grad considered only the moments corresponding to integrals of f, fv_α, $fv_\alpha v_\beta$, and $fv_\alpha v_\beta v_\beta$. Levermore [70] showed that this choice cannot lead to a closed,

Galilean-invariant system of equations, and at a minimum, one must also consider the integral of $f|\boldsymbol{v}|^4$, leading to a set of 14 scalar degrees of freedom. It should be emphasized that both of these methods still yield closed models only near thermal equilibrium, an assumption that is usually invalid at larger Knudsen numbers. Obtaining closed models in other dynamical regimes requires additional modifications and may necessitate introducing an even more comprehensive set of moments.

To address this deficiency, there has been a recent proliferation of data-driven approaches to learning hydrodynamic equations of MBS. In Ref. [59], an NN-based approach was introduced for learning generalized moments that are not polynomial in $\boldsymbol{v}$ using Galilean-invariant autoencoders. While this method showed promising results for multiple models based on the Boltzmann equation, it still relies on concrete assumptions about the governing equations of the microscopic dynamics and learns black-box representations for the moments that are not easily interpretable. In contrast, most work on learning PDE models for MBS from data has employed interpretable methods based on symbolic [125, 76] or sparse regression [2, 77, 109]. The primary weaknesses of these latter approaches are

1. lack of an appropriate, systematic specification of which candidate terms to use in constructing the model and/or

2. evaluation of the terms that is not sufficiently precise to produce highly accurate models in the presence of limited data.

For instance, none of these methods allow for the discovery of *general* moment equations in the style of Levermore, although notably Ref. [2] explicitly included expressions representing all of Grad's moments in their library. In this section, we will present an extension of the library construction used in the continuous version of SPIDER to the case of hydrodynamic PDEs for MBS, which is more general and

quantitatively accurate than prior approaches and thus presents a more useful tool for modeling a wide variety of MBS.

First, let us clarify how exactly we can measure the quantitative correctness of PDE models for MBS. While the domain of the PDEs is the entire spatiotemporal domain Ω, the actual data g_E are available only along the trajectories of the particles, that is, on a closed subset E that is a union of N curves. These curves are at a minimum continuous but can often be taken to be several times continuously differentiable, and $E \cap (X \times \{t\})$ is the union of N distinct points at any time $t \in [0, T]$. What it means for the PDEs to approximate the system is that they make accurate and specific predictions about its state. Ideally, we would like the continuous functions g modeled by the PDEs to meet two objectives:

Objective 1: approximately satisfy the PDEs at all points in X and

Objective 2: agree closely with g_E at all points in E.

There are multiple compelling options for transforming the discrete measurements g_E into globally defined fields g. One approach is to search for exact smooth extensions of the observables such that

$$g|_E = g_E. \tag{2.29}$$

The existence of such extensions has been known since 1934 by Whitney's extension theorem [120, 121, 122]. More recent work by Charles Fefferman established the existence of bounded linear operators T such that $g = Tg_E$ satisfies (2.29) and g is a function with m derivatives that each have small Lipschitz norm [39], with m derivatives that each have small modulus of continuity [40], and with small norm in the Sobolev space $L^{m,p}(\mathbb{R}^n)$ [41], as well as efficient algorithms for the computation of g when E is a finite set [42] (as is the case if we separately solve the extension problem at each time t that data are sampled). However, it is not clear that such an exact extension presents a good solution in the mean-field sense. In most examples of

MBS, collisions between particles are frequent, resulting in many instances of nearly intersecting trajectories that have significantly differing values of the particle velocity. Even though such events do not lead to singular behavior, an exact extension g may have arbitrarily rapid variation in a small neighborhood of the colliding particles. Thus, it is unclear to what degree this extension would accurately model the locally averaged dynamics. Furthermore, exact extension has a much higher computational cost than alternative approaches relying on local averaging.

In practice, the second objective stated above is unrealistic, since it requires g to be specified accurately on a spatial resolution comparable to the mean inter-particle spacing, which all but necessitates modeling the individual interactions, precisely the task we meant to avoid in the first place. Instead, we aim for a weaker version:

Objective 2 (modified): At any point $(\boldsymbol{x}, t)$, g agrees closely with g_E *averaged over the nearby trajectories in space.*

While the first objective can be expressed rigorously enough, this new second objective is inherently informal, as it depends on the particular averaging method used. Extreme smoothing, such as defining g as the average of g_E over *all* trajectories independently of x, clearly lacks any practical utility: the resulting predictions may be internally self-consistent but cannot be specific at all. However, coarse-graining over a finite length scale can lead to a reasonable model. The simplest way to perform this coarse-graining is through a simple binning approach, in which the spatial domain is divided into a grid of cells and each observable and its derivatives are defined per cell by averaging over all particles in that cell (for instance, see Refs. [125, 76]). This procedure is very sensible (and computationally efficient) when the particle density is extremely high, since this allows for a fine cell partition of the domain such that each cell still contains enough particles to compute the averages with low uncertainty. However, for lower particle densities, the cells must be quite large, meaning that the resolution on which the estimates of the observables are available is very limited; too

much information is discarded by effectively snapping the position of each particle to the nearest point on a coarse grid, making it difficult to evaluate quantities such as gradients of the observables precisely and with high resolution.

We will instead express g as the convolution of g_E with a sufficiently smooth spatial averaging kernel W_h. We aim for the length scale of this kernel, h, to be several times larger than the mean inter-particle spacing, so that the averaged dynamics includes the contributions of many particles at each point; additionally, we would like h to be much smaller than the correlation length L_c of the mean-field dynamics, so that the averaging does not oversmooth information.

Let $\boldsymbol{x}_i(t)$, $i \in [N]$ denote the position of each particle in the system at time t and

$$\mu_t(\boldsymbol{x}) = \sum_{i=1}^{N} \delta(\boldsymbol{x} - \boldsymbol{x}_i(t)) \tag{2.30}$$

be the corresponding empirical measure. Similarly, $y_i(t)$ denotes the value of the observable y for particle i at time t. Also, let $W_h(\boldsymbol{r}) : \mathbb{R}^n \to \mathbb{R}$ be a radially symmetric kernel with finite moments of all orders, satisfying the unit norm condition

$$\int_{\mathbb{R}^n} W_h(\boldsymbol{r})\, d\boldsymbol{r} = 1 \tag{2.31}$$

and having second moment equal to h^2:

$$\int_{\mathbb{R}^n} |\boldsymbol{r}|^2 W_h(\boldsymbol{r})\, d\boldsymbol{r} = h^2. \tag{2.32}$$

As the simplest example, we can take W_h to be a Gaussian function with variance h^2. For computational efficiency, it is even better to choose a compactly supported

kernel, such as one of the polynomial kernels of the family

$$W_h(\boldsymbol{r}) = \begin{cases} c_h \left(a_h^2 - |\boldsymbol{r}|^2\right)^p, & |\boldsymbol{r}| < a_h \\ 0, & |\boldsymbol{r}| \geq a_h \end{cases} \tag{2.33}$$

for a positive integer p (which should be large enough to ensure that at least the first two derivatives are continuous everywhere, i.e., $p \geq 3$) and constants a_h, c_h chosen to satisfy (2.31) and (2.32). We can now define the coarse-graining operator $\rho[\cdot]$:

$$\rho[y](\boldsymbol{x}, t) = \int W_h(|\boldsymbol{x} - \boldsymbol{x}'|) y(\boldsymbol{x}', t)\, d\mu_t(\boldsymbol{x}') = \sum_{i=1}^{N} W_h(|\boldsymbol{x} - \boldsymbol{x}_i(t)|) y_i(t), \tag{2.34}$$

where y can be an observable or a product of observables. As a special case, we also obtain a coarse-grained density field:

$$\rho(\boldsymbol{x}, t) = \rho[1](\boldsymbol{x}, t) = \sum_{i=1}^{N} W_h(|\boldsymbol{x} - \boldsymbol{x}_i(t)|). \tag{2.35}$$

(However, note that it is often more convenient to rescale the operator $\rho[\cdot]$ by a constant factor equal to the mean number density in order to make its output $O(1)$ – we will remark when our empirical results implicitly include this rescaling.) This approach is reminiscent of smoothed-particle hydrodynamics [48, 75], which is a mesh-free computational method for continuum media based on a similar coarse-graining procedure. It should be noted that, as is often done in smoothed-particle hydrodynamics, we could let the kernel radius h vary over $(\boldsymbol{x}, t)$ so that the kernel is wider in regions with few particles and narrower where the density is large. Although the approach in this section is capable of using a variable bandwidth kernel, for simplicity, we take h to be fixed for a given dataset. For some systems, this may even be the objectively correct choice: if the particle interaction range is characterized by a

particular length scale, then, in order for the kernel to effectively encode interaction terms, it is preferable to take h on the order of that length scale.

It now remains to use this operator $\rho[\cdot]$ to define the construction of libraries for mean-field models. There are again a few seemingly reasonable choices. For instance, following the framework of Toner and Tu [112, 113], one could attempt to bring the mean-field libraries into the same symbolic form as the continuous libraries. Even if we cannot directly observe any other properties of a particle, we have access to at least one observable: the particle velocity vector $\boldsymbol{v} = \partial_t \boldsymbol{x}$. In addition, the scalar density ρ defined via (2.35) should also appear in the libraries. Thus, one could treat ρ and $\boldsymbol{v}$ as if they were continuous observables by defining

$$\boldsymbol{v} = \frac{\rho[\boldsymbol{v}]}{\rho}, \tag{2.36}$$

and then construct library terms as (contracted) products of the primes $\partial_t^i \nabla^j y$ for $y_k \in \{\rho, \boldsymbol{v}\}$. However, this approach encounters both numerical and conceptual obstacles. In regions of very low particle density, the quotient in (2.36) is at best very poorly conditioned if the kernel does not have compact support, and it is completely undefined at points sufficiently far away from all particles if the kernel is compactly supported. There is no straightforward solution to this problem: a well-defined extension of the particle observables cannot exist in regions with no particles. Moreover, the simplistic definition (2.36) erases fundamental thermodynamic distinctions. For instance, we would like to be able to define both a momentum density $\rho[v_\alpha]$ and an energy density $\rho[v_\alpha v_\alpha]$. The energy density need not be small at a given point even if the components of $\rho[v_\alpha]$ are nearly zero. On the other hand, there is no independent way to define these two quantities using only the primes ρ and $\boldsymbol{v}$.

A much better definition is to take the library primes to be derivatives of coarse-grained products of observables. For instance, for systems with only the veloc-

ity observable, the primes have the form $\partial_t^i \nabla^j p$ for p in the hierarchy of tensors $\rho, \rho[v_\alpha], \rho[v_\alpha v_\beta]$, and so on. As in the continuous case, the library terms are simply contractions of the products of primes.

The definition of complexity for continuous terms can easily be adapted to the discrete case as well. With the intuition that ρ can be considered to have complexity 1 like other scalars, we compute the complexity of a *coarse-grained product* (CGP) of observables y_i according to

$$\mathcal{C}\left[\rho\left[\prod_{i=1}^{n} y_i\right]\right] = 1 + \sum_{i=1}^{n} \mathcal{C}[y_i], \tag{2.37}$$

with the special case $\mathcal{C}(\rho) = 1$. As in the continuous case, the complexity of an observable y is given by

$$\mathcal{C}[y] = \min\left(\text{rank}(y), 1\right), \tag{2.38}$$

the complexity of a prime is

$$\mathcal{C}[\partial_t^i \nabla^j p] = i + j + \mathcal{C}[p], \tag{2.39}$$

and complexity is additive for products of primes and unchanged by contraction.

In contrast to the continuous case, it is often useful to constrain the library terms according to other considerations in addition to complexity. For instance, terms which feature high powers of the density ρ generally do not have a clear physical interpretation, except perhaps as high orders in a series representation of the effective local pressure. Moreover, since MBS such as molecular gases often reside in states close to thermodynamic equilibrium, density variations in such systems tend to be small. Many of the accurate relations describing these states take the form of algebraic relations such as

$$\rho^4 + c_3\rho^3 + c_2\rho^2 + c_1\rho = 0, \tag{2.40}$$

which achieve very small residuals on the typical range of densities observed but shed little light on dynamical mechanisms. Limiting the number of appearances of ρ in a term (i.e., the number of primes in the product comprising the term) drastically reduces the number of such equations and allows the meaningful dynamical equations and equations of state to be more readily apparent in the list of relations instead.

For illustration, we list below example scalar, vector, and second-rank unsplit and split libraries for a MBS with only one observable, the velocity vector $\boldsymbol{v}$. The complexity threshold of the libraries is 4 and the maximum number of primes in a term is 2, which are the minimum values of these parameters necessary to discover the Navier-Stokes equation. With these constraints, the unsplit libraries are

$$\begin{aligned}\mathcal{L}_0 = \{&\rho, \rho^2, \rho\cdot\partial_\alpha^2\rho, \rho\cdot\partial_t\rho, \rho\cdot\partial_t^2\rho, \rho\partial_\alpha\rho[v_\alpha], \rho\cdot\rho[v_\alpha\cdot v_\alpha],\\ &\partial_\alpha\rho\cdot\partial_\alpha\rho, \partial_\alpha\rho\cdot\rho[v_\alpha], \partial_\alpha^2\rho, \partial_t\rho, (\partial_t\rho)^2, \partial_t\partial_\alpha^2\rho, \partial_t^2\rho, \partial_t^3\rho,\\ &\rho[v_\alpha]\cdot\rho[v_\alpha], \partial_\alpha\rho[v_\alpha], \partial_t\partial_\alpha\rho[v_\alpha], \rho[v_\alpha\cdot v_\alpha], \partial_t\rho[v_\alpha\cdot v_\alpha]\},\end{aligned} \tag{2.41}$$

$$\begin{aligned}\mathcal{L}_1 = \{&\rho\partial_\alpha\rho, \rho\partial_t\partial_\alpha\rho, \rho\cdot\rho[v_\alpha], \rho\cdot\partial_t\rho[v_\alpha], \partial_\alpha\rho, \partial_\alpha\rho\cdot\partial_t\rho,\\ &\partial_\alpha\partial_\beta^2\rho, \partial_t\rho\cdot\rho[v_\alpha], \partial_t\partial_\alpha\rho, \partial_t^2\partial_\alpha\rho, \rho[v_\alpha], \partial_\alpha\partial_\beta\rho[v_\beta], \partial_\beta^2\rho[v_\alpha],\\ &\partial_t\rho[v_\alpha], \partial_t^2\rho[v_\alpha], \partial_\beta\rho[v_\alpha\cdot v_\beta], \partial_\alpha\rho[v_\beta\cdot v_\beta], \rho[v_\alpha\cdot v_\beta\cdot v_\beta]\},\end{aligned} \tag{2.42}$$

and

$$\begin{aligned}\mathcal{L}_2 = \{&\rho\partial_\alpha\partial_\beta\rho, \rho\partial_\alpha\rho[v_\beta], \rho\cdot\rho[v_\alpha\cdot v_\beta], \partial_\alpha\rho\cdot\partial_\beta\rho, \partial_\alpha\rho\cdot\rho[v_\beta], \partial_\alpha\partial_\beta\rho,\\ &\partial_t\partial_\alpha\partial_\beta\rho, \rho[v_\alpha]\cdot\rho[v_\beta], \partial_\alpha\rho[v_\beta], \partial_t\partial_\alpha\rho[v_\beta], \rho[v_\alpha\cdot v_\beta], \partial_t\rho[v_\alpha\cdot v_\beta]\}.\end{aligned} \tag{2.43}$$

Furthermore, while the symmetric trace-free second-rank library is again the result of applying the STF projection individually to each term, the antisymmetric second-rank library includes the AS projections of only the following 4 terms:

$$\mathcal{L}_{2,AS} = \{\rho\partial_\alpha\rho[v_\beta], \partial_\alpha\rho\cdot\rho[v_\beta], \partial_\alpha\rho[v_\beta], \partial_t\partial_\alpha\rho[v_\beta]\} \tag{2.44}$$

The size of the libraries grows very little even if we allow more primes to be included in a single term: Table 2.2 lists the number of terms in the scalar, vector, and rank-2 unsplit complexity-4 libraries with a limit of 2 to 4 primes per term.

Table 2.2: Sizes of representative complexity-4 unsplit discrete libraries for various combinations of rank and maximum number of primes per term.

Rank	2 primes	3 primes	4 primes
0	20	22	23
1	18	20	20
2	12	12	12

It is also instructive to consider the lowest-complexity libraries capable of representing the closures considered by Levermore in Ref. [70]. Although he refers to moment closures up to fourth rank without contractions, the "14-moment closure" spanned by the moments $\{1, v_\alpha, v_\alpha v_\beta, v_\alpha \cdot v_\beta \cdot v_\beta, v_\alpha \cdot v_\alpha \cdot v_\beta \cdot v_\beta\}$ is sufficient to derive the Navier-Stokes equation from first principles; none of these terms exceed second rank. In our notation, the coarse-grained estimate of each of these quantities, which uses the $\rho[\cdot]$ operator in place of the distribution function f, has complexity no higher than 5 since the highest power of $\boldsymbol{v}$ is quartic. As another example, the advective derivative of $\rho[v_\alpha v_\beta]$ includes no terms of complexity higher than 5 (which is achieved by the advection term $\partial_\gamma \rho[v_\alpha v_\beta v_\gamma]$). It is possible to construct modestly-sized complexity-5 libraries: Table 2.3 lists the number of terms in the scalar, vector, and rank-2 unsplit complexity-5 libraries with a limit of 2 to 5 primes per term.

Table 2.3: Sizes of representative complexity-5 unsplit discrete libraries for various combinations of rank and maximum number of primes per term.

Rank	2 primes	3 primes	4 primes	5 primes
0	39	49	51	52
1	52	58	60	60
2	34	40	40	40

In addition to its essential ability to model closure relations between different moments of the density function ρ, the integro-differential calculus we have introduced

can also successfully encode short-range non-local interaction terms. The convolutional transformation from point particles to smoothed densities allows for terms to represent fields in the sense of field theory; interactions between particles can be represented through multi-prime terms that are nonlinear in ρ, and self-interactions (such as active propulsion or drag from the surrounding medium) of course also fit within this framework.

Chapter 3

Automatic Generation and Manipulation of Symbolic Libraries

Whereas the previous chapter provided a high-level overview of how the libraries in SPIDER are defined, we will now describe the symbolic algorithms required to perform automated reasoning with these libraries in practice as well as the structure of the representations used to create a computer encoding of tensor calculus. These algorithms are essential to making SPIDER usable by the scientific community at large, as manually programming this symbolic logic is an extremely laborious and error-prone task. Our presentation will focus on the continuous libraries, which are described by the slightly simpler continuous symbolic language; however, the minimal modifications necessary to adapt the algorithms to the case of the discrete libraries (discrete language) will be noted as necessary.

3.1 Formal Grammars and Semantics

First, it is important to fully specify the symbolic representations used by SPIDER. Thus, let us describe SPIDER's formal grammar, which we summarize in extended Backus-Naur form in Figure 3.1. The most fundamental type used in SPIDER is an

index, which formalizes the concept of a single index in our index notation. In fact, there are several distinct subtypes of the index type. We call the concrete spatial indices of a tensor *literal indices*, such as x and y in A_{xy}. Any tensor expression with a full set of literal indices can be evaluated in strong form at a given grid point $(\boldsymbol{x}, t)$ or in weak form over an integration domain Ω_k (see Section 3.3). In computer science terminology, such an expression is a ground expression, as it contains no variables. In contrast, the indices of Einstein notation are called *variable indices*. To evaluate an expression with variable indices, they must first be mapped to literal index values. For example, consider the expression $e_\alpha = u_\beta \partial_\beta u_\alpha$. Both α and β are variable indices, though they have different semantic meanings. Here α is a *free variable*, which can be assigned to any literal index value, with each choice corresponding to a different component of the vector $\boldsymbol{e}$. On the other hand, since β is a repeated index, it represents a *bound variable*, which must be summed over *all* possible literal indices when evaluating any component of $\boldsymbol{e}$. (In contrast to computer science literature, where variable binding is generally explicit, for brevity we follow the Einstein summation convention, which implicitly binds any variable that is used twice, benefiting from the special rules of tensor calculus as a language.) Library terms involve only variable indices. Both variable and literal indices can be represented not only as letters but also non-negative integers: e.g., $\alpha, x \leftrightarrow 0$, $\beta, y \leftrightarrow 1$, etc. While this representation leaves the type of an index ambiguous, it defines a useful total ordering of each index type.

Alongside the primary symbolic language in which expressions have variable and literal indices, it is helpful to define the closely related *template language*, which instead uses two auxiliary index subtypes but has otherwise identical syntax. These subtypes are the *index hole*, which can only take one value notated as {} and represents a slot in a tensor where an index should be placed, and the `SMTIndex`, which is a variable used as a temporary placeholder index while generating assignments from

index holes to variable indices by solving a satisfiability modulo theory (SMT) problem (see Section 3.2). These assignments map template expressions, which do not have a semantic (or physical) meaning of their own, to all corresponding expressions in the primary language.

Well-formed expressions are required to contain instances of only a single index type; for instance, the expression $\partial_x u_\alpha$ is not meaningful in the continuous language. Thus, objects representing expressions can be specified according to their index type – for instance, an observable with variable indices or a term with literal indices.

Figure 3.1: Extended Backus-Naur form (EBNF) specification for the continuous SPIDER language. The discrete language instead takes a prime to be a tuple consisting of a derivative and a coarse-grained product (CGP), where a CGP is the terminal symbol ρ operating on a list of observables. Finally, the corresponding template languages use index hole and `SMTIndex` types instead of variable and literal indices.

name = string;
variable index := "α" | "β" | ...;
literal index := "x" | "y" | ...;
index := **variable index** | **literal index**;
rep = irreducible or reducible representation;
coefficient $\in \mathbb{R}\backslash\{0\}$;
observable := **name**, **rep**, {**index**};
differential := "∂", (**index** | "t")
derivative := {**differential**};
prime := **derivative**, **observable**;
term := {**prime**, "$\times$"}, **prime**;
equation := {**coefficient**, "$\cdot$", **term**, "+"}, **coefficient**, "$\cdot$", **term**

The next objects in the type hierarchy are the *observable* and the *derivative*. An observable's attributes are its name and the irreducible or reducible representation to which it belongs (providing its rank and symmetry properties, e.g., rank 2 antisymmetric), as well as the list of indices, which has length equal to the rank. General observables have no special symmetry properties, but, for instance, second-rank observables may be explicitly symmetric or antisymmetric. These symmetry properties are important as they determine whether an observable's indices can be freely per-

muted (possibly introducing a sign change for antisymmetric observables). A derivative is a subexpression representing the operator $\partial_t^i \partial_{I_1}^{j_1} \partial_{I_2}^{j_2} \dots \partial_{I_m}^{j_m}$ where $I_1, I_2, \dots, I_m$ are indices of a given type and $i, j_1, j_2, \dots, j_m \in \mathbb{Z}_{\geq 0}$. Similarly to an observable, a derivative can be expressed in terms of its list of indices (with possible repetitions), but all of the indices in this list always commute just as the corresponding differentials do.

In the continuous language, a prime can be represented as a tuple (D, O) for a derivative D and observable O and is written in algebraic form by simply concatenating their symbolic representations. The situation is more complicated for the discrete language: in this case, a prime is a derivative operating on a *coarse-grained product* (CGP), which we previously denoted as $\rho[O_1 O_2 \dots O_n]$. The only attribute of a CGP type is its list of component observables $O_1, \dots, O_n$, which can be reordered arbitrarily since it represents a commutative tensor product.

The following step in the hierarchy is a term, which is a product of an unordered nonempty list of primes. Although any term with indices of literal, hole, or `SMTIndex` type is generally valid, a term with variable indices must satisfy additional constraints to be a well-formed expression. First, each index must either be free (occurring exactly once in the whole recursive tree of attributes defining the tree) or bound (occurring exactly twice). Second, we adopt the convention that the free indices should be numbered $0, 1, \dots, r-1$ and the bound indices, $r, r+1, \dots, r+q-1$ (here using the integer rather than letter representation), where r is the rank of the term. Together, these conditions ensure that each term represents a valid contraction of a product of full-rank primes (that is, primes with index holes). Note that there is no need to generate terms corresponding to distinct irreducible representations – the unsplit libraries suffice, as projection onto the irreducible representations is handled during evaluation (see Section 3.3).

Finally, an equation is a linear combination of terms with nonzero real coefficients, that is, an unordered nonempty list of tuples $(c, \mathcal{T})$, where $c \in \mathbb{R}\backslash\{0\}$ and $\mathcal{T}$ a term. (Of course, we would like to identify *parsimonious* equations, that is, linear combinations of only a small number of terms that approximately equal zero when evaluated.) To extend the definition of complexity to equations, we let an equation's complexity equal the highest complexity of any of its terms. An equation with variable indices must satisfy the constraint that all terms belong to the same irreducible representation, which is enforced by requiring the terms to reside in the same library (and use the same free indices). Per the note immediately above, it is sufficient to take terms with the same rank and record the projection necessary to go from the unsplit library to the irreducible representation for which the equation holds. Equations are also semantically invariant with respect to rescaling all of the term weights by a nonzero scalar.

In addition to the evident semantic symmetries of our tensor calculus with respect to swapping indices or subexpressions (such as observables in CGPs or primes) that commute, in the case of variable indices, there is an additional symmetry due to the fact that the labels of bound variables are arbitrary. Thus, permuting these labels in any way leaves the meaning of the expression unchanged, inducing another $q!$-fold symmetry for an expression with q free indices. (This is the notion commonly known as α-equivalence.) For terms and equations in one of the irreducible representations, by the definition of the projection operators STF and AS, there is a similar symmetry with respect to permuting *free* indices as well.

Finally, it is worth noting specific useful operators that will be used in Section 3.4, acting on primes, terms, or equations:

- the product operator $\times$, which inputs two arguments that may each be a prime or a term; the output is the term consisting of the concatenation of all primes contained in each operand.

- the derivative operator ∂, which inputs a differential (with respect to t or an index i of either variable or literal type) and a prime, term, or equation. For a prime, this outputs a new prime, simply appending the differential to its derivative part; for terms, the output is a sum of terms with unit coefficients computed according to the product rule (which can be represented by the equation type); and for equations, it acts linearly on each term.

- contraction $*$, which inputs two free indices α, β and a term or equation containing those free indices and outputs a new term or equation where these free indices are remapped to a new bound index.

- and scalar multiplication $\cdot$, which inputs a real number c and equation and outputs the equivalent equation where each coefficient is rescaled by c.

3.2 Library Generation

We now turn our attention to the problem of generating the full libraries up to a given complexity threshold. If multiple equivalent terms are included in the library, then the regression procedure will consistently discover relations between the equivalent terms, which are trivial identities that imply nothing about the underlying dataset. Thus, we must instead enumerate all semantically distinct terms of a given rank with variable indices without including any duplicates. (Here, *semantic* equality, the mathematical equality of two expressions, contrasts with *syntactic* equality, equality of two representations in the formal language. For instance, the terms $u \cdot v$ and $v \cdot u$ are semantically but not syntactically equal. By default, we will use the term "equality" to indicate semantic equality, as this agrees more closely with our algebraic intuition.)

Typically, enumerating a grammar in computer science is a straightforward task. Naively, library generation could be accomplished by writing all contractions of all products of all primes, keeping only one representative from each set of equal terms,

and then sorting by rank. The idea of selecting a representative from each equivalence class of semantically equal terms is formalized by the concept of *canonicalization* (as exemplified by canonical forms in mathematics). If we choose a set of properties satisfied by only one representative from the equivalence class, we may distinguish the canonical form of a term as the one that satisfies these properties. Perhaps the simplest choice in the case of terms, which possess (1) finite equivalence classes and (2) a relatively simple to express total ordering relation, is to take only the minimal term from each class with respect to the ordering to be canonical. As an example, an injective mapping from the set of terms to the set of strings of characters induces a total lexicographic ordering on terms. We take the lexicographic ordering induced by such a string representation as our starting point for defining which terms are canonical.

However, iterating over all of the possible terms in the language and then computing the equivalence classes is a very inefficient procedure, since the vast majority of these terms will not be in canonical form and will ultimately be discarded; instead, we prefer to generate terms that are already in canonical form whenever possible. Therefore, we enumerate the grammar using a less conventional two-step approach. First, we generate all canonical products of primes directly, which fixes the structure of the terms. Then, considering each product of primes separately, it remains only to determine which contractions correspond to distinct terms. These canonical index assignments are found by solving a satisfiability modulo theory (SMT) problem, obtaining a set of valid mappings from the list of index holes to a list of variable index labels.

This procedure leverages the fact that the symmetries of the grammar take two fundamentally different forms: one is commutativity of subexpressions, which concerns only the structure of the terms, and the other is reordering of commuting indices and relabeling of bound indices, which does not affect the structure. As a result, we

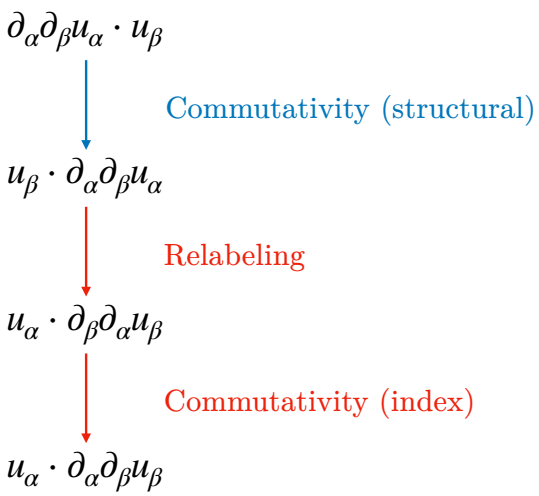


Figure 3.2: A sequence of term rewrites transforming the term $\partial_\alpha \partial_\beta u_\alpha \cdot u_\beta$ to its canonical form. Rewrites employing structural (index) symmetries are labeled in blue (red).

decompose the notion of canonicity into two parts: structural canonicity, which can be enforced before declaring any of the concrete indices, and index canonicity, which is defined only once the structure of the term is fixed. As an example, Figure 3.2 shows a sequence of term rewrites relying on all three different symmetries that brings a term $\partial_\alpha \partial_\beta u_\alpha \cdot u_\beta$ to its canonical form. The first rewrite is a reordering of the structure which immediately leads to structural canonicity, whereas the second and third rewrites both exploit index symmetries; the second rewrite relabels $\alpha \leftrightarrow \beta$ and the third swaps the commuting indices in the derivative.

Given a template term containing index holes only, structural canonicity is trivial to enforce by sorting any commuting subexpressions of the term. Generating only structurally canonical terms is equally simple: the primes must simply be listed in their lexicographic order. Thus, it is easy to generate the structures of all terms in the library by direct enumeration: first, we write all primes up to the chosen complexity threshold, and then we list all of their ordered products that do not exceed this threshold.

In contrast, index canonicity requires considering all possible transformations of a term $\mathcal{T}$ with respect to two symmetry groups: the group S of permissible index permutations (which depends on the structure of the term) and the group L of index relabelings. Thus, the canonical indexing must be selected from the equivalence class which is a double coset $S\mathcal{T}L$. An algorithm for solving this problem in the case of general permutation groups was introduced in Ref. [22] and adapted to tensor calculus as the so-called Butler-Portugal algorithm [79], which was later implemented as a `Mathematica` routine [81]. This algorithm relies on sophisticated techniques from computational group theory and can be inefficient in practice for tensors with large sets of permutable indices [88]. (For instance, these may commonly arise when considering products of several copies of the same tensor.)

In this work, we take an alternative approach that utilizes an SMT formulation in linear integer arithmetic (LIA) to construct all possible canonical indexings of a given template term. Each index hole of the template term is replaced with an `SMTIndex` variable v_i, which will represent the index of the indexed term at position i. First, we introduce cardinality constraints: an unindexed term with m holes can have any rank $r \in \{m, m-2, \ldots, m \mod 2\}$; once r has been fixed, the labels $0, 1, \ldots, r-1$ correspond to free indices and therefore must each be assigned to exactly one `SMTIndex`, whereas the remaining labels $r, r+1, \ldots, \frac{m+r}{2} - 1$ signify bound indices and must each be used twice. Next, canonicity with respect to index relabeling implies an ordering among the first occurrences of each bound index in the expression: label r must first appear before label $r+1$, and so on. Similarly, since the canonical form of tensors also has the free indices used in order (the other combinations appear only when symmetrizing or antisymmetrizing as part of projecting onto the irreducible representations), the labels $0, \ldots, r-1$ appear in order as well. These constraints may be encoded by introducing a pair of auxiliary variables s_i, p_i for each index hole, representing, respectively, the smallest free and bound index labels not in use up to

position i in the expression – these variables are the only values of v_i that are allowed to be used for the *first* time in the expression at this position. If $v_i = s_i$ ($v_i = p_i$), respectively, we require $s_{i+1} = s_i + 1$ ($p_{i+1} = p_i + 1$); else, they are not incremented. Finally, it is necessary to ensure that commuting indices or subexpressions does not result in an index labeling that lexicographically precedes the candidate labeling. These symmetries are fixed through four separate sets of constraints:

1. given any commuting set of index holes C, the labels v_i must be monotonically nondecreasing for $i \in C$, i.e., $v_i \leq v_j$ whenever $i < j$ and $i, j \in C$. For instance, $\partial_\alpha \partial_\beta \partial_\beta$ is valid but $\partial_\beta \partial_\alpha \partial_\beta$ is not.

2. the indexings assigned to equal subexpressions in the template term (for instance, multiple copies of an observable $O_{\{\}}$ in a CGP or multiple copies of the same prime in a term) are lexicographically ordered: given $i < j$ and subexpressions $A_{v_i,\ldots,v_{i+r-1}}$, $A_{v_j,\ldots,v_{j+r-1}}$, we have $(v_i, \ldots, v_{i+r-1}) \leq_{lex} (v_j, \ldots, v_{j+r-1})$. As an example, this means that the assignment $A_{\beta\alpha} \cdot A_{\alpha\gamma}$ is invalid, although $A_{\beta\alpha} \cdot B_{\alpha\gamma}$ would be allowed.

3. For each commuting set of index holes C, the labeling must be lexicographically minimal under simultaneous reordering of C and α-conversion (that is, permuting the names of the bound index labels). This can be checked as follows: let I be the set of all distinct bound index labels used in C, and I_2 the set of labels that are repeated within C. Then, when considering all occurrences of the labels in I *not in* C and the leftmost occurrences of the labels in I_2 (which are in C by definition), the labels in I must appear in order from left to right. For instance, the expression $u_\alpha \cdot \partial_\alpha \partial_\beta \partial_\beta \partial_\gamma \partial_\delta u_\gamma \cdot w_\delta$ is canonical. Labeling the index holes from left to right as $0, 1, \ldots$, here we have $C = \{1, 2, 3, 4, 5\}$, $I = \{0, 1, 2, 3\}$, and $I_2 = \{1\}$. The relevant positions $v_0 = 0\,(\alpha)$, $v_2 = 1$ (β), $v_6 = 2$ (γ), and $v_7 = 3$ (δ) are in the proper order $(0 < 2 < 6 < 7)$. In contrast,

$\partial_\alpha \partial_\beta \partial_\beta u_\alpha$ is not canonical: $I = \{0, 1\}$, $v_1 = 1$ and $v_3 = 0$, but $3 > 1$. The canonical form is instead $\partial_\alpha \partial_\alpha \partial_\beta u_\beta$, which does satisfy this constraint.

4. Similarly, for each set of structurally equal subexpressions E, the labeling must be lexicographically minimal under simultaneous reordering of E and α-conversion. It is sufficient to check that permuting the bound index labels used in E and then sorting the subexpressions again cannot result in a labeling that lexicographically precedes ours.

The last two constraints are the most complex, as they prevent all violations of index canonicity as a combination of both commuting indices or subexpressions and relabeling of indices. While they could be encoded explicitly as part of the SMT problem, this results in a substantially more expensive formulation. For instance, constraint 3 would require approximately $|C|m$ constraints, since I and I_2 are not fixed until a solution has already been generated; constraint 4 is even trickier to encode. Instead, we check constraints 3 and 4 as a post-processing step once a candidate solution has been generated, discarding the candidate solutions that fail to satisfy them.

Frequently, multiple indexings of the same unindexed term are canonical. To identify all canonical indexings, every time the indexing SMT problem is solved, we record this solution and introduce a new set of constraints prohibiting it until the problem becomes unsatisfiable. In `PySPIDER`, we use the Z3 SMT solver [33]. We have found that our procedure for generating the libraries has an insignificant computational cost compared to other steps in the SPIDER algorithm, such as library evaluation. For instance, the complexity-7 hydrodynamic unsplit libraries up to rank 2, which in total contain 2634 terms, were generated in 7.0 seconds or 2.7 ms per term on a 2024 MacBook Air M3. In contrast, evaluating a single term on a typical number of 100 integration subdomains takes O(100-1000) ms on average.

The SMT problem described in this section can also be modified in order to canonicalize the indexing of terms arising by the operations defined at the end of Section 3.1, which is an important step in efficient equation synthesis (see Section 3.4). In this case, the labels can no longer be chosen as freely. Instead, each pair of equal (bound) indices denotes a contraction and must always be remapped to another pair of bound indices; this is most easily accomplished by using the same SMT variable to represent both instances of the same bound index. Otherwise, the variables and constraints remain the same, although the solution of the problem now represents a mapping from variable indices to a new set of variable indices, rather than from index holes to variable indices.

Finally, it is worth noting that the differences between the continuous and discrete language are mostly inconsequential to library generation: these affect only the list of primes used to generate the terms. The indexing SMT problem is language-agnostic, requiring knowledge of the symmetries of the term only, not the type of its subexpressions.

3.3 Term Evaluation

Now that we have established SPIDER's ability to generate the libraries automatically, we now discuss how to evaluate each term in a library. Generalizing the brief discussion in Section 1.2, for each term (with variable indices) $\mathcal{T}$ belonging to the library $\mathcal{L}$ corresponding to the metric space V, this is simply a matter of computing n_w inner products (where $n_w \gg n_t = |\mathcal{L}|$) of the form

$$\langle f_T, w_i \rangle_\Omega = \int_\Omega \langle f_T(\boldsymbol{x}, t), w_i(\boldsymbol{x}, t) \rangle_V \, d\boldsymbol{x} \, dt \tag{3.1}$$

where $f_\mathcal{T} : \Omega \to V$ is the evaluation of $\mathcal{T}$ as a V-valued field, and $w_i : \Omega \to V$ is a weight function supported on an integration subdomain $\Omega_i \subset \Omega$ for each $i \in [n_w]$.

Once these inner products have been evaluated, the sparse regression problem reduces to finding a regularized solution $\boldsymbol{c}$ of $G\boldsymbol{c} = 0$, where each entry G_{ij} is the inner product of the ith weight function and jth term.

For the much simpler setting of random $n_w \times n_t$ Fourier feature matrices with entries $\exp(i\langle f, w\rangle)$ evaluated using i.i.d. normal features $f \sim \mathcal{N}(0, \gamma^2 I_d)$ and weights $w \sim \mathcal{N}(0, \sigma^2 I_d)$, it is known that it is sufficient to take $n_w \propto n_t \log n_t$ in order to ensure that the matrix is well-conditioned with high probability [28], leading to low test error. There is no analogous rigorous theoretical result corresponding to (3.1). In practice, n_w is generally chosen large enough that the residual and coefficient errors scale as $n_w^{-1/2}$ due to a central limit theorem-type scaling of the effect of noise and/or discretization error [57].

Let us formalize the intuition encapsulated in the notational conventions of tensor calculus and specify what it means to evaluate $\mathcal{T} \mapsto f_{\mathcal{T}}$. If V contains tensors of rank r in n spatial dimensions, then evaluating $\mathcal{T} \mapsto f_{\mathcal{T}}(\mathbf{x}, t)$ for fixed $\mathbf{x}$ and t requires computing each of the n^r components of the tensor, which correspond to each of the possible assignments of free indices $a_f : \{0, 1, \ldots, r-1\} \to \{1, 2, \ldots, n\}$ (with the assigned values more naturally represented as $x, y, \ldots$). As an example, second-rank tensors in two spatial dimensions have components $(\alpha, \beta) = (x, x), (x, y), (y, x), (y, y)$. Notating the component of a tensor corresponding to a_f by $[\cdot]_{a_f}$, the inner product in (3.1) can be expressed as

$$\langle f_{\mathcal{T}}, w_i \rangle_\Omega = \int_\Omega \sum_{a_f} [f_{\mathcal{T}}]_{a_f}(\boldsymbol{x}, t) \cdot [w_i]_{a_f}(\boldsymbol{x}, t)\, d\boldsymbol{x}\, dt \tag{3.2}$$

To evaluate a component $[f_{\mathcal{T}}]_{a_f}$, we must then sum over *all* assignments of the bound indices $a_b : \{r, \ldots, i_{max}\} \to \{1, \ldots, n\}$. Thus, evaluating $\mathcal{T}$ requires considering all possible assignments (a_f, a_b) of the variable indices to literal indices in $\{1, \ldots, n\}$,

with each assignment mapping $\mathcal{T}$ to a term L_{a_f,a_b} that has only literal indices:

$$\langle f_{\mathcal{T}}, w_i \rangle_\Omega = \int_\Omega \sum_{a_f} \sum_{a_b} L_{a_f,a_b}(\boldsymbol{x}, t) \cdot [w_i]_{a_f}(\boldsymbol{x}, t)\, d\boldsymbol{x}\, dt \tag{3.3}$$

Such a term may be directly evaluated at each point $(\boldsymbol{x}, t)$ using a naive recursive procedure. An observable with literal indices is evaluated by looking up the appropriate component at the given grid point. A prime in the continuous language can be evaluated by computing the relevant derivatives of the observable numerically, such as by finite differences, and a term, by taking the product of all of its primes.

For the discrete language, evaluation becomes more complex. In that case, recall that a CGP $\rho[O^{(1)} \cdot \cdots \cdot O^{(k)}]$ is defined by the convolution

$$\rho[O^{(1)} \cdot \cdots \cdot O^{(k)}](\boldsymbol{x}, t) = \sum_{i=1}^{N} \left[W_h(|\boldsymbol{x} - \boldsymbol{x}_i(t)|) \prod_{j=1}^{k} O_i^{(j)}(t) \right], \tag{3.4}$$

where $O_i^{(j)}(t)$ is the value of the jth observable (with literal indices indicating the appropriate component) for particle i at time t. Moreover, the data are no longer available on a grid but rather are localized to the particle positions. Nevertheless, we can evaluate the *coarse-grained* data on a uniform spatial grid with any spacing of our choosing. (This spacing may be chosen by balancing computational cost and error due to the limited grid resolution.) Thus, at each time t and spatial grid point x, we compute a weighted sum of kernels $W_h(|\boldsymbol{x} - \boldsymbol{x}_i(t)|)$ centered at N different points $\boldsymbol{x}_i$. This step can be optimized if W_h is compactly supported: the sum can be taken over only the set of particles that have convolutions overlapping with $\boldsymbol{x}$, and this set is computed efficiently using k-d trees [5]. The rest of the process remains the same as in the continuous case: a prime can be evaluated by taking the appropriate derivatives of the CGP and a term is once again simply a product of primes.

3.3.1 Weight Functions and Integration by Parts

However, as we know, numerical evaluation of high-order derivatives of the data suffers from sensitivity to noise, so we aim to alleviate this problem by integrating $L_{a_f,a_b}[w_i]_{a_f}$ by parts whenever possible. To help establish how such a procedure might work, let us briefly discuss appropriate choices of the weight functions w_i and integration subdomains Ω_i.

Guidelines for selecting Ω_i are mostly qualitative. It is generally advantageous to take the Ω_i's to be a random sample of $(n+1)$-dimensional hypercubes within the dataset, e.g., in three spatial dimensions

$$\Omega_i = \{(x, y, z, t) \mid \underline{x}_i \leq x \leq \overline{x}_i, \underline{y}_i \leq y \leq \overline{y}_i, \underline{z}_i \leq z \leq \overline{z}_i, \underline{t}_i \leq t \leq \overline{t}_i\} \tag{3.5}$$

This allows data exhibiting diverse dynamical regimes to be included in the regression. If the sampled data are too similar, the models learned by SPIDER may not be sufficiently general. For instance, in Ref. [56], we show that the location of the integration subdomains determines both the level of accuracy of the coefficients in the learned equations and, for noisy data, the identified form of the equations. In that study, SPIDER was used to learn the equations describing highly turbulent flow of an incompressible fluid in a channel, namely incompressibility and the Navier-Stokes equation; when only mid-channel data was used, which exhibits relatively minor effects of viscosity, SPIDER no longer detects the viscosity term at moderate noise levels. The dimensions of each subdomain are normally chosen to be roughly several times the characteristic length/time scale of the data [57]. It is also important to ensure that each dimension includes enough grid points (typically at least 20) in order to allow (3.1) to be evaluated by quadratures with sufficient accuracy.

For data that lie on a uniformly spaced grid, the integral in (3.1) can be computed by trapezoidal quadrature, which achieves extremely high accuracy when the peri-

odization of the integrand is sufficiently smooth [114]. As noted in Refs. [57] and [83], this can be accomplished by taking w_i such that its first several derivatives vanish at the boundary $\partial\Omega_i$. For instance, we may take w_i to be composed of one-dimensional polynomial factors as follows:

$$w_i(x, y, \ldots, t) = \tilde{w}(\hat{x})\tilde{w}(\hat{y}) \ldots \tilde{w}(\hat{t})A, \tag{3.6}$$

where $A \in V$ is a tensor with constant coefficients,

$$\tilde{w}(\hat{x}) = \begin{cases} (1 - \hat{x}^2)^q, & |\hat{x}| \leq 1 \\ 0, & |\hat{x}| > 1 \end{cases} \tag{3.7}$$

for q a moderately large integer (typically $q \sim 10$) [56], and the hat denotes nondimensionalization using the order-preserving affine map $[\underline{x}, \bar{x}] \to [-1, 1]$. A more flexible choice of the weight function is

$$w_i'(x, y, \ldots, t) = w_i(x, y, \ldots, t)g_i(x, y, \ldots, t) \tag{3.8}$$

where g_i is any smooth scalar-valued modulating function, such as a Legendre polynomial or Fourier mode [57, 50]. Using different modulating functions g for multiple weight functions defined on the same subdomain Ω_i allows more independent information to be extracted from a given data sample.

The vanishing of the derivatives of w_i on $\partial\Omega_i$ allows us to perform the desired integration by parts without introducing boundary terms. Moreover, as long as g_i can be differentiated analytically any number of times, the weight $w_i' = w_i g_i$ as a whole can be as well (since the remaining part is polynomial). We now explain an algorithm for minimizing the highest order of derivative that needs to be evaluated directly on the data. Derivatives along each coordinate axis $(x, y, \ldots, t)$ can be considered

independently, so without loss of generality, let us consider integrating by parts with respect to x. There are two patterns of terms that can be improved by integrating by parts. First, let P be a prime (with literal indices) and w some scalar component of the weight function. Then any term-weight product of the form $(\partial_x P \cdot P^m) \cdot w, m \geq 0$ can be integrated by parts to obtain $P^{m+1} \cdot \left(-\frac{1}{m+1} \partial_x w\right)$, a product of a single term and a different weight function. Additionally, if $P_1, P_2, \ldots, P_p$ are primes containing no x derivatives and $n_1 - 2 \geq n_2 \geq \cdots \geq n_p$ are nonnegative integers, then

$$\begin{aligned} w \prod_{i=1}^{p} \partial_x^{n_i} P_i \xrightarrow{\text{i.b.p.}} -\partial_x^{n_1 - 1} P_1 \cdot \partial_x \left(w \prod_{i=2}^{p} \partial_x^{n_i} P_i \right) = \\ = -\partial_x^{n_1 - 1} P_1 \left(\partial_x w \cdot \prod_{i=2}^{p} \partial_x^{n_i} P_i + w \cdot \sum_{i=2}^{p} \left(\partial_x^{n_i + 1} P_i \cdot \prod_{j \in [p] \setminus \{1, i\}} \partial_x^{n_j} P_j \right) \right) \end{aligned} \tag{3.9}$$

is an integration by parts that decreases the maximum order of x derivative on the data by one, at the cost of converting one term-weight product into a sum of p new term-weight products. For each integration axis $x, y, \ldots, t$, each of these rules is repeatedly applied to every term-weight product as long as they fit one of these two patterns, after which integration by parts provides no further benefit. Thus, the integration by parts procedure ultimately produces an optimal rewrite of the form

$$\int_\Omega L(\boldsymbol{x}, t) \cdot w(\boldsymbol{x}, t) \, d\boldsymbol{x} \, dt = \sum_s \int_\Omega L_s(\boldsymbol{x}, t) \cdot w_s(\boldsymbol{x}, t) \, d\boldsymbol{x} \, dt \tag{3.10}$$

for some new set of terms L_s and weights w_s.

3.3.2 Splitting among Irreducible Representations

Up until now, we have not addressed splitting the terms among irreducible representations. Note that we have defined each weight function (and its derivatives) as the product of a scalar-valued part (which we will simply call w) and the tensor A with constant coefficients. Suppose $\mathcal{T}_{\alpha\beta}$ is a second-rank tensor-valued term and $A_{\alpha\beta}$ is an antisymmetric second-rank tensor. Then (3.1) becomes

$$\begin{aligned}\langle f_{\mathcal{T}}, wA\rangle_\Omega &= \int_\Omega w \sum_{\alpha,\beta} (f_{\mathcal{T}})_{\alpha\beta} A_{\alpha\beta}\, d\Omega = \sum_{\alpha,\beta} \int_\Omega w (f_{\mathcal{T}})_{\alpha\beta} \cdot \frac{1}{2}\left(A_{\alpha\beta} - A_{\beta\alpha}\right) d\Omega = \\ &= \sum_{\alpha,\beta} \int_\Omega \frac{1}{2}\left((f_{\mathcal{T}})_{\alpha\beta} - (f_{\mathcal{T}})_{\beta\alpha}\right) wA_{\alpha\beta}\, d\Omega,\end{aligned} \tag{3.11}$$

which is equivalent to an evaluation of the *antisymmetric part* of $\mathcal{T}$. More generally, if A lies in any subspace $W \subset V$, we have $\langle f_{\mathcal{T}}, wA\rangle = \langle \mathrm{proj}_W f_{\mathcal{T}}, wA\rangle$. Letting A vary over a *basis* of W is sufficient to produce the full set of independent evaluations of $\mathrm{proj}_W \mathcal{T}$. Hence, in order to evaluate the rank-2 antisymmetric library, we use weight functions with a set of $n(n-1)/2$ different A's that form a basis for the space of antisymmetric matrices. For instance, in $n = 3$ spatial dimensions, we may take

$$A_1 = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},\ A_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix},\ A_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}. \tag{3.12}$$

Similarly, the symmetric trace-free library can be evaluated using a $(n(n+1)/2-1)$-dimensional basis for the symmetric trace-free matrices, e.g.,

$$A_4 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, A_5 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, A_6 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \tag{3.13}$$

$$A_7 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, A_8 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}.$$

The simpler cases of the scalar library $V = \mathbb{R}$ and vector library with $V = \mathbb{R}^n$ require no such splitting, since V is already irreducible. For the vector library, we may take a vector basis $A_i = e_i$ of unit vectors along each axis, and for the scalar library A is redundant and can be set to 1.

As noted in Section 2.2, not all terms (with variable indices) have nontrivial projections onto both the antisymmetric and symmetric trace-free irreducible representations. The exceptions can be detected systematically and removed from the corresponding library as follows. First, swap the free indices of the term and re-canonicalize it; if this recovers the pre-swap term, then record whether the sign of the expression was changed by any resulting permutations of commuting indices. Respectively, if the sign is unchanged (changed), the term is symmetric (antisymmetric) and so its antisymmetric (symmetric trace-free) projection is zero.

3.3.3 Evaluating Boundary Conditions

Finally, let us briefly touch on how this procedure can be adapted to evaluating the boundary condition libraries formulated in Section 2.3. As noted earlier, we can no longer use $(n+1)$-dimensional integration domains Ω_i; instead, we flatten each integration domain to lie on the boundary of the dataset, while leaving its other dimensions unchanged. For instance, for a flat boundary wall $y = 0$ of a dataset with three spatial dimensions, we would take

$$\Omega_i = \{(x, 0, z, t) \mid \underline{x}_i \le x \le \overline{x}_i, \underline{z}_i \le z \le \overline{z}_i, \underline{t}_i \le t \le \overline{t}_i\} \tag{3.14}$$

Correspondingly, the derivatives with respect to the axis normal to the boundary cannot be eliminated via integration by parts and can only be evaluated numerically. In this example, we might estimate the derivatives with respect to y by finite differencing, using the first few $y = c$ slices closest to the wall. If the wall-normal direction is $+y$ (or $-y$), then the normal vector $\boldsymbol{n}$ appearing in the boundary condition libraries simply evaluates to the constant $\boldsymbol{e}_y$ (or $-\boldsymbol{e}_y$).

Just as the weight functions are split into separate bases corresponding to various irreducible representations for the bulk equation libraries, we can generalize this approach to boundary conditions. Since a full set of boundary conditions can be obtained from only the tangential libraries, the weight basis can omit the normal direction. For instance, in the present example, we need only consider $A = \boldsymbol{e}_x, \boldsymbol{e}_z$ for the tangential vector library. For the antisymmetric and STF irreducible representations of the second-rank tangential library, we would use the respective bases consisting of the matrices $\{A_2\}$ and $\{A_5, A_8\}$ from (3.12)–(3.13).

An example of this procedure was shown in Ref. [56], where by combining results from the scalar and vector boundary condition libraries, we learned the well-known no-slip boundary condition $\boldsymbol{u} = 0$ for an incompressible channel flow.

3.4 Equation Synthesis

The previous section has established how to compute a library matrix G by evaluating each term in a SPIDER library $\mathcal{L}$. Suppose there exists a deterministic algorithm for finding a single sparse approximate solution $\boldsymbol{c}$ of the matrix regression problem $G\boldsymbol{c} = 0$, along with a measure r of the solution's accuracy, which we call its "residual." (In the next chapter, we will discuss such algorithms in depth.) The solution obtained by this oracle corresponds to a single relation that approximately holds on the present dataset. Using the oracle, can we also efficiently identify other approximate relations expressible using the terms in $\mathcal{L}$? This section describes an algorithm for finding *all* relations contained in all of the libraries.

To start, let us determine when one equation implies another. Given an equation representing that a linear combination of terms with variable indices is zero,

$$C = \sum_{i \in S} c_i \mathcal{T}_i = 0, \tag{3.15}$$

the derivatives of the equation also hold:

$$\partial_t C = \sum_{i \in S} c_i \partial_t \mathcal{T}_i = 0 \tag{3.16}$$

and

$$\partial_\alpha C = \sum_{i \in S} c_i \partial_\alpha \mathcal{T}_i = 0 \tag{3.17}$$

for a free index α not used in any of the terms comprising C. Similarly, we may multiply the equation by any prime p:

$$p \cdot C = \sum_{i \in S} c_i (p \cdot \mathcal{T}_i) = 0 \tag{3.18}$$

(In the upcoming discussion, it is sufficient to consider full-rank primes, i.e., ones with all free indices.) Finally, we can contract any two free indices in C:

$$\mathrm{tr}_{\alpha,\beta}(C) = \sum_{i \in S} c_i \mathrm{tr}_{\alpha,\beta}(\mathcal{T}_i) = 0 \tag{3.19}$$

On the other hand, the reverse of any of these operations does not necessarily return another true equation. Note that the set of valid equations is also closed under linear combinations (i.e., it is a linear subspace), but this produces nontrivial implications only when starting from at least two linearly independent equations. We will consider these later in this section.

In fact, all logical implications between two equations that are general (hold for any dataset) can be generated by repeated application of the implication operators $\{\partial_t, \partial_\alpha, p\cdot, \mathrm{tr}_{\alpha,\beta}\}$ (excluding the trivial implication $C \Rightarrow C$, which is generated by applying none of these). As an illustration, Figure 3.3 shows an example of a directed graph of implied equations that can be generated from the incompressibility condition $\nabla \cdot \boldsymbol{u} = 0$ in a hydrodynamic library with observables $\{p, \boldsymbol{u}\}$.

The set of implication operators also satisfies another powerful property. Applying the first three operators evidently increases the complexity of a term (except when $p = 1$, in which case the implication is trivial). The contraction operator keeps complexity constant and reduces rank by 2 by taking a (generalized) trace of the input tensor. However, since irreducible representations of rank 2 are trace-free, any contraction of an equation obtained in a rank-2 irreducible representation is equivalent to the trivial identity $0 = 0$. This means that useful implications involving the contraction operator must also use one of the other three operators. It follows that **any equation learned by SPIDER has lower complexity than each of the equations that it directly implies**.

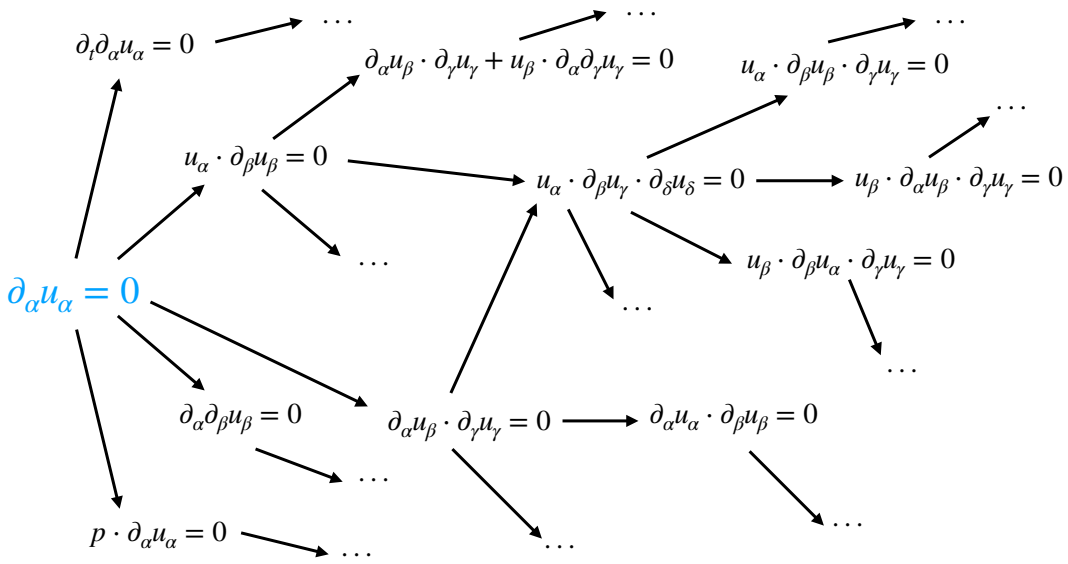


Figure 3.3: A directed graph showing some of the equations implied by the incompressibility condition $\nabla \cdot \boldsymbol{u} = \partial_\alpha u_\alpha = 0$ in a hydrodynamic library with observables $\{p, \boldsymbol{u}\}$. Note that all equations have been canonicalized.

SPIDER uses a set of libraries in bijection with the irreducible representations: $\{\mathcal{L}_i\}_{i \in \text{irreps}}$. The symbolic structure of implication between equations can be leveraged by noting that each library $\mathcal{L}_i = \mathcal{L}_i^{(c_{max})}$ with complexity threshold c_{max} gives rise to a nested chain of sublibraries with complexity thresholds $1 \leq c \leq c_{max}$: $\mathcal{L}_i^{(1)} \subset \mathcal{L}_i^{(2)} \subset \cdots \subset \mathcal{L}_i^{(c_{max})}$. Any implied equation in $\mathcal{L}_i^{(c)}$ is a consequence of an equation contained in another sublibrary with complexity threshold at most $c-1$; thus, once all equations with complexity up to $c-1$ have been identified, the implied equations of complexity c can be completely enumerated.

Equations also represent natural rewrite rules for terms. Specifically, given an equation, we may eliminate its most complex term $\mathcal{T}_j$ (breaking ties arbitrarily) by writing it as a linear combination of the remaining terms:

$$C = \sum_{i \in S} c_i \mathcal{T}_i = 0 \quad \Rightarrow \quad \mathcal{T}_j = \sum_{i \in S \setminus \{j\}} -\frac{c_i}{c_j} \mathcal{T}_i \tag{3.20}$$

Therefore, any other independent equation containing $\mathcal{T}_j$ can also be written without using $\mathcal{T}_j$, and so the sublibrary not containing $\mathcal{T}_j$ is sufficient to recover relations other than C.

A minor complication is that applying the direct definition of an implication operator to an equation may not result in a new equation where all terms are written in canonical form. This could be problematic, as applying the rewrite rule (3.20) requires each term to be recognizable as a library term. However, this issue can be addressed by canonicalizing each term after applying the implication operator. (This canonicalization may need to be incomplete for the right-hand-side terms of (3.20), since the free indices across all terms in the equation must be bound consistently to variables. This can be ensured by reusing the same remapping of free indices that canonicalizes the left-hand-side term in the remaining terms of the equation.)

Let us also acknowledge the role of irreducible representations in determining which implications are valid. Among the four irreducible representations considered in this dissertation—scalar, vector, rank-2 antisymmetric, and rank-2 STF—equations in the latter two representations have only a limited scope. While the scalar and vector equations may imply relations in any of the four irreducible representations, the rank-2 equations are implicitly projected onto a proper subspace of $\mathbb{R}^{n\times n}$ and thus do not imply equations in other irreducible representations.

After this preliminary discussion, we are ready to present an algorithm for identifying all fundamental and implied equations (Algorithm 1). Let us fix a residual threshold r_{max} such that we would consider any relation with $r \leq r_{max}$ to be approximately correct. (The choice of threshold is mostly subjective; for instance, one may choose to rerun the regression step for multiple choices of r_{max} and select the largest value that appears to return only reasonable equations. Chapter 4 recommends "relative" residuals r that always lie in $[0, 1]$ and denote the relative error of a relation: as an example, $r_{max} = 0.05$ can be said to correspond to a maximum acceptable

Algorithm 1: Full Equation Synthesis

Data: irreps: list of irreducible representations,
$\{(\mathcal{L}_i, G_i)\}_{i \in irreps}$: list of libraries and library matrices associated with irreps,
r_{max}: residual threshold, c_{max}: maximum library complexity
Result: $E_F = \{(e, r)\}_{e \in E}$: list of fundamental equations and associated residuals, E_I: list of implied equations

$E_F = \{\emptyset\}$;
$E_I = \{\emptyset\}$;
for $c \leftarrow 1$ **to** c_{max} **do**
  foreach i *in irreps* **do**
    $G_i^{(c)} \leftarrow$ columns of G_i for terms of $\mathcal{L}_i$ with $\mathcal{C} \leq c$;
    repeat
      $(e, r) =$ regressionOracle$\left(G_i^{(c)}\right)$;
      if $r \leq r_{max}$ **then**
        $E_F \leftarrow E_F \cup (e, r)$;
        foreach e_I *in impliedEquations*(e, c_{max}) **do**
          if $e_I \neq e$ **then**
            $E_I \leftarrow E_I \cup e_I$;
          end
          $j =$ irrep(e_I);
          if *rank*$(i) < 2$ *or* $i = j$ **then**
            eliminate max $\mathcal{C}$ term of e_I from library $(\mathcal{L}_j, G_j)$
          end
        end
      end
    until $r > r_{max}$;
  end
end

error of 5%.) First, we iterate over the complexity-1 sublibraries $\mathcal{L}_i^{(1)}$ corresponding to each irreducible representation i. Each sublibrary has a corresponding submatrix $G_i^{(1)}$ of the full regression matrix G_i that contains only the columns corresponding to terms of complexity at most 1. We now call the regression oracle, which we assume returns some equation e with $r \leq r_{max}$ whenever such an equation exists. So long as the residual is below the threshold, we accept the equation and record all of its implied equations e_I up to complexity c_{max}. Furthermore, we eliminate the most complex term in each e_I from its corresponding library as in (3.20) and permanently ignore the corresponding column of the library matrix G for this term's irreducible representation. Notably, e implies e, so the next equation returned by the regression oracle cannot contain all of the terms in e and hence must be distinct from e.

We continue to sample equations for this sublibrary from the regression oracle until $r > r_{max}$, at which point all independent acceptable equations present in the sublibrary have been generated and we can continue to the next complexity-1 sublibrary. Similarly, once we have considered all complexity-1 sublibraries, this process is then repeated for complexity 2 and so on, up to c_{max}. Algorithm 1 is complete in the sense that *all* valid equations up to complexity c_{max} can be expressed as some linear combination of equations in E_I, though the choice of E_F may depend on which equations are returned first by the regression oracle.

The procedure described above is not only essential to obtain multiple equations from a deterministic regression oracle but also improves the computational efficiency of regression substantially. Every identified equation, whether fundamental or implied, contains a term that can be pruned from the library. This means that successful equation synthesis at one complexity threshold reduces the size of the sublibraries that need to be considered at all higher complexities. This is especially true when an equation is very simple (low complexity) due to the combinatorial branching of the implication graph, as seen in Figure 3.3.

The discussion up to this point has mostly considered only logical implications of a single equation. Are there any additional equations that can be derived only by non-pairwise implication? For instance, one can consider linear combinations of equations such as $a_1 C_1 + a_2 C_2 = 0$ or products such as $C_1 C_2 = 0$. The latter is also a linear combination of equations of the form $(\text{term} \cdot C_2) = 0$, which present no independent interest. By definition, all such implications up to the complexity threshold are in the linear span of E_I; however, it is worth examining whether this span might contain a more sparse equation that should be present in E_F over another, less parsimonious one. Most likely, such an equation will already have been selected by an earlier regression iteration; empirically, we have not observed any dataset where a simpler fundamental equation that could be found by taking such a linear combination is missed by the regression, even though this is possible in theory. Instead, the reverse may occur: equations are occasionally reported as fundamental when they are actually implied by a combination of linear superposition and the implication operators used in SPIDER. One such example can be seen in Section 5.1, where the implied pressure-Poisson equation is listed among the fundamental equations. This situation should not present a problem for SPIDER users, since it is sufficiently rare and these equations are still often of interest to the user.

Chapter 4

Robust Sparse Regression

We have just seen how, with the help of an oracle that inputs a matrix of evaluated terms G and outputs the best candidate relation corresponding to this matrix data (or even just a sufficiently good one), it is possible to infer all approximate relations contained within a given library. We will now describe the most promising approaches for identifying such a candidate relation, taking into account not only the entries of G but also the context provided by physical considerations; the multiple recommendations at each stage can be mixed and matched, forming a flexible family of regression algorithms.

4.1 Quantifying Solution Performance

First, it is essential to characterize carefully what we seek from a "good" solution of the overdetermined system

$$G\boldsymbol{c} = 0, \tag{4.1}$$

where G is a $h \times w$ matrix with $h \gg w$ (in the context of Section 3.3, h is the number of terms and w is the number of weight functions). Clearly, (4.1) is meaningless unless we constrain the norm of $\boldsymbol{c}$, so we require $\|\boldsymbol{c}\| = 1$. (By default, we will take $\|\cdot\|$ to

mean the 2-norm.) The meaning of the underlying relation is, of course, unchanged by rescaling of $\boldsymbol{c}$, so for clarity of presentation, we will usually write relations with the largest coefficient scaled to 1. More importantly, we would like $\boldsymbol{c}$ to balance parsimony, measured in terms of the number of nonzero entries of $\boldsymbol{c}$, and accuracy, measured by the residual.

We aim to define a residual that is minimally sensitive to the scaling of G and comparable across problems. While the absolute residual

$$r_a = \|G\boldsymbol{c}\| \tag{4.2}$$

is not a strong candidate for either of these goals, there are multiple useful definitions of relative residual: in particular, the Frobenius relative residual

$$r_F = \frac{\|G\boldsymbol{c}\|}{\|G\|_F} \tag{4.3}$$

which is normalized by the Frobenius norm of G [53], and the dominant-term relative residual [98]

$$r_d = \frac{\|G\boldsymbol{c}\|}{\max_{i\in[w]} \|\boldsymbol{g}_i c_i\|}, \tag{4.4}$$

which is normalized by the largest contribution of a single term to $G\boldsymbol{c}$, namely the norm of some ith column of G times its coefficient. Both of these relative residuals are tightly bounded from above by 1 for reasonable choices of $\boldsymbol{c}$. The Frobenius norm is bounded from below by the operator norm—indeed, for typical library matrices G, this bound is also fairly tight (as shown in Table 4.1, generally within a factor of 2)—so

$$r_F \leq \frac{\|G\boldsymbol{c}\|}{\|G\|_{op}} \leq \|\boldsymbol{c}\| = 1. \tag{4.5}$$

Meanwhile, r_d is equal to 1 for a single-term relation, and adding terms with reasonable coefficients can only decrease the numerator and hence r_d as well. Thus,

Table 4.1: Ratio of the Frobenius norm and operator norm, $\|G\|_F/\|G\|_{op}$, for the library matrices G corresponding to the four different irreducible representations (scalar, vector, rank-2 symmetric trace-free, and rank-2 antisymmetric) in the cases of the incompressible fluid example of Section 5.1 and the compressible fluid example of Section 5.2.

System	scalar	vector	rank-2 STF	rank-2 AS
Fluid (continuous)	1.44	1.60	1.15	1.30
Fluid (discrete)	1.35	1.42	1.74	1.63

both relative residuals reasonably measure how much better the identified relation is than one with $\boldsymbol{c}$ selected by random chance ($r \approx 1$). In practice, although the dominant-term residual is more physically intuitive because of the connection with the method of dominant balance, it is also more difficult to use, since

1. we do not know at the start of regression which term will appear in the relation,
2. direct comparison of relations containing different terms is challenging, even when one is a proper subset of the other, and
3. it is meaningless for single-term relations.

Therefore, the Frobenius residual is generally preferred.

Occasionally, we may also use a hybrid of the Frobenius and dominant-term residual to compare the quality of different relations. We define the hybrid residual by

$$r_h = \frac{\|G\boldsymbol{c}\|}{\|G_S\|_F}, \tag{4.6}$$

where G_S is the submatrix of G containing only the columns corresponding to the support S, consisting of the indices of nonzero coefficients of $\boldsymbol{c}$. (For single-term relations, $r_h = r_d$, and for $S = [w]$ we have $r_h = r_F$.) Similarly to the dominant-term residual, the hybrid residual is poorly suited for selecting the support S, since it is extremely sensitive to small changes in $\boldsymbol{c}$ – for instance, the regression is encouraged to include columns with large norms but small coefficients to artificially inflate the

denominator in (4.6). Nevertheless, we find that r_h is an extremely useful secondary metric for distinguishing between physically meaningful and meaningless relations (see Section 5.2).

However, even with a good definition of the residual, there is no single best method of resolving the trade-off between parsimony and accuracy. Any k-term relation minimizing r_a or r_F for some value of k is Pareto optimal [85], such that including more terms in the relation will generally decrease the residual at the cost of introducing additional complexity. Hence, the choice of relation is ultimately at the user's discretion. SPIDER can use several criteria to select k:

1. choose the smallest k such that $r_k/r_1 < \epsilon \ll 1$, where r_k is the residual of the best k-term model;

2. choose the largest k such that $r_{k-1}/r_k > \gamma$ for an acceptable residual jump factor $\gamma > 1$ [57, 56];

3. for $k \in [k_{max}]$, choose k minimizing the Akaike information criterion (AIC) [1, 78] or Bayesian information criterion (BIC) [103], with a likelihood estimate using the residual sum of squares (RSS) r_k^2.

It is worth noting that, when criterion 3 uses AIC without the finite sample correction, it is equivalent to criterion 2 with $\gamma = e^{1/m}$. In practice, we generally pick higher values of γ in order to sparsify relations much more aggressively.

It is straightforward to perform cross-validation during regression as an additional precaution against overfitting. Since the integration domains are randomly sampled within the dataset, it is possible to select a random partition of domains into training and test sets, compute the solution of regression $\boldsymbol{c}$ on the training set, and then validate the result by comparing the training residual with the residual of the same relation on the test set.

4.2 Nondimensionalization and Normalization

Even so, there are major pitfalls when searching for a good solution to (4.1): both the identified coefficient vector $\boldsymbol{c}$ and the residual may be very sensitive to the particular definition of the G matrix. For instance, suppose the case in which there exists a good single-term relation describing the system, such as the incompressibility condition $\partial_\alpha u_\alpha = 0$ for fluids. For any real dataset with finite noise or numerical error, this relation cannot hold exactly, and therefore the corresponding residual may be very large or small solely depending on the scaling of this term. **A physics-agnostic regression approach is incapable of resolving whether such a relation is accurate.** Additionally, some rows of G may have disproportionate effects on the residual. Each row corresponds to integrals with respect to a different combination of domain and weight function, which might have scales spanning many orders of magnitude, and these scales are inherited by the corresponding entries of the residual. To the contrary, we would like different data samples to be weighted roughly equally in evaluating the accuracy of the solution.

These observations show that it is crucial to preprocess G in a way that maximizes the odds that we learn physically meaningful relations. As a starting point for preprocessing, it is useful to consider how various simple operations on G would impact an *exact* solution $\boldsymbol{c}_e$ of (4.1). Clearly, all elementary row operations on G leave $\boldsymbol{c}_e$ unchanged, although, as we have seen, they may affect a general least-squares solution. Orthogonal transformations of G satisfy an even stronger property: since they are isometries, $\|G\boldsymbol{c}\|$ remains unchanged for any value of $\boldsymbol{c}$. Finally, dividing any ith column of G by a rescaling factor s_i results in a growth of the ith entry of $\boldsymbol{c}_e$ by the same factor (prior to renormalization of $\boldsymbol{c}$ to unit norm).

As the first preprocessing step in the regression, we perform a physically-informed nondimensionalization of the columns of G. The simplest approach to this would be to define characteristic scales corresponding to any elementary dimensional quantity

involved in each term and then use dimensional analysis to normalize each term. However, there are still subtle issues with this strategy. Specifically, the scales of an observable and its derivatives may be completely different: for example, the pressure of a fluid can have a large mean value and yet undergo only small spatiotemporal fluctuations; indeed, there is a gauge freedom in choosing the mean pressure. To address this concern, SPIDER uses a slightly more subtle nondimensionalization scheme. For each observable y, we denote its root mean square over the whole dataset by m_y and its standard deviation by σ_y. (In principle, one can extend this method to higher moments, but it is usually sufficient to consider only the standard deviation as a simple proxy for the scale of variations in the fields.) In addition, let L and T be characteristic length and time scales of the system; these may be known ahead of time based on physical considerations or derived using the data. For instance, the scales can be taken as the point at which the spatial or temporal correlation function of one of the observables crosses 0.5, or as the ratio between the scale of variation in the observable and of its derivative:

$$L = \frac{\sigma_y}{\langle \|\nabla y\| \rangle}, \quad T = \frac{\sigma_y}{\langle \|\partial_t y\| \rangle} \tag{4.7}$$

Then, for the continuous library, we define the characteristic scales $S[\cdot]$ of the primes and terms according to the following set of rules:

$$S[y] = m_y, \tag{4.8}$$

$$S[\partial_t^i \nabla^j y] = T^{-i} L^{-j} \sigma_y, \tag{4.9}$$

$$S\left[\prod_{i=1}^{n} p_i\right] = \prod_{i=1}^{n} S[p_i] \tag{4.10}$$

where p_i are any primes. (Note that Ref. [56] uses a slightly modified version of the nondimensionalization scheme just described, in which separate length and time

scales are computed for each observable. Both options are reasonable.) In the discrete case, we slightly amend this definition to account for the coarse-graining operator ρ: the scale of primes without derivatives additionally includes a factor of the mean scale of ρ (computed as the mean particle density over the spatial domain), and for primes with derivatives, a factor of the standard deviation of ρ (taken over all grid points after coarse-graining). Finally, after the characteristic scale $S[\mathcal{T}_i]$ of each term $\mathcal{T}_i$ has been computed, the ith column in G is divided by $S[\mathcal{T}_i]$. The dimensional scaling is later reinstated by multiplying each c_i by $S[\mathcal{T}_i]$ and then renormalizing $\boldsymbol{c}$.

In Ref. [52], it was shown empirically that both simple nondimensionalization based on dimensional analysis and the nondimensionalization scheme described here substantially improve noise robustness of model discovery for the Lorenz system of ODEs [73]. For hydrodynamic PDEs or coarse-grained models of MBS, the advantages of the latter scheme are much more pronounced, since the ratio between m_y and σ_y is often more than an order of magnitude if the values of the underlying observable are not centered about zero. (For an observable with zero mean, the root mean square and standard deviation are equal, i.e., $m_y = \sigma_y$.) For instance, Table 4.2 lists m_y and σ_y for each observable in the two systems examined in Chapter 5. Only the observable ρ in the compressible fluid of Section 5.2 is not zero-centered, leading to a large scale difference.

Table 4.2: Characteristic scale of the mean m and variation σ computed for pressure p and flow velocity $\boldsymbol{u}$ in the incompressible fluid of Section 5.1 and the (rescaled) density ρ and particle velocity $\boldsymbol{v}$ of the compressible fluid of Section 5.2.

Observable	m	σ
p	0.084	0.084
$\boldsymbol{u}$	0.28	0.28
ρ	1	0.093
$\boldsymbol{v}$	0.69	0.69

After nondimensionalizing the columns, to put each row on equal footing, we individually normalize each row of G to have norm $\sqrt{w}$ (so that the entries are order

1); as noted previously, this operation preserves exact relations. Finally, for the remaining computations involved in the regression procedure, it is useful to replace the resulting G with the matrix R arising from the thin QR factorization $G = QR$ [53]. Since Q is an orthonormal matrix, $\|G\boldsymbol{c}\| = \|R\boldsymbol{c}\|$ identically, and it is easier to work with R, which is upper triangular and often much smaller ($w \times w$ instead of $h \times w$) than G.

4.3 Iterative Heuristic Algorithms for Regression

In Section 4.1 we have cast the informal task of finding good solutions to (1.2) as the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{c} \in \mathbb{R}^w} \ & \boldsymbol{c}^T \Sigma \boldsymbol{c} \\ \text{s.t. } & \boldsymbol{c}^T \boldsymbol{c} = 1, \\ & \|\boldsymbol{c}\|_0 \leq k \end{aligned} \tag{4.11}$$

where $\Sigma = G^T G$ is a positive semidefinite $w \times w$ matrix and $k \in \mathbb{N}$. (Up to a constant normalization factor, the objective function is proportional to the squares of both the absolute residual r_a and the Frobenius residual r_F.) Of course, if $k = 0$, there is no feasible solution. On the other hand, this optimization problem has a simple solution if $k = w$, namely $\boldsymbol{c} = \boldsymbol{v}_N$, the eigenvector corresponding to the smallest eigenvalue of Σ (or equivalently, the last right singular vector of G). Given a fixed support $S = \{i \mid c_i \neq 0\}$, the optimal $\boldsymbol{c}$ with this support is similarly found as the smallest eigenvector of the submatrix of Σ induced by S.

In all other cases, the problem is nonconvex because of the cardinality constraint on S, with the optimal value monotonically non-increasing with k. Although in principle the optimal solution can be found by iterating over all possible supports S, this is combinatorially intractable unless w is moderate and k small: for instance, if

$w = 100$ and $k = 6$, the total number of possible combinations of exactly 6 terms is $\binom{100}{6} \approx 1.19 \times 10^9$. Thus, it is typically very challenging to solve (4.11) exactly, even with state-of-the-art optimization techniques (these will be examined in more depth in Section 4.4). Instead, it is reasonable to search for approximate solutions via heuristics.

Before continuing, it is important to note that (4.11) is equivalent to the well-studied sparse principle component analysis (PCA) problem

$$\begin{aligned} &\max_{\boldsymbol{c} \in \mathbb{R}^w} \boldsymbol{c}^T \Sigma' \boldsymbol{c} \\ &\text{s.t. } \boldsymbol{c}^T \boldsymbol{c} = 1, \\ &\quad \|\boldsymbol{c}\|_0 \leq k \end{aligned} \tag{4.12}$$

with Σ' also positive semi-definite, under the substitution $\Sigma' = \lambda_1 I - \Sigma$, where λ_1 is the largest eigenvalue of Σ. This is because

$$\boldsymbol{c}^T \Sigma' \boldsymbol{c} = \lambda_1 \boldsymbol{c}^T \boldsymbol{c} - \boldsymbol{c}^T \Sigma \boldsymbol{c} = \lambda_1 - \boldsymbol{c}^T \Sigma \boldsymbol{c} \geq 0. \tag{4.13}$$

Therefore, there already exists a rich literature for solving (4.11). However, passing straight to (4.12) can often result in a much more poorly-conditioned problem. For instance, if (4.11) has many approximate solutions, then Σ may have several very small eigenvalues $\lambda \ll \lambda_1$; in this case, the relative eigenvalue gap is significantly diminished when replacing $\lambda \to \lambda_1 - \lambda$.

In this section, we describe some of the most effective heuristic methods for solving (4.11). First, let us touch briefly on some standard approaches in the literature that have been shown to perform poorly in the present context. The Lasso [111] (also known as basis pursuit denoising) is well known to be neither computationally efficient nor able to recover the correct support reliably [107, 52]. The same is true of sequentially thresholded least squares (STLS) and sequentially thresholded ridge

regression (STRidge), which were introduced in the original formulations of SINDy [20, 99]; in particular, these approaches *never* recover terms with small coefficients, as the thresholding step removes all small coefficients regardless of level of physical significance, and such approaches are known to be suboptimal in the sparse PCA literature [87, 31].

However, many other greedy algorithms do perform very well in practice. All of these algorithms hypothesize a family of approximately optimal heuristic solutions $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_n$ with nested supports $S_1 \subset S_2 \subset \cdots \subset S_n$. (Unlike in STLS and STRidge, where these supports may have arbitrary cardinalities, here we require $|S_i| = i$.) Then, given a known heuristic solution $\boldsymbol{c}_i$, one can search for a new approximately optimal support with either one fewer term, S_{i-1}, or one more term, S_{i+1}, by iterating over all possible terms and removing/adding the term that minimizes the optimal objective value, $\boldsymbol{v}_N^T \Sigma \boldsymbol{v}_N$, for the new support. This family of approaches was first introduced and motivated in the sparse PCA literature [87, 31], only to be later rediscovered in the context of homogeneous sparse regression in SPIDER [57]. A theoretical motivation for this approach is given by Cauchy's interlacing theorem [123]: for a $n \times n$ symmetric matrix A with eigenvalues $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(\mathrm{A})$ and a $k \times k$ principal submatrix A_k with eigenvalues $\lambda_i(A_k)$,

$$\lambda_i(A) \geq \lambda_i(A_k) \geq \lambda_{i-n+k}(A), \tag{4.14}$$

and in particular,

$$\lambda_1(A) \geq \lambda_1(A_{n-1}) \geq \lambda_2(A) \geq \cdots \geq \lambda_{n-1}(A) \geq \lambda_{n-1}(A_{n-1}) \geq \lambda_n(A). \tag{4.15}$$

It follows that optimizing over all possible supports with one fewer (more) term provides a good upper (lower) bound for the optimal value of the objective function of (4.11) corresponding to the present support.

Several types of greedy algorithm are considered in Ref. [87] and commonly in use in SPIDER:

1. Forward synthesis: start with a small support S and iteratively add terms one at a time.

2. Backward elimination: start with a *large* support S and iteratively *remove* terms.

3. Backward-forward regression: combine passes of forward synthesis and backward elimination. In [87], it is recommended to compare the solutions obtained in an independent backward and forward pass and select the best of the two. In SPIDER, we instead generally repeat several iterations of computationally inexpensive backward and forward passes for moderate k (e.g., $1 \leq k \leq 10$ or $1 \leq k \leq 20$), each starting from the support selected at the end of the previous pass, and then keep the supports corresponding to the solutions with the lowest residuals. (Other criteria for switching between backward and forward passes are possible, such as comparing values of the residual for successive k.) One example of a problem where forward synthesis identifies suboptimal solutions (by a factor of about 1000) that are improved on the backward pass is examined in Section 3.6 of [52].

Of these three options, backward-forward regression is usually the most robust, although it is more computationally expensive than its counterparts. Empirically, the backward and forward passes have different weaknesses. Backward elimination is substantially more computationally expensive, with the naive implementation starting from $k = n$ achieving a time complexity of $O(n^4)$ vs a conservative worst-case estimate of $O(n^3)$ for forward synthesis [87]. Both passes can be made substantially more efficient using a bisection search for the smallest eigenvalues of the principal submatrices of Σ, which was introduced in [49] by the name of SPRINT; empirical results indicate

time complexities scaling approximately as $O(n^{3.38})$ and $O(n^{1.65})$. Additionally, the backward and forward passes display different patterns of becoming trapped by local minima: backward elimination may discard necessary terms too early (i.e., for large k), while forward synthesis requires a strong initial guess in order to avoid starting with the wrong terms for small k [49]. These problems are largely alleviated through good initialization strategies, as discussed in Section 4.5: in particular, backward elimination with the appropriate initialization almost always recovers a good solution in practice.

It is also worth noting that backward elimination often does not recover single-term relations, as these very small terms have a minimal contribution to the dominant balance of larger relations and tend to be eliminated during earlier iterations. A straightforward solution is to iterate over single-term relations separately and simply select the best result as the $k = 1$ solution.

A different approach to solving (4.11) is to use a truncated power method [129]. The original TPower method for finding the largest k-sparse eigenvector of a symmetric matrix generalizes standard power iteration

$$\boldsymbol{x}_t = \frac{A\boldsymbol{x}_{t-1}}{\|A\boldsymbol{x}_{t-1}\|} \tag{4.16}$$

by adding a hard thresholding step at each iteration, in which all but the k largest entries of $\boldsymbol{x}_t$ in terms of absolute value are set to zero. This algorithm can be adapted to our task of finding the *smallest* eigenvector by replacing power iteration with either inverse iteration or Rayleigh quotient iteration [96]. Inverse iteration is slightly more robust but Rayleigh quotient iteration tends to converge more quickly. In either case, the exact solution $\boldsymbol{v}_N$ and objective value λ_N for $k = w$ can serve as initial guesses for the sparse "eigenvector" and final Rayleigh quotient, $\boldsymbol{x}_0$ and μ_0. Additionally, the convergence of each iteration can be improved by taking the next eigenvector estimate

$\boldsymbol{x}_t$ as the exact smallest eigenvector of the principal submatrix corresponding to the current support estimate, rather than simply hard thresholding – this is similar to the strategy recommended in [87]. Algorithm 2 provides pseudocode for SPIDER's truncated Rayleigh quotient iteration (TRQI); truncated inverse iteration (TII) is identical except the Rayleigh quotient estimate μ is never updated.

Algorithm 2: Truncated Rayleigh Quotient Iteration (TRQI)

Data: $\Sigma \in \mathbb{S}_+^w, k \in \mathbb{N}, \boldsymbol{x}_0 \in \mathbb{R}^w$
Result: x_t
$t \leftarrow 0$;
$\boldsymbol{x}_0 \leftarrow \boldsymbol{x}_0 / \|\boldsymbol{x}_0\|$;
repeat
  $\mu_t = \boldsymbol{x}_t^T \Sigma \boldsymbol{x}_t$;
  $\boldsymbol{y}_t \leftarrow$ solution of $(\Sigma - \mu_t I)\boldsymbol{y}_t = \boldsymbol{x}_t$;
  $S_t = \text{supp}(\boldsymbol{y}_t, k)$, the k indices with the largest absolute values;
  $\Sigma_t =$ principal submatrix of Σ induced by S_t;
  $\boldsymbol{x}_{t+1} =$ smallest eigenvector of Σ_t, padded with zeros for indices in $[w] \backslash S_t$;
  $t \leftarrow t + 1$;
until *convergence*;

Although, unlike greedy regression, power methods only furnish a solution for a single value of k, this solution can be used as a very good initial guess for the next iteration with $k - 1$. Because power iteration is already very computationally inexpensive and convergence from the previous solution generally requires few additional iterations, it is still able to solve (4.11) for all values of k quickly.

In an empirical analysis, Ref. [9] compared the performance of greedy forward regression and truncated power iteration in recovering the true optimal support with two more sophisticated mixed-integer optimization methods, which are the subject of the next section. For a 150×150 matrix Σ corresponding to 100 true PCA components corrupted with random uniform noise, given a ten minute time budget, both heuristic algorithms performed on par with the mixed-integer methods while requiring 10 to 1000 times less computation time.

There are a few other approaches for solving sparse PCA in the statistics literature that are outside the scope of this dissertation but may warrant further exploration as alternative regression methods. One of these is approximate message-passing [34, 44], which is a generalization of the truncated power methods described above with a more flexible thresholding function g_t and the addition of a "memory term" $b_t = f(\boldsymbol{x}_t)$ aimed at decoupling statistical error across successive iterations. This formulation leads to the iteration step

$$\hat{\boldsymbol{x}}_t = g_t(\boldsymbol{x}_t), \quad \boldsymbol{x}_{t+1} = A\hat{\boldsymbol{x}}_t - b_t\hat{\boldsymbol{x}}_{t-1} \tag{4.17}$$

Other recent work has used proximal gradient methods to approximate the solution of sparse PCA [27].

Finally, it is occasionally useful to search for the best relation containing a certain specific term $\mathcal{T}_i$. For instance, one may look for an explicit evolution equation for a specific quantity q and hence be interested in the term $\partial_t q$ (an approach taken by methods such as SINDy). One option is to find sparse least-squares solutions of the inhomogeneous version of (1.2), namely

$$G_{\setminus i}\tilde{\boldsymbol{c}} = \boldsymbol{g}_i \tag{4.18}$$

where g_i is the ith column of G and $G_{\setminus i}$ is the submatrix of G formed by discarding this column. This leads to the following alternative optimization problem:

$$\begin{aligned} &\min_{\tilde{\boldsymbol{c}}\in\mathbb{R}^w} \|G_{\setminus i}\tilde{\boldsymbol{c}} - \boldsymbol{g}_i\| \\ &\text{s.t. } \|\tilde{\boldsymbol{c}}\|_0 \leq k \end{aligned} \tag{4.19}$$

Both greedy regression and truncated power iteration can be adapted to this case with only minor modifications to their iteration steps. A simpler shortcut is to retain the formulation (4.11) and require the support at each step to contain the desired term $\mathcal{T}_i$.

4.4 Certifiably Optimal Mixed-Integer Regression

While simple and usually capable of finding good solutions in practice, the algorithms considered in the previous section are in a way still fundamentally flawed: they neither provide any concrete guarantees that their solutions are anything more than local minima, nor is there any theoretical result showing that these algorithms output optimal or near-optimal solutions with any probability. On the other hand, modern optimization software for solving convex and/or mixed-integer optimization problems is extremely computationally efficient and generally employs primal-dual methods [67], which do provide certificates of optimality or near-optimality; this is done by simultaneously solving the original problem, generating candidate solutions, and its dual, which furnishes bounds on the optimal objective value that can be achieved by any solution. Moreover, exact formulations cannot become stuck in local minima, as in theory the solver must converge to the optimal solution if given enough computation time and memory. (However, for large and challenging instances, the scaling of computational effort may be practically prohibitive.) In this section, we present a few closely related approaches based on mixed-integer optimization that leverage these desirable properties of optimization solvers to find certifiably optimal or near-optimal solutions to the sparse regression problems (4.11) and (4.19) across a range of problem sizes.

Applying mixed-integer optimization to sparse regression-based equation learning has already been explored briefly in Ref. [10]. This study casts the inhomogeneous version of the regression problem, (4.19), as a mixed-integer second-order cone problem that encodes the sparsity constraint via type-1 specially ordered sets (SOS-1) [13], as derived in Ref. [11]. Besides its focus on the simpler to solve but less flexible inhomogeneous formulation of sparse regression, this study has a few other noteworthy weaknesses:

- The mixed-integer approach is compared only to a subset of greedy regression algorithms that are known to have poor recovery of the true support, which is reflected in their underwhelming performance in the reported testing.

- The SOS-1 formulation is scalable only up to libraries with at most a few thousand terms, which may soon be eclipsed by more computationally efficient implementations of sparse regression (for instance, a library of 650 terms was evaluated on 1376 integration subdomains in a modest 18 hours in [51]).

However, these latter issues are likely fairly simple to address: alternative formulations that scale to $O(10^5)$ or more terms have performed well in a general sparse regression setting [12].

The more critical limitation of this work is its focus on the fundamentally restrictive inhomogeneous problem (4.19). SPIDER requires instead solving (4.11), which as we saw in the previous section is equivalent to sparse PCA and thus necessitates formulation as a more challenging mixed-integer semidefinite problem (MISDP) [32, 6, 9]. The rest of the present section will summarize such an approach rooted in the sparse PCA equivalence, which we originally developed independently of Ref. [10].

The first key insight is that, as observed in Ref. [32], defining a new positive semidefinite matrix $X = \boldsymbol{c}\boldsymbol{c}^T$ allows us to rewrite (4.11) as the equivalent rank-constrained semidefinite problem

$$\min_{X \in S_+^w} \langle X, \Sigma \rangle \text{ s.t. } \text{tr}(X) = 1,\ \|X\|_0 \leq k^2,\ \text{Rank}(X) = 1 \tag{4.20}$$

(where S_+^w is the set of $w \times w$ positive semidefinite matrices and $\langle X, \Sigma \rangle = \text{tr}(X^T \Sigma)$ is the Frobenius inner product). We may model whether each X_{ij} is nonzero by binary variables $z_i \in \{0, 1\}$ with the big-M constraints $-M_{ij} z_i \leq X_{ij} \leq M_{ij} z_i$. Note that, by symmetry, either of $z_i = 0$ or $z_j = 0$ implies $X_{ij} = 0$. Furthermore, by a direct adaptation of Theorem 1 in Ref. [9] (obtained by swapping max and min and upper

bounds with lower bounds), the following MISDP without any rank constraint has the same optimal solution as (4.20):

$$\begin{aligned} \min_{\boldsymbol{z}\in\{0,1\}^w} \min_{X\in S_w^+} \; & \langle X, \Sigma \rangle \\ \text{s.t.} \; & \sum_{i=1}^{w} z_i \leq k, \\ & \operatorname{tr}(X) = 1, \\ & |X_{ij}| \leq M_{ij} z_i \quad \forall i,j \in [w], \\ & \sum_{j=1}^{w} |X_{ij}| \leq \sqrt{k} z_i \quad \forall i \in [w] \end{aligned} \tag{4.21}$$

where $M_{ii} = 1$ and $M_{ij} = \frac{1}{2}$ for $i \neq j$. The problem (4.21) can be solved directly by branch-and-cut algorithms [64] only for very modest problem sizes, but Ref. [9] derived an alternative reformulation as a saddle-point problem that can be solved by outer approximation [36]. This latter approach was generally able to solve sparse PCA test problems with w in the 100s and k up to 10 exactly within a 10-minute time budget and obtain certificates of near-optimal solutions with an optimality gap within a few percent in the remaining cases; however, it does not scale well to w in the 1000s. Moreover, this approach requires a special-purpose solver since outer approximation methods are not implemented in standard solvers.

Prioritizing the ability of our regression method to converge to a near-optimal solution over a reasonable time horizon, rather than certify exact optimality while failing to scale to larger problems, we instead recommend an adaptation of the greedy rounding method also developed in Ref. [9] for regression problems featuring 100 or more terms. (In Ref. [9], the greedy rounding method was empirically observed to identify the optimal solutions along with certificates of near-optimality for all test cases considered with $w = 50$ or fewer variables, so arguably it is an excellent option for all reasonable problem sizes.) This method relies on the fact that the following

convex semidefinite problem is a valid relaxation of (4.20):

$$\begin{aligned}
\min_{\boldsymbol{z}\in[0,1]^w} \min_{X\in S_w^+} \; & \langle X, \Sigma\rangle \\
\text{s.t.} \; & \sum_{i=1}^{w} z_i \leq k, \\
& \mathrm{tr}(X) = 1, \\
& |X_{ij}| \leq M_{ij} z_i \quad \forall i,j \in [w], \\
& \sum_{i,j=1}^{w} |X_{ij}| \leq k, \\
& \sum_{j=1}^{w} X_{ij}^2 \leq z_i X_{ii} \quad \forall i \in [w].
\end{aligned} \tag{4.22}$$

To show this, first, note that we have convexified the constraints on the z_i's such that they merely need to lie in the range $[0,1]$. Second, let $S \in \mathbb{R}^{w\times w}$ be the matrix with 1s exactly at the entries in the support of X and 0s elsewhere. Then, by application of the Cauchy-Schwarz inequality to X and S, we have

$$\begin{aligned}
\sum_{i,j=1}^{w} |X_{ij}| \leq \|S\|_F \|X\|_F = \sqrt{k^2}\sqrt{\mathrm{tr}(X^T X)} = \\
= k\sqrt{\mathrm{tr}(\boldsymbol{c}^T \boldsymbol{c}\boldsymbol{c}\boldsymbol{c}^T)} = k\sqrt{\mathrm{tr}(X)} = k.
\end{aligned} \tag{4.23}$$

Finally, the last set of constraints follows by summing over j the constraints $X_{ij}^2 \leq z_i X_{ii} X_{jj}$, which derive from the fact that the 2×2 principal minors of X must be positive semidefinite together with $\mathrm{tr}(X) = 1$.

As discussed in Ref. [8], we may further relax the constraint $X \in S_w^+$ by replacing it with the already included constraints $X_{ij}^2 \leq z_i X_{ii} X_{jj}$ and a small number of cutting planes of the form $\langle X, \boldsymbol{v}_t \boldsymbol{v}_t{}^T\rangle \geq 0$, where each $\boldsymbol{v}_t$ corresponds to the most violated constraint by the previous optimal solution X_{t-1} taking into account only the cuts up to $\boldsymbol{v}_{t-1}$. (In fact, $\boldsymbol{v}_t$ is simply the eigenvector corresponding to the most negative eigenvalue of X_{t-1}.) In doing so, we obtain a convex second-order cone problem

(SOCP) that can be solved extremely efficiently by standard optimization solvers such as `Gurobi`. Once the optimal solution of the SOCP relaxation of (4.22) has been identified, an approximate solution to (4.20) can be obtained by setting the k largest entries of $\boldsymbol{z}$ to 1 and the rest to 0, followed by solving a much simpler convex semidefinite program with fixed $\boldsymbol{z}$

$$\min_{X \in S_+^w} \langle X, \Sigma \rangle \text{ s.t. } \mathrm{tr}(X) = 1,\ X_{ij} = 0 \text{ if } z_i z_j = 0\ \forall i, j \in [w], \tag{4.24}$$

completing the greedy rounding procedure.

It is worth nothing that, as with the truncated Rayleigh quotient iteration method discussed in the previous section, there are fairly straightforward options for warm-starting all of these mixed-integer optimization-based methods to compute solutions for various k values. One possibility is to take as a starting point for sparsity $k+1$ the k-sparse solution, which is always a feasible solution of the less restrictive $(k+1)$-sparse optimization problem. Another warm-start strategy that does not require solutions for other values of k is to use the output of truncated inverse iteration (TII), which can be evaluated extremely quickly, as an initial guess – the corresponding strategy with truncated power iteration (TPower) was used in Ref. [9].

The current Python implementation of SPIDER, `PySPIDER`, uses greedy backward-forward regression rather than the mixed-integer approach described in this section, as it achieves comparable performance [9] but is considerably simpler to implement. The modular structure of `PySPIDER` allows mixed-integer regression to be added in the future.

4.5 Initializing Regression

Because of the nonconvexity of the optimization problem (4.11), all of the methods described earlier in this chapter benefit immensely from access to an approximate

solution for some sparsity value k that can serve as a good initial guess. In this section, we will describe various simple initialization strategies and benchmark their performance when combined with greedy backward-forward regression.

Rather than considering the values of the full coefficient vector $\boldsymbol{c}$, it is sufficient to focus on the choice of support S, since the corresponding optimal $\boldsymbol{c}$ is given by the last singular vector of the submatrix of G induced by S. The most straightforward initialization is the dense solution corresponding to the support $S = [w]$. Trivially, its advantages are that this initial guess is optimal for $k = w$ and it is already a superset of the correct set of terms for all $k < w$. On the other hand, since we are generally interested in $k \ll w$, greedy methods will usually take many iterations to reach the desired small values of k from this initial guess, and one might expect many locally optimal but globally suboptimal choices in the long term to occur during this sparsification trajectory. However, we will see that while greedy iteration from this initialization is computationally expensive, it is in fact empirically extremely robust.

A more computationally efficient variation of this initialization is instead to select the initial support S by running one of the truncated power methods introduced in Section 4.3 for an intermediate sparsity value $k_m < w$ greater than the desired range of k for the equations that we would like to discover. In effect, this allows us to screen the full set of w candidate terms and focus the search on the k_m most promising terms before further continuing subset selection. Although faster, this initialization turns out to be less effective on average for greedy iteration in our testing.

Finally, we may instead try to look for an initialization that is a *subset* of the full relation. Thus, a third approach is to choose the initial guess by solving (4.11) exactly for very small k_s (usually 2 or 3) by performing a combinatorial search over all k_s-term subsets of the library. This strategy is both efficient and allows us to start from an exact solution; unfortunately, we will see that greedy forward synthesis

is sometimes less robust than backward elimination, and initializations with small supports can rarely lead to catastrophically bad solutions.

In the remainder of this section, we will consider the performance of greedy regression using each of these initialization strategies on synthetic data, deferring a similar discussion for realistic SPIDER test cases to Chapter 5. First, let us discuss how to generate synthetic data $G \in \mathbb{R}^{h \times w}$ such that $G\boldsymbol{c}^* = 0$ holds exactly or approximately for any fixed true $\boldsymbol{c}^*$. Indeed, for this equation to hold exactly, it is sufficient to let each row $\boldsymbol{d}_i$ of G be a random vector orthogonal to $\boldsymbol{c}^*$; for instance, we may take $\tilde{\boldsymbol{d}}_i$ with i.i.d. standard normal entries and

$$\boldsymbol{d}_i = \tilde{\boldsymbol{d}}_i - \frac{\tilde{\boldsymbol{d}}_i \boldsymbol{c}^*}{\|\boldsymbol{c}^*\|^2} (\boldsymbol{c}^*)^T. \tag{4.25}$$

To generate a noisy version of the problem, we normalize each row to unit norm and then define $G' = G + \frac{\sigma}{\sqrt{w}} U$, where the entries of U are i.i.d. standard normal and σ is the noise level (the inverse of the signal-to-noise ratio).

For testing, we take 100 random realizations of $G \in \mathbb{R}^{150 \times 100}$,

$$\boldsymbol{c}_i^* = \begin{cases} \frac{11-i}{10}, & 1 \leq i \leq 10 \\ 0, & 11 \leq i \leq 100 \end{cases}, \tag{4.26}$$

and $\sigma = 1/8$. Two important performance metrics for each method are the mean fraction $\bar{F}$ of the true support $\{1, \ldots, 10\}$ of $\boldsymbol{c}_i^*$ that is recovered by the estimated solution $\boldsymbol{c}$ of (4.11) with sparsity $k = 10$, as well as the mean $\bar{R}$ of the ratio of the residuals achieved by $\boldsymbol{c}$ and the optimal solution $\boldsymbol{c}_T$ with the true support,

$$R = \frac{\|G\boldsymbol{c}\|}{\|G\boldsymbol{c}_T\|}. \tag{4.27}$$

(Due to the added noise term U, $\boldsymbol{c}_T$ is not necessarily equal to $\boldsymbol{c}^*$, and so we generally have $\|G\boldsymbol{c}_T\| < \|G\boldsymbol{c}^*\|$. Moreover, for sufficiently large σ, we may have overfitting corresponding to $R < 1$, because the optimal support is no longer the support of $\boldsymbol{c}^*$; we have chosen σ as large as possible without R being significantly less than 1 for any of the approximate solutions identified.) We also examine the number of realizations such that $R < 1$ (i.e., the method identified a better support for the noisy synthetic data than the ground truth support) and the highest value of R (worst performance).

For each realization, we solve (4.11) approximately for the following choices of initializations:

1. full support ($S = [w]$)
2. truncated inverse iteration (TII) with $k_m = 10, 20$
3. truncated Rayleigh quotient iteration (TRQI) with $k_m = 10, 20$
4. combinatorial search with $k_s = 2, 3$

Additionally, for all methods except full support initialization, we test whether the solutions can be improved by different numbers P of greedy backward-forward passes (with $P = 1$ corresponding to only one backward or forward pass). Each pass ranges over $2 \leq k \leq 20$, with an initial direction of backward elimination for the truncated power methods and forward synthesis for the combinatorial search. For the full support initialization, we complete only one backward pass, which starts from $k = w = 100$. For each method, we report the total execution time across all 100 realizations on a 2024 MacBook Air M3.

The results for $P = 1$ are summarized in Table 4.3. The performance of simple backward iteration from the full support is excellent: in only 6 of 100 cases did it fail to recover the exact true support; moreover, 3 of these cases were due to it finding a better solution for the noisy data, and the worst value of R across the 3

Table 4.3: Support recovery, residual ratio, number of realizations where $R < 1$, worst-case R, and execution time for various initialization strategies. These include using full support, truncated inverse iteration (TII), truncated Rayleigh quotient iteration (TRQI), and combinatorial search (CS) after one backward or forward pass of greedy search. For methods other than full support, the sparsity k of the initialization is shown in parentheses.

Metric	full	TII(10)	TII(20)	TRQI(10)	TRQI(20)	CS(2)	CS(3)
$\bar{F}$	0.994	0.950	0.986	0.939	0.981	0.964	0.994
$\bar{R}$	1.0002	1.027	1.0027	1.037	1.0044	1.11	0.9998
# of $R < 1$	3	0	2	0	2	6	6
worst-case R	1.01	1.13	1.05	1.16	1.05	4.86	1
time (s)	265	0.9	2.0	1.0	2.6	25	467

cases where it found suboptimal supports was only 1.01. Truncated power iteration, while about two orders of magnitude faster, more frequently made small mistakes and recovered only 9/10ths of the true support. For instance, TII with $k_m = 10$ correctly identified $\bar{F} = 0.95$ of the true support on average, which is the same performance as immediately applying the thresholding step of taking the 10 largest entries by absolute value of the last singular vector. (This specific TII initialization with less than 2 backward-forward passes is equivalent to identifying an approximate solution via TII without any greedy iteration.) Moreover, TRQI performed no better than TII on all metrics.

The results for initialization via combinatorial search are qualitatively different. Combinatorial search among 2-term initializations did not recover the ground truth support in 11 of 100 realizations; of these, 6 were cases in which better solutions were found and a different set of 3 were catastrophic failures in which $R > 3.5$ and only 0 or 1 of the terms in the true support were recovered. Combinatorial search among 3-term initializations, while the most computationally expensive method, also performed the best according to all other criteria. Indeed, all 6 cases where the true support was not recovered were due to the greedy iteration finding better solutions for the noisy data!

In some cases, the solution after one pass may be improved by additional backward/forward passes. For instance, Table 4.4 lists the performance of TII initialization with $k_m = 10$ for $P = 1, 2, 3, 4$. In this case, the second and third passes lead to improvements in the mean performance, but subsequent passes confer no further benefit: each even pass returns the same output as the second pass and each odd pass, the same as the third. The third pass performs better than full support initialization and nearly as well as 3-term combinatorial search with $P = 1$, while remaining more than an order of magnitude more computationally efficient than either of these alternatives. (Interestingly, these are also the same solutions as found by TII with $k_m = 20$ after multiple passes.) Similarly, some of the solutions identified after initializing with 2-term combinatorial search are slightly improved by a second pass, but the third pass offers no benefit compared to only a single pass. In particular, no number of passes is sufficient to repair any of the 3 catastrophic failures of this initialization to find a good solution.

Table 4.4: Support recovery, residual ratio, number of realizations where $R < 1$, worst-case R, and execution time for greedy iteration initialized with truncated inverse iteration (TII) with support size $k_m = 10$, after varying numbers P of backward-forward passes.

Metric	$P = 1$	$P = 2$	$P = 3$	$P = 4$
$\bar{F}$	0.950	0.992	0.995	0.992
$\bar{R}$	1.0027	1.0018	0.9998	1.0018
# of $R < 1$	0	6	5	3
worst-case R	1.13	1.13	1	1.13
time (s)	0.9	10.5	12.9	20.8

In addition to comparing different initializations, the empirical results presented in this section confirm that neither of backward and forward greedy regression is always strictly better than the other. We recommend performing additional backward-forward passes until the solution is unchanged after further pairs of passes and then selecting the solution with the lowest residual among all of the passes. This is a generalization of the strategy previously identified in Ref. [87]. For instance, selecting the

best solutions across the first 3 passes for TII initialization with $k_m = 10$ identifies the same set of supports for all 100 realizations as the best single-pass method, 3-term combinatorial search initialization, but requires less than 0.5% of the computation time.

With the results of this section as well as Chapter 5 in mind, we suggest primarily employing TII initialization and selecting the best solution from several backward-forward passes, as this method is one of the most computationally inexpensive options but does not sacrifice solution quality. The results obtained by TII can be validated by full support initialization for modest library sizes ($w \leq 200$), while we will see that combinatorial search initialization is unreliable in practical applications.

Chapter 5

Illustrative Applications of SPIDER

Finally, in this chapter, we present two illustrative examples in which SPIDER is used to discover a complete set of governing equations describing numerically generated data. In Section 5.1, these data represent an incompressible fluid flow in the continuum limit, and in Section 5.2, they correspond to a compressible fluid composed of discrete interacting particles. These examples validate SPIDER's ability to learn quantitatively accurate continuum descriptions of the dynamics and provide general guidance for employing SPIDER in other applications.

5.1 Continuous: Incompressible Fluid Flow

First, we test SPIDER's ability to learn the governing equations of a continuous system where the ground truth is already known. It is worth noting that SPIDER has previously been validated to an extent using several different types of numerically generated data describing solutions of canonical nonlinear PDEs (see Section 1.3). However, the results presented in this dissertation are novel in several ways:

- For the first time, a version of SPIDER with fully systematic and automatic library generation has been used.

- A comprehensive set of independent equations has been identified automatically.
- We validate the ability of SPIDER to discover known second-rank equations with negligible error.

Specifically, we consider a high-fidelity simulation of freely decaying turbulence described by the Navier-Stokes equation (2.12) on a two-dimensional square domain with periodic boundary conditions and a kinematic viscosity of $\nu = 10^{-4}$ in nondimensionalized units. (Note that these boundary conditions are not amenable to the representations used in 2.3, which are fundamentally local, since they relate velocities at opposite sides of the simulation domain. Due to limitations of the `PySPIDER` codebase at the time of writing, we do not identify boundary conditions in this chapter; however, this has been done for a similar dataset with no-slip boundary conditions using a manually generated library in Ref. [56].) The single-precision velocity data $\boldsymbol{u}$ of the simulation were generated using a pseudospectral solver implemented in JAX-CFD [65] on a $L \times L = 2\pi \times 2\pi$ spatial domain over a time horizon $T = 5$ (both in nondimensionalized units), evaluated on a 2048×2048 spatial grid with timestep $\Delta t = 10^{-3}$. The data input to SPIDER were subsampled to a $1024 \times 1024 \times 100$ grid, and the pressure data p were computed by solving the pressure-Poisson equation (2.13) spectrally on the subsampled grid. The correlation time and length of the velocity field are $\tau \sim 0.5$ and $\ell \sim 0.2$, respectively, and the root mean squared velocity is $m_u = 0.282$, corresponding to a high Reynolds number $Re = m_u L/\nu \approx 17700$. Figure 5.1 shows 4 representative snapshots of the vorticity field $\omega = \partial_x u_y - \partial_y u_x$ that are evenly spaced in time, displaying the fine-scale structure that is characteristic of turbulence.

We now discuss the SPIDER hyperparameters used for this dataset. The dynamics in the bulk of the fluid are described by the incompressibility condition (2.11) and the Navier-Stokes equation (2.12); there are many additional implied equations that also describe the data, with the most well-known being the pressure-Poisson equation

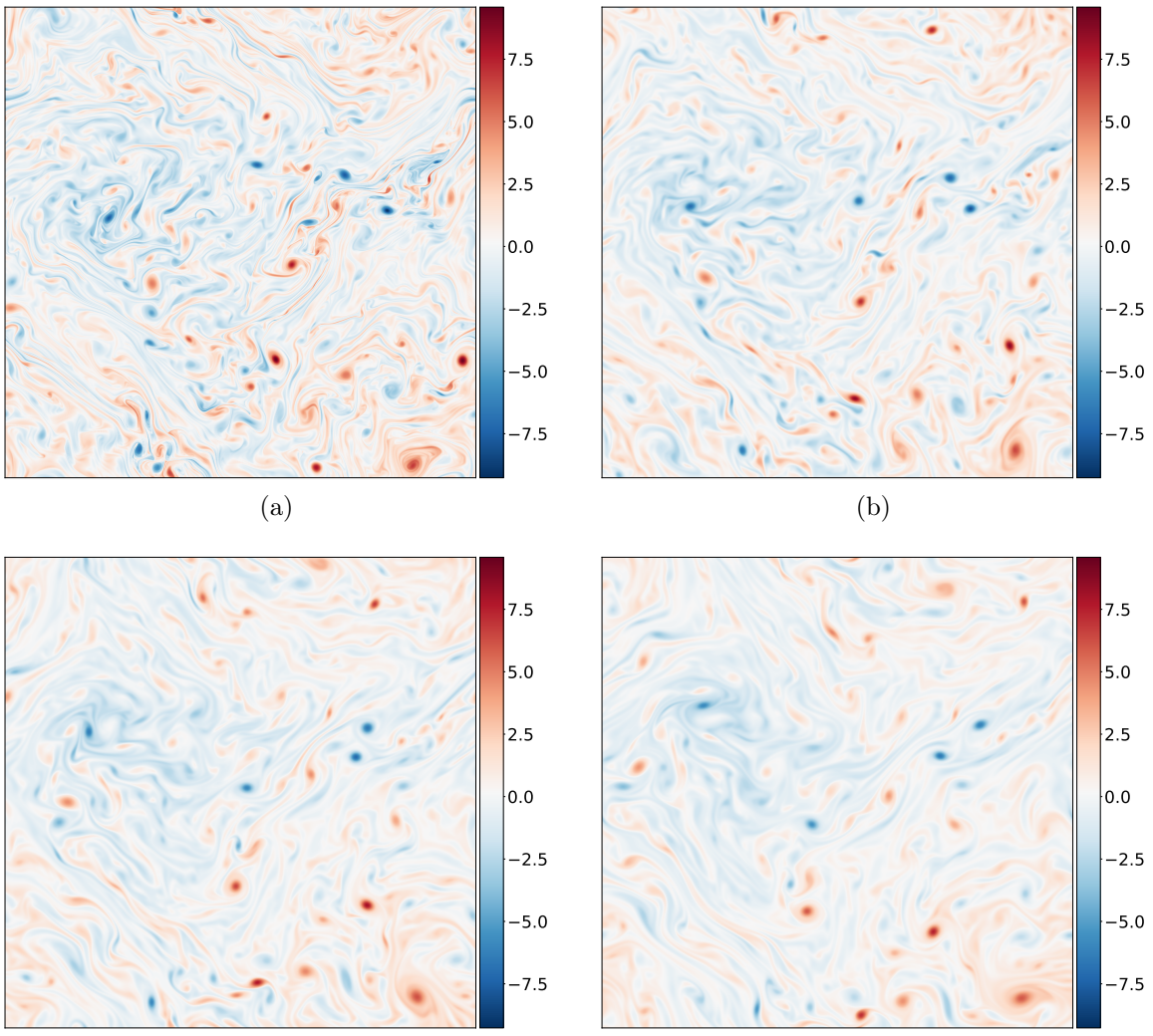


Figure 5.1: Snapshots of the vorticity field $\omega = \partial_x u_y - \partial_y u_x$ for the JAX-CFD incompressible fluid flow dataset at times $t =$ (a) 0, (b) 1.5, (c) 3, (d) 4.5.

(2.13) and the energy equation (2.14). As discussed in Section 2.2, all of these relations can be expressed using terms of complexity 4 or less involving only the velocity vector $\boldsymbol{u}$ and scalar pressure p. Thus, we direct SPIDER to generate and identify all relations in the scalar, vector, rank-2 antisymmetric, and rank-2 STF hydrodynamic libraries with complexity threshold 4, which contain 43, 51, 23, and 38 terms, respectively. (Naturally, for an unknown system, we would not have advance knowledge of the correct complexity threshold, but one can be selected by trial and error.)

The size of integration domains was set to $L_x \times L_x \times L_t = 40 \times 40 \times 20$ grid points or $0.25 \times 0.25 \times 1 \approx \ell \times \ell \times 2\tau$ in nondimensional units and multiples of the correlation length and time, respectively. This choice was found to minimize the residuals of the most difficult to detect ground truth relations (i.e., those with the highest residuals), although these residuals are fairly robust with respect to the choice of L_x and L_t. Since the largest library size is 49, we randomly sample $h = 100$ integration domains across the dataset to ensure that $h \gg w$. (Again, the specific choice of h is not crucial to the success of the regression; in particular, adding additional integration domains can only be helpful.) We use the unmodulated factorizable polynomial weight functions defined in (3.6)–(3.7) with $q = 12$.

In contrast, as we will soon see, the choice of initialization (which we discussed in detail in Section 4.5) does impact which relations are learned. As recommended in Section 4.1, we measure the accuracy of relations using the Frobenius relative residual r_F. Following Ref. [56], the residual jump criterion (option 2 in Section 4.1) with $\gamma = 1.5$ is used to select the optimal sparsity for each relation. When using full support initialization or TII initialization (with $k_m = 10$), the same set of fundamental equations, shown in Table 5.1, is discovered for any residual threshold r_{max} (see Section 3.4) in the range $[4 \times 10^{-7},\ 5 \times 10^{-6}]$ and after any number of backward-forward iterations over $1 \leq k \leq 10$. (The 40 implied equations identified

by SPIDER are not shown.) Additionally, Table 5.2 lists in order the relative errors of the coefficients in the multi-term equations.

#	sublibrary	equation	r	type
1.	rank 0, $\mathcal{C} = 2$	$\partial_\alpha u_\alpha = 0$	7.7×10^{-14}	F
2.	rank 1, $\mathcal{C} = 3$	$\partial_\alpha p + u_\beta \cdot \partial_\beta u_\alpha - 10^{-4} \cdot \partial_\beta^2 u_\alpha + \partial_t u_\alpha = 0$	2.8×10^{-8}	F
3.	rank 0, $\mathcal{C} = 4$	$\partial_\alpha^2 p + \partial_\alpha u_\beta \cdot \partial_\beta u_\alpha = 0$	7.3×10^{-8}	I
4.	rank 2 AS, $\mathcal{C} = 4$	$\mathrm{AS}[\partial_\alpha u_\gamma \cdot \partial_\gamma u_\beta] = 0$	1.8×10^{-7}	G+I
5.	rank 2 AS, $\mathcal{C} = 4$	$\mathrm{AS}[u_\alpha \cdot \partial_\gamma^2 u_\beta - 10^{-4} \cdot \partial_\alpha \partial_\gamma^2 u_\beta + \partial_t \partial_\alpha u_\beta] = 0$	2.8×10^{-7}	G+I
6.	rank 2 STF, $\mathcal{C} = 4$	$\mathrm{STF}[\partial_\alpha u_\gamma \cdot \partial_\gamma u_\beta] = 0$	1.8×10^{-7}	G+I
7.	rank 2 STF, $\mathcal{C} = 4$	$\mathrm{STF}[\partial_\alpha u_\gamma \cdot \partial_\beta u_\gamma + \partial_\gamma u_\alpha \cdot \partial_\gamma u_\beta] = 0$	2.5×10^{-7}	G+I
8.	rank 2 STF, $\mathcal{C} = 4$	$\mathrm{STF}[\partial_\alpha \partial_\beta p + u_\alpha \cdot \partial_\gamma^2 u_\beta + u_\gamma \cdot \partial_\alpha \partial_\beta u_\gamma - 10^{-4} \cdot \partial_\alpha \partial_\gamma^2 u_\beta + \partial_t \partial_\alpha u_\beta] = 0$	3.5×10^{-7}	G+I

Table 5.1: List of all fundamental equations identified by sparse regression for JAX-CFD dataset using optimal SPIDER hyperparameters. In addition to the equation itself, we list the irreducible representation and complexity $\mathcal{C}$ of the sublibrary in which it is first identified, its residual, and its interpretation: whether it is a fundamental equation for incompressible fluid flows (F), a general identity (G), an implied equation (I), or a combination of multiple types of equation.

#	coeff. rel. errors
2.	$-6 \times 10^{-8},\ -8 \times 10^{-8},\ -8 \times 10^{-7},\ 0$
3.	$-2 \times 10^{-7},\ 0$
5.	$0,\ -7 \times 10^{-6},\ -4 \times 10^{-7}$
7.	$-5 \times 10^{-7},\ 0$
8.	$0,\ -2 \times 10^{-6},\ -2 \times 10^{-6},\ 4 \times 10^{-5},\ -2 \times 10^{-6}$

Table 5.2: List of coefficient relative errors for multi-term equations in Table 5.1. Each equation is referenced by its row number in Table 5.1 and errors are listed in the same order as the terms of the equation. The errors reported as 0 correspond to the largest coefficients of each equation, which are normalized to exactly 1.

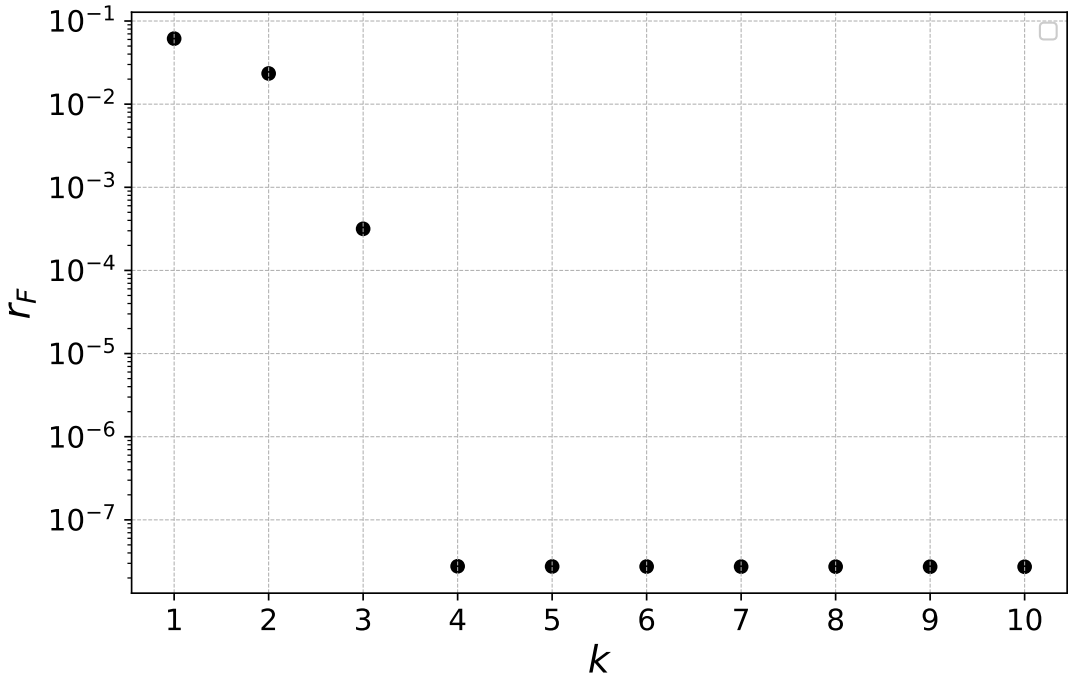
SPIDER's performance for these hyperparameter choices is remarkable: it identifies all of the true equations with no false positives, no spurious terms, and tiny errors in the coefficients – even the very small viscosity $\nu = 10^{-4}$ is identified with a relative error of less than 10^{-6}! (`PySPIDER` also solves this problem with high computational efficiency: the entire process requires less than a minute on a 2024 MacBook Air M3.) This is true even when the terms cannot fully be integrated by parts during

evaluation. Since the noise level of the data is extremely low, we used 6th-order finite differences to evaluate the first-order derivatives that cannot be eliminated. The residuals and relative coefficient errors show that the resulting discretization error is on the order of only 10^{-7} or 10^{-6}, which is comparable to the machine precision for this dataset. Equally interesting is the set of equations chosen as fundamental (rather than implied) by SPIDER: they are either independently physically meaningful, or they illustrate special identities that hold for the rank-2 irreducible representations of the orthogonal symmetry group O(2), which are only one- or two-dimensional.

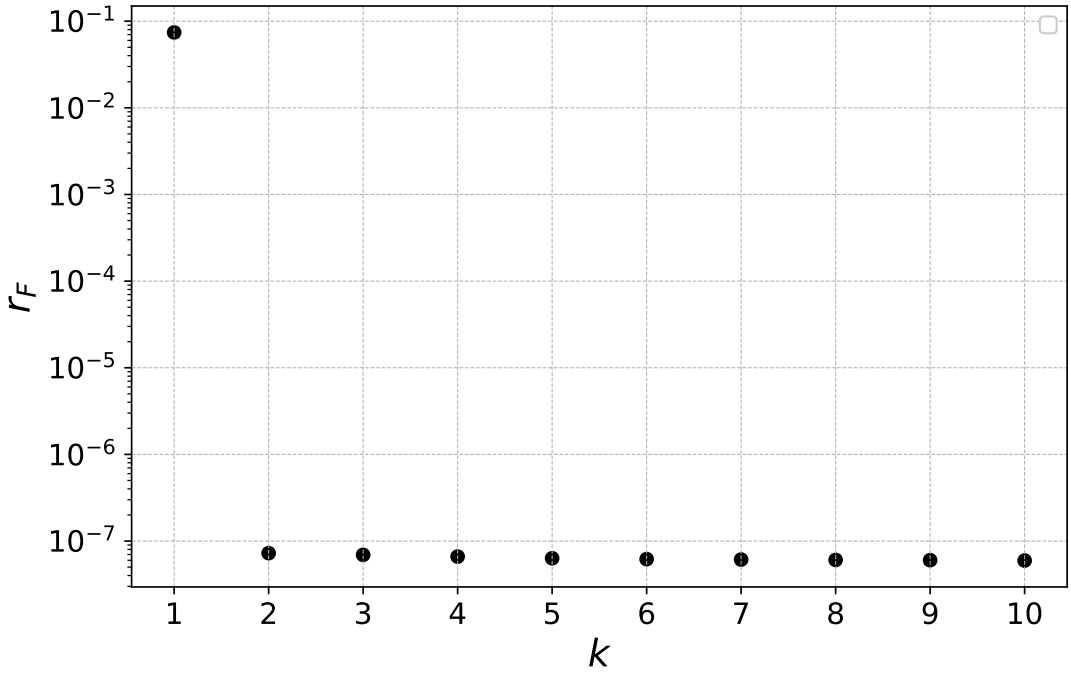
The first equations to be discovered are the incompressibility condition and the Navier-Stokes equation. Next is the pressure-Poisson equation, which is a linear combination of the contracted gradient of the Navier-Stokes equation and the implications of incompressibility $\partial_t \partial_\alpha u_\alpha = 0$ and $\partial_\alpha \partial_\beta^2 u_\alpha = 0$. Because this is a fairly complicated linear combination of implied equations, SPIDER does not recognize that it is also an implied equation. However, for the same reason, the pressure-Poisson equation is not immediately obvious as a direct consequence of incompressibility and Navier-Stokes, and it is independently interesting in its own right. Figure 5.2 shows the residual r as a function of the sparsity k for the Navier-Stokes and pressure-Poisson equations, illustrating that the proper cutoff corresponds to a very abrupt corner in the plot. (In fact, including additional terms up to $k = 10$ decreases the residual by less than 2%.) This is also true for the remaining fundamental equations, for which the corresponding plots are not shown.

The 4th equation identified by SPIDER illustrates an identity specific to the rank-2 antisymmetric irreducible representation in two spatial dimensions: component by component, there is an equality between the antisymmetric projections

$$\mathrm{AS}[\partial_\alpha u_\gamma \cdot \partial_\gamma u_\beta] = \mathrm{AS}[\partial_\alpha u_\beta \cdot \partial_\gamma u_\gamma], \tag{5.1}$$



(a)



(b)

Figure 5.2: Frobenius residual r_F as a function of sparsity k for the best multi-term models identified by SPIDER for the JAX-CFD dataset, corresponding to (a) the Navier-Stokes equation and (b) the pressure-Poisson equation. The residual decreases sharply as terms are added until the correct sparsities (4 and 2, respectively) are reached.

where the right-hand side is zero as an implication of incompressibility. (Such dimension-dependent identities commonly arise in general relativity computations [74].) Note that this identity could have been identified directly, but it features a term that is already known to vanish identically because of incompressibility. As a result, SPIDER outputs the identity only in indirect form. As we are about to see, this phenomenon occurs with the remaining equations as well.

The 5th equation is also correct, combining a more complicated chain of implications with another identity for the rank-2 AS representation in 2D. It is equivalent to the antisymmetrized gradient of the Navier-Stokes equation (i.e., the vorticity equation describing the time evolution of $\omega = \nabla \times \boldsymbol{u}$), after combining the simplification

$$\mathrm{AS}[\partial_\alpha(u_\gamma \partial_\gamma u_\beta)] = \mathrm{AS}[u_\gamma \cdot \partial_\alpha \partial_\gamma u_\beta] + \mathrm{AS}[\partial_\alpha u_\gamma \cdot \partial_\gamma u_\beta] = \mathrm{AS}[u_\gamma \cdot \partial_\alpha \partial_\gamma u_\beta], \tag{5.2}$$

where the last equality follows from the 4th equation identified by SPIDER, and the 2D identity

$$\mathrm{AS}[u_\gamma \cdot \partial_\alpha \partial_\gamma u_\beta] = \mathrm{AS}[u_\alpha \cdot \partial_\gamma^2 u_\beta] \tag{5.3}$$

If the rewrite (5.3) is not used, the 5th equation takes exactly the standard form of the vorticity equation for an incompressible flow upon the substitution $\mathrm{AS}[\partial_\alpha u_\beta] = \omega$:

$$\partial_t \omega + (\boldsymbol{u} \cdot \nabla)\omega - \nu \nabla^2 \omega = 0 \tag{5.4}$$

The 6th equation identified by SPIDER is analogous to the 4th except in a different irreducible representation (symmetric trace-free rather than antisymmetric). The 7th equation is also implied by incompressibility and a 2D rank-2 symmetric trace-free identity:

$$\mathrm{STF}[\partial_\alpha u_\gamma \cdot \partial_\beta u_\gamma + \partial_\gamma u_\alpha \cdot \partial_\gamma u_\beta] = 2 \cdot \mathrm{STF}[\partial_\alpha u_\beta \cdot \partial_\gamma u_\gamma] \tag{5.5}$$

Finally, the 8th equation follows by taking the STF projection of the gradient of Navier-Stokes and then applying the 2D rank-2 STF identity

$$\mathrm{STF}[u_\alpha \cdot \partial_\gamma^2 u_\beta + u_\gamma \cdot \partial_\alpha \partial_\beta u_\gamma] = \mathrm{STF}[u_\gamma \cdot \partial_\alpha \partial_\gamma u_\beta] + \mathrm{STF}[u_\alpha \cdot \partial_\beta \partial_\gamma u_\gamma], \tag{5.6}$$

where the last term vanishes due to an implication of incompressibility.

Although all 8 equations above found by SPIDER hold exactly given the ground truth (incompressibility plus Navier-Stokes), relations with higher residuals are only approximately correct. For example, the next-best relation achieves a residual $r = 5.5 \times 10^{-6}$:

$$\begin{aligned}\mathrm{STF}[\partial_\alpha p \cdot u_\beta + 10^{-4} \cdot \partial_\alpha \partial_\beta p - u_\alpha \cdot u_\gamma \cdot \partial_\beta u_\gamma + u_\alpha \cdot \partial_t u_\beta + \\ + u_\gamma \cdot u_\gamma \cdot \partial_\alpha u_\beta + 10^{-4} \cdot u_\gamma \cdot \partial_\alpha \partial_\beta u_\gamma + 10^{-4} \cdot \partial_t \partial_\alpha u_\beta] = 0\end{aligned} \tag{5.7}$$

This relation equals the symmetric trace-free projection of the Navier-Stokes equation times $\boldsymbol{u}$, after replacing the resulting term $u_\alpha \cdot \partial_\gamma^2 u_\beta$ (with an additional factor of $-\nu$) using the 8th equation identified by SPIDER and then discarding the term $\nu^2 \partial_\alpha \partial_\gamma^2 u_\beta$ as negligible. This approximation is physically justified for flows with such a low viscosity as considered in our example.

Now that we have illustrated SPIDER's ability to discover exactly the correct equations with high accuracy with good hyperparameter choices, let us briefly analyze the results of a poor choice. When using combinatorial initialization coupled with greedy iteration, SPIDER generally no longer identifies all of the equations. In fact, for $k_s < 5$ and $r_{max} = 3 \times 10^{-5}$, no number of backward-forward iterations is sufficient for SPIDER to recover either the 8th equation in Table 5.1 or equation (5.7), which should both be accepted for this value of the residual threshold. Only at $k_s = 5$ is the 8th equation finally found, but this is already equivalent to exhaustive combinatorial search for every single equation in Table 5.1! This is an indication that, in contrast

to its competitive performance for synthetic problems in Section 4.5, in a practical setting, combinatorial initialization is unable to identify the right combination of terms reliably for k_s less than the true sparsity of the relation. Instead, either full support initialization or TII should be used to select the initial guess for the support.

5.2 Discrete: Compressible Fluid Flow

Having validated SPIDER's ability to learn complete models of continuous systems, we turn to the more challenging case of many-body systems (MBS). We simulated a dense compressible fluid consisting of $N = 10000$ particles with unit mass, interacting under simple Gaussian pair potentials

$$U(\boldsymbol{x}) = Ae^{-\|\boldsymbol{x}\|^2/(2\sigma^2)} \tag{5.8}$$

The simulation domain is a 1×1 (all quantities are in dimensionless units) 2D square with periodic boundary conditions. We use a sixth-order symplectic integrator [128] with timestep $\Delta t = 0.00125$ to compute the trajectories of the particles on a time interval of length $T_{max} = 0.64$. The "width" of the potential σ and mean interparticle spacing d were taken to be equal: $\sigma = d = 1/\sqrt{N} = 0.01$; this choice ensures that each particle is affected by non-negligible forces from multiple neighbors at almost all times. The initial positions of the particles are selected randomly; on the other hand, their velocities are a deterministic periodic function of position, chosen to lead to long-lasting vortices as well as moderate-intensity density waves, with a ratio of kinetic to potential energy $K/V \approx 0.187$ and a root-mean-squared velocity $m_v \approx 0.69$. The conditions of the simulation correspond to a highly compressed gas. Figure B.1 in Appendix B shows 4 evenly spaced snapshots of the data, with the small colored arrows indicating the positions and directions of the velocity for each particle. Over

the course of the simulation, mixing of the fluid slowly leads to momentum diffusion and increased local disorder, corresponding to a trend towards thermal equilibrium.

The data available to SPIDER consist of the positions and velocities of each particle, so the libraries are constructed by applying the coarse-graining operator ρ to tensor functions of a single vector-valued observable $\boldsymbol{v}$. We perform regression on the resulting scalar, vector, rank-2 antisymmetric, and rank-2 STF hydrodynamic libraries with complexity threshold 5 and terms containing at most 2 primes; these contain 41, 50, 14, and 33 terms, respectively.

To coarse-grain the discrete data, we use a quartic polynomial kernel given by (2.33) with $p = 4$, evaluated on a 256×256 spatial grid. (An even finer 512×512 grid was not found to yield qualitatively different results.) The first major question is how to pick the kernel width h. To assist with this task, we compute the coarse-grained density field ρ and the components of the coarse-grained momentum, $\rho[v_x]$ and $\rho[v_y]$, for a specific snapshot using a few candidate values of h and qualitatively assess the smoothness of the coarse-grained data by eye. Figure 5.3 shows the results for ρ and $\rho[v_x]$ at time $t = 0.125$. (Note that the coarse-grained fields approach zero near the edges of the domain, since the coarse-graining kernel does not wrap around the periodic boundaries.) We observe that the coarse-grained fields are too granular for $h = 0.01$ [panels (a) and (b)], with the effect of the individual discrete particles clearly visible. This is unsurprising, as $h = d$, which should lead to blurring rather than averaging. Conversely, for $h = 0.05$ [panels (e) and (f)], very little of the fine-scale structure can still be seen: the coarse-graining clearly oversmooths the data. Finally, an intermediate value of $h = 0.02$ [panels (c) and (d)] strikes a good balance between ensuring the coarse-grained fields are sufficiently smooth and preserving detail of the density waves. This choice of kernel width averages over roughly $\pi(h/d)^2 \sim 13$ particles to obtain the coarse-grained fields. For more disordered dynamics where correlations between the velocities of neighboring particles are much smaller (e.g.,

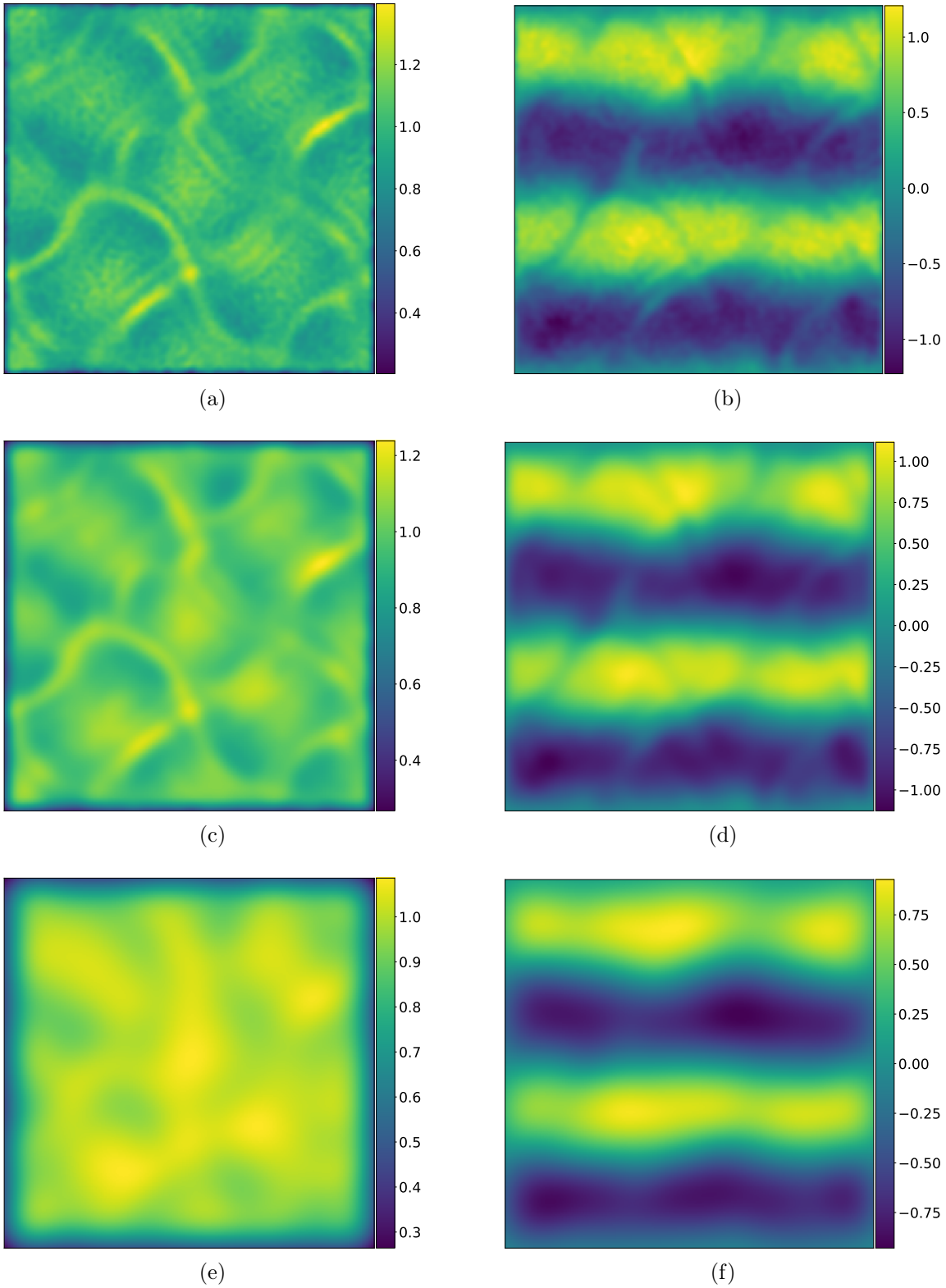


Figure 5.3: Snapshots of coarse-grained density ρ (left column) and x component of momentum $\rho[v_x]$ (right column) at time $t = 0.125$ for $h =$ (a)-(b) 0.01, (c)-(d) 0.02, (e)-(f) 0.05.

a gas instead of a liquid), a significantly wider kernel averaging over much larger numbers of particles would typically be necessary, although this may result in a loss of information at small length scales.

Another non-trivial task is determining appropriate length and time scales for the purpose of normalizing the library matrix (see Section 4.2). We would like the norms of most columns of the nondimensionalized library matrix G (i.e., after rescaling by the corresponding term's scale $S[y]$ as defined in (4.8)–(4.10)) to be roughly comparable. The most straightforward choice using the correlation length and time of the coarse-grained velocity, $\ell \sim \tau \sim 0.25$, does not achieve this goal: the norms of the nondimensionalized columns vary by 2–3 orders of magnitude. Instead, a better choice corresponds to the length and time scales of the much narrower density waves: $L \sim T \sim 0.02$. In that case, the nondimensionalized columns have scales mostly spanning only a single order of magnitude. We also rescale the coarse-graining operator ρ by a constant factor of N^{-1} in order to make the coarse-grained density field $O(1)$.

Because coarse-graining is the most computationally expensive step in the current implementation of `PySPIDER`, the runtime of `PySPIDER` for discrete systems is generally roughly proportional to the product of the total volume of the integration subdomains Ω_i and the number of distinct indexed primes used in the library. However, we may improve the runtime of the algorithm by using multiple (preferably, orthogonal) weight functions supported on the same Ω_i. Here we use 8 different modulated weight functions $w_i' = w_i g_i$ as defined in (3.8). The polynomial envelopes w_i are given by (3.6)–(3.7) with $q = 8$, while each g_i is equal to the product of one-dimensional Legendre polynomials of zeroth or first order (i.e., $g_i = 1, x, y, t, xy, xt, yt, xyt$). This allows us to sample only a small number of integration domains $N_d = 30$ and still obtain an overdetermined problem for all of the libraries: for instance, $n_w = 240 \gg 41$ for the scalar library. The integration domains are randomly sampled $0.1 \times 0.1 \times 0.05$ cubes

(in nondimensional units) sufficiently far from the boundary of the domain, so that the results of the coarse-graining are not influenced by the boundary.

As in the previous section, we use the Frobenius relative residual r_F and the residual jump criterion with $\gamma = 1.5$ for selecting the sparsity of each relation. We compare the performance of full support initialization with a single pass of greedy backward elimination, TII initialization with $k_m = 10$ and up to 10 backward-forward passes ($1 \leq k \leq 10$), and combinatorial search initialization with $k_s = 3$ and 10 backward-forward passes. The residual threshold r_{max} must be set much higher in this case than the example we considered in Section 5.1. With the exception of the conservation of mass equation $\partial_t \rho + \partial_\alpha \rho[v_\alpha] = 0$, which holds to a high degree of accuracy for any discrete data of reasonable quality, none of the discovered relations achieve a residual lower than $r = 9 \times 10^{-4}$. This is a consequence of the noise fundamentally present in this type of problem: with a total of only 10^4 particles, the unmodeled degrees of freedom of each individual particle have a significant contribution to the overall coarse-grained fields. We use $r_{max} = 2 \times 10^{-3}$, which is sufficient to learn several physically meaningful equations as well as some relations that do not have any deep physical interpretation. We also cross-validated the learned equations by using a train-test split of 80% to 20%, but there was no substantial gap between the training and test errors for any of the equations. However, changing the random sampling of integration domains leads to coefficient fluctuations on the order of 10%.

The same set of fundamental equations is learned when using full support or TII initialization with any number of backward-forward passes; these are listed in Table 5.3. An additional 62 implied equations were also output by SPIDER and are not shown. (The set of implied equations includes not only equations that can be expressed using terms in the libraries, each of which includes at most 2 primes, but rather all equations up to complexity 5 with no limit on the number of primes per term.) As in Section 5.1, we find that combinatorial search initialization performs

eqn. ID	sublibrary	equation	r_F	r_h	type
(a)	rank 0, $\mathcal{C} = 3$	$\partial_t \rho + \partial_\alpha \rho[v_\alpha] = 0$	6.4×10^{-7}	1.4×10^{-6}	K
(b)	rank 0, $\mathcal{C} = 4$	$-0.996 \cdot \rho \cdot \partial_\alpha^2 \rho + 0.268 \cdot \rho \cdot \partial_t^2 \rho + \partial_\alpha^2 \rho - 0.27 \cdot \partial_t^2 \rho = 0$	1.6×10^{-3}	5.9×10^{-3}	K+C
(c)	rank 1, $\mathcal{C} = 4$	$-0.997 \cdot \rho \cdot \partial_\alpha \rho - 0.191 \cdot \rho \cdot \partial_t \rho[v_\alpha] + \partial_\alpha \rho + 0.192 \cdot \partial_t \rho[v_\alpha] = 0$	7.7×10^{-3}	5.5×10^{-3}	K+C
(d)	rank 0, $\mathcal{C} = 5$	$0.714 \cdot \rho - 0.709 \cdot \rho \cdot \rho + \rho \cdot \rho[v_\alpha \cdot v_\alpha] - 0.992 \cdot \rho[v_\alpha \cdot v_\alpha] = 0$	1.2×10^{-3}	0.015	C
(e)	rank 0, $\mathcal{C} = 5$	$\partial_\alpha^2 \rho + 3.46 \times 10^{-5} \cdot \partial_\alpha^2 \partial_\beta^2 \rho - 0.167 \cdot \partial_t^2 \rho + 0.162 \cdot \partial_\alpha \partial_\beta \rho[v_\alpha \cdot v_\beta] = 0$	1.6×10^{-3}	0.013	K
(f)	rank 1, $\mathcal{C} = 5$	$-\rho \cdot \partial_\alpha \rho + \partial_\alpha \rho = 0$	1.0×10^{-3}	0.062	C
(g)	rank 1, $\mathcal{C} = 5$	$\rho \cdot \rho[v_\alpha] - 0.727 \cdot \rho \cdot \rho[v_\alpha \cdot v_\beta \cdot v_\beta] - 0.997 \cdot \rho[v_\alpha] + 0.718 \cdot \rho[v_\alpha \cdot v_\beta \cdot v_\beta] = 0$	9.4×10^{-4}	7.8×10^{-3}	C
(h)	rank 1, $\mathcal{C} = 5$	$\partial_\alpha \rho + 4.33 \times 10^{-5} \cdot \partial_\alpha \partial_\beta^2 \rho + 0.158 \cdot \partial_t \rho[v_\alpha] + 0.16 \cdot \partial_\beta \rho[v_\alpha \cdot v_\beta] = 0$	1.4×10^{-3}	0.011	K
(i)	rank 2 STF, $\mathcal{C} = 5$	$\mathrm{STF}\big[-0.5 \cdot \partial_\gamma^2 \rho[v_\alpha \cdot v_\beta] + \partial_\alpha \partial_\gamma \rho[v_\beta \cdot v_\gamma] - 0.5 \cdot \partial_\alpha \partial_\beta \rho[v_\gamma \cdot v_\gamma]\big] = 0$	4.0×10^{-17}	8.5×10^{-17}	G

Table 5.3: List of all fundamental equations identified by SPIDER with full support initialization for the discrete compressible fluid simulation. In addition to the equation itself, we list the irreducible representation and complexity $\mathcal{C}$ of the sublibrary in which it is first identified, its residuals r_F and r_h, and its interpretation: whether it is a known physical equation (K), a general identity (G), an approximate implication of $\rho = 1$ (C), or a combination of multiple types of equation.

poorly: regardless of the number of backward-forward iterations, it is unable to infer equations (d) and (e) in Table 5.3, which are found by the other initializations.

Note that some equations in Table 5.3 do not initially appear in the smallest sublibraries (corresponding to lowest term complexity) that contain them but are instead discovered at higher complexities. This is because r_F depends on the size of the sublibrary being considered and decreases with complexity. Hence, a relation that may be above the threshold r_{max} at low $\mathcal{C}$ can still be discovered at higher $\mathcal{C}$. (In effect, considering high-complexity libraries slightly broadens the search among low-complexity relations as well.) In contrast, the hybrid residual r_h defined in (4.6) does not directly scale with complexity, but it is a worse choice for use during regression since it is not monotonic with the number of retained terms k. Instead, r_h is useful as a library-independent measure of the quality of inferred relations.

The residual r_F as a function of the number of terms retained in each equation is shown in Figure 5.4. Many of the identified equations are closely related to the approximate relation $\rho = 1$, which is true only in an averaged sense. A smaller and more physically meaningful subset of fundamental equations is obtained if these are secondarily filtered by r_h, discarding all multi-term equations with $r_h \geq 0.015$; in fact, equations (d) and (f) are direct (approximate) implications of $\rho = 1$.

Let us discuss the interpretations of the fundamental equations learned by SPIDER. Equation (a) is the conservation of mass equation:

$$\partial_t \rho + \partial_\alpha \rho[v_\alpha] = 0 \tag{5.9}$$

The coefficients of the two terms in this equation are identified with excellent precision, deviating from each other by only 5×10^{-8}. Equation (b) can be approximately factored as

$$(1 - \rho)\left(a^2 \partial_\alpha^2 \rho - \partial_t^2 \rho\right) = 0, \tag{5.10}$$

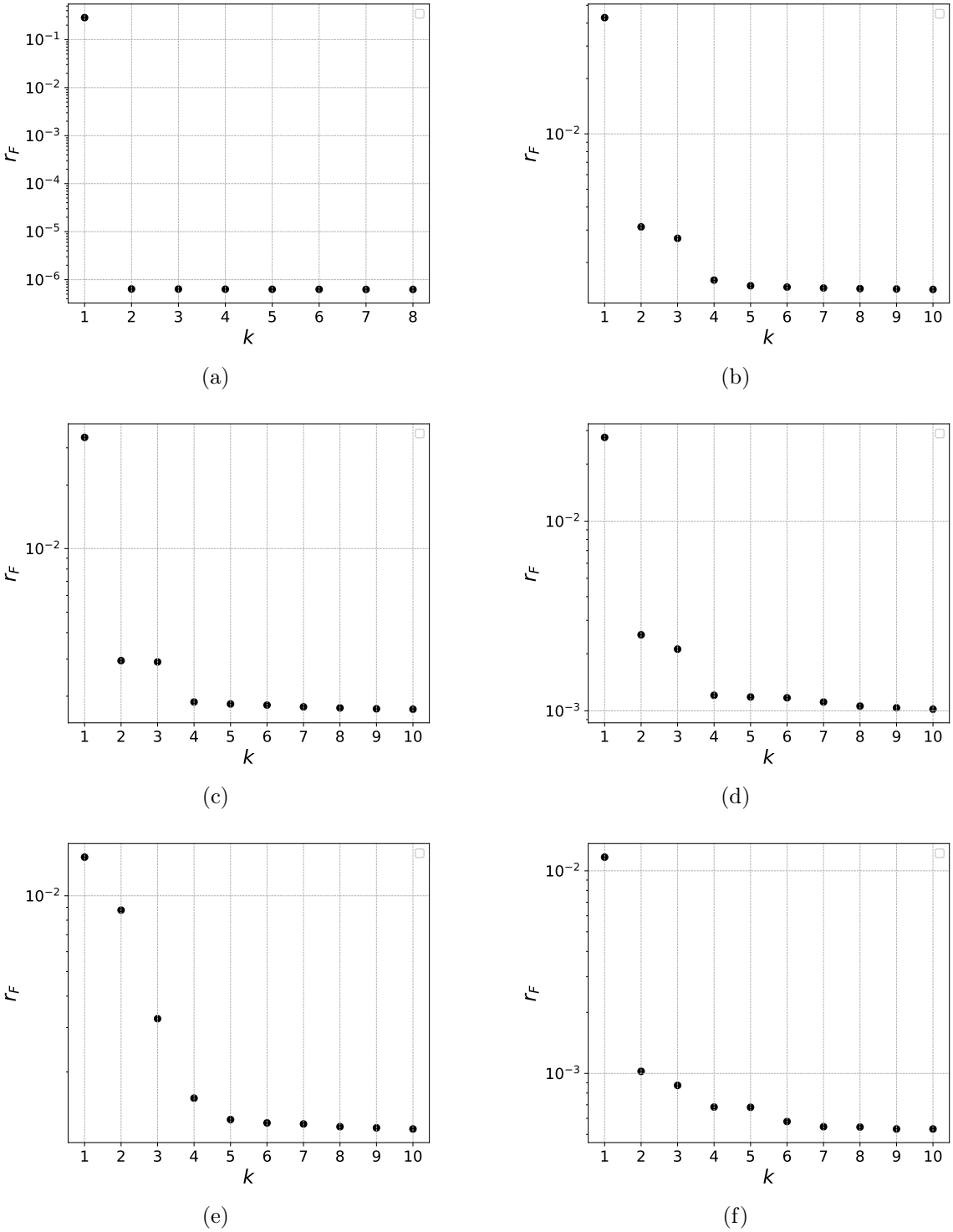
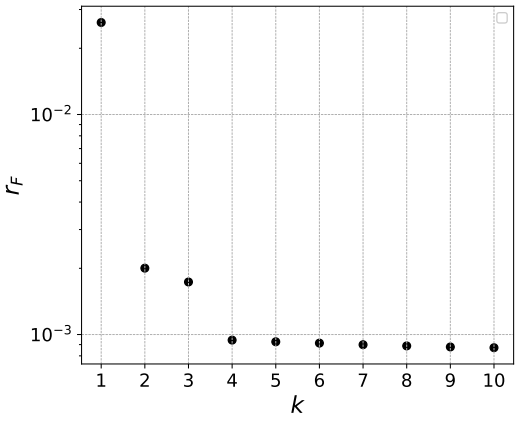
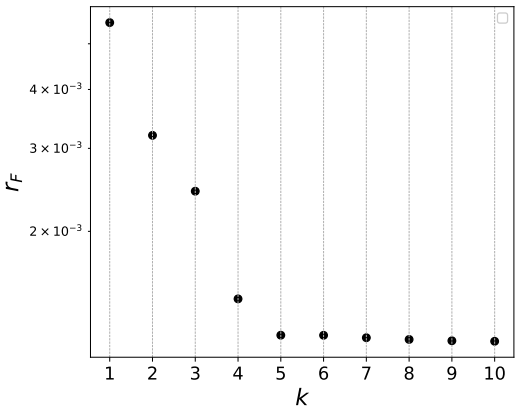


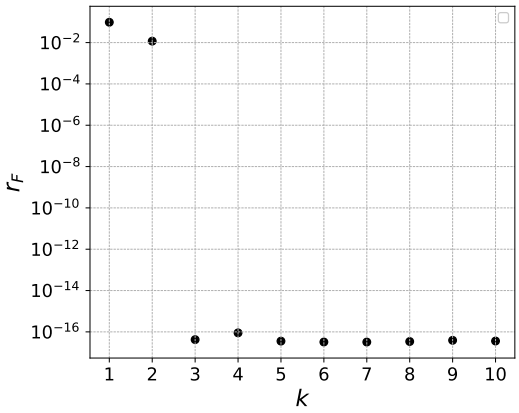
Figure 5.4: Frobenius residual r_F as a function of sparsity k for the fundamental equations identified by SPIDER for the discrete compressible fluid dataset. The panels (a)-(i) correspond to the labels in Table 5.3. The relationship shown in panel (i) is non-monotonic due to limited machine precision.



(g)



(h)



(i)

Figure 5.4: Frobenius residual r_F as a function of sparsity k for the fundamental equations identified by SPIDER for the discrete compressible fluid dataset. The panel labels (a)-(i) correspond to the equation IDs in Table 5.3. The relationship shown in panel (i) is non-monotonic due to limited machine precision.

i.e., the product of the wave equation and $1-\rho$. Such factored equations are a common discovery of SPIDER when working with MBS; in effect, they allow SPIDER to decrease the residual artificially by a factor roughly corresponding to the mean of $|1-\rho|$. The hybrid residual of the wave equation without the factor $1-\rho$ is much higher ($r_h = 0.11$), so equation (b) should be considered spurious. For all relations that can be factored in this way, we list their hybrid residuals after removing the factor of $1-\rho$ in Table 5.4. Indeed, this extra factor deflates the residual by roughly 20-fold in all cases.

eqn. ID	factored form	r_h	r_h/(unfactored r_h)
(b)	$\partial_\alpha^2\rho - 0.246 \cdot \partial_t^2\rho = 0$	0.15	25
(c)	$\partial_\alpha\rho + 0.154 \cdot \partial_t\rho[v_\alpha] = 0$	0.11	20
(d)	$-0.956 \cdot \rho + \rho[v_\alpha \cdot v_\alpha] = 0$	0.26	17
(f)	$\partial_\alpha\rho = 0$	1	16
(g)	$\rho[v_\alpha] - 0.602\rho[v_\alpha \cdot v_\beta \cdot v_\beta] = 0$	0.14	17

Table 5.4: Hybrid residuals r_h of factorized relations after eliminating the factor of $1-\rho$ and refitting coefficients. We also report the ratio between the values of r_h without and with the factor.

There are several other factored relations, all of which are spurious. Equation (c) is closely related to (b). Indeed, its factored form

$$\tilde{a}^2\partial_\alpha\rho + \partial_t\rho[v_\alpha] = 0 \tag{5.11}$$

can be differentiated with respect to α to obtain

$$\tilde{a}^2\partial_\alpha^2\rho + \partial_t\partial_\alpha\rho[v_\alpha] = \tilde{a}^2\partial_\alpha^2\rho - \partial_t^2\rho = 0 \tag{5.12}$$

by an application of (5.9), although the coefficients differ somewhat from (5.10). Again, the hybrid residual of (5.11) is high. Similarly, equation (d) can be approximately factored as

$$(1-\rho)(\bar{v}^2\rho - \rho[v_\alpha v_\alpha]) = 0, \tag{5.13}$$

so it is related to the approximation of $|\boldsymbol{v}|^2$ using a characteristic velocity:

$$\rho[v_\alpha v_\alpha] \approx \bar{v}^2 \rho. \tag{5.14}$$

Equation (g) can be interpreted similarly, except it includes an additional factor of $\boldsymbol{v}$. The remaining approximate implication of $\rho = 1$, namely equation (f), is even less trustworthy, as it is itself a more sparse and less accurate version of (c).

On the other hand, equation (h) is actually physically meaningful. It takes the standard form of a momentum equation

$$\partial_t \rho[v_\alpha] + c_1 \partial_\beta \rho[v_\alpha v_\beta] - \partial_\beta \tau_{\alpha\beta} = 0. \tag{5.15}$$

where $c_1 \approx 1$, as required by Galilean invariance, and

$$\tau_{\alpha\beta} = -\left(c_2 \rho + c_3 \partial_\gamma^2 \rho\right) \delta_{\alpha\beta} \tag{5.16}$$

is the Cauchy stress tensor. We may rewrite (5.16) in a more familiar form using the approximate relation (5.14):

$$\tau_{\alpha\beta} \approx -\left(p_N + \mu \partial_\epsilon^2 \rho[v_\gamma v_\gamma]\right) \delta_{\alpha\beta}, \tag{5.17}$$

where we have introduced the pressure

$$p_N = \frac{c_2}{\bar{v}^2} \rho[v_\alpha v_\alpha] = 8.9 \cdot \rho\left[|\boldsymbol{v}|^2\right] \tag{5.18}$$

and

$$\mu = \frac{c_3}{\bar{v}^2} \approx 4 \times 10^{-4} \tag{5.19}$$

is a non-Newtonian analogue of the bulk viscosity. However, we remark that it is the velocity-independent stress (5.16) rather than (5.17) that yields the more accurate general description of the dynamics in this system. This reflects the fact that the pressure is dominated by the effect of the potential rather than kinetic energy, running counter to the usual intuition derived from kinetic theory.

A natural question is whether retaining more terms in equation (h) might allow us to learn velocity-dependent corrections to the pressure or viscosity. We find that not to be the case: the 5-term version of the equation, which achieves only a slightly smaller residual, adds a term $c_4\rho\cdot\partial_\beta\rho[v_\alpha v_\beta]$; this term violates the Galilean symmetry of the system and indicates overfitting. Keeping additional terms has a negligible influence on the residual r_F, as shown in Figure 5.4(h). (This could cease to be true if higher-complexity libraries are considered.)

Next, equation (e) is directly implied by (h): indeed, with slight adjustments in the coefficients, (e) is the trace of the gradient of the (h). However, because (e) is discovered earlier than (h), it is not excluded by implication rules in this particular example. Equation (e) also has a standard physical interpretation: it is closely related to the foundational equation of aeroacoustics, Lighthill's equation [71]. To see this, we can write (e) as

$$\partial_t^2\rho - c_1'\partial_\alpha\partial_\beta\rho[v_\alpha v_\beta] + \partial_\alpha\partial_\beta\tau'_{\alpha\beta} = 0, \tag{5.20}$$

where τ' is defined by (5.16) but with slightly different coefficients. Grouping terms into the Lighthill stress tensor given by

$$T_{\alpha\beta} = \rho[v_\alpha v_\beta] - \tau'_{\alpha\beta} - a_0^2\rho\delta_{\alpha\beta}, \tag{5.21}$$

and noting that $c_1' \approx 1$, we obtain

$$\partial_t^2\rho - a_0^2\partial_\alpha^2\rho = \partial_\alpha\partial_\beta T_{\alpha\beta}, \tag{5.22}$$

which is indeed Lighthill's equation. However, (5.22) contains an implicit dependence on the Cauchy stress tensor (5.16), which takes a non-standard form here as previously mentioned.

Using τ' instead of τ gives rise to a slightly different viscosity estimate $\mu' \approx 3 \times 10^{-4}$. However, even a 25% uncertainty in the coefficient of this small term is impressive given the limited number of particles in the simulation and the resulting high noise in the coarse-grained fields. Similarly to (h), Figure 5.4(e) shows that adding further terms does not meaningfully decrease the residual of equation (e) either.

Finally, equation (i) is a data-independent identity for the rank-2 symmetric trace-free irreducible representation in 2D, which is directly analogous to (5.5) in the continuous library.

Although no second-rank equations with a physical interpretation were discovered, the scalar equations (a) and (e) and vector equation (h) are together sufficient to form a complete, highly accurate, and physically interpretable mean-field model of our discrete system via integro-differential equations. Moreover, these equations are mutually consistent up to small coefficient errors that reflect the challenge of applying a continuum approximation to a system of only 10^4 particles. SPIDER would likely be able to learn mean-field models achieving a lower error given a version of this system with a higher particle number. However, the fact that SPIDER is able to obtain a quantitatively accurate continuous description for a discrete system consisting of so few particles is remarkable in its own right.

Chapter 6

Conclusion

This dissertation has presented a systematic and almost fully automatic implementation of the SPIDER framework for model discovery via sparse regression. The symbolic reasoning algorithms we have described eliminate the necessity of manually specifying the rules for representing and evaluating each individual term used for sparse regression. This allows the user to focus entirely on tuning a small number of hyperparameters to incorporate physical considerations such as the characteristic dimensional scales of the data, rather than the laborious and error-prone construction of the term libraries. Moreover, the identification of logical implication between different relations allows SPIDER to identify all equations describing the data immediately, instead of requiring the user to manually prune the library and rerun the regression in order to learn one new equation at a time. These advances remove a critical barrier to adoption of SPIDER by the wider scientific community and enable the use of much larger and more expressive libraries.

We have also shown empirically that SPIDER is capable of discovering comprehensive physical models for both systems featuring continuous data and many-body systems (MBS) composed of discrete interacting units. In particular, the ability to learn mean-field PDE models for MBS addresses a key problem in multiscale modeling

and represents a major conceptual generalization of SPIDER compared to previously published work. The machine precision-level accuracy and robust identification of the equations for the continuous fluid flow example of Section 5.1 exemplifies SPIDER's strength when the algorithm is provided with the correct set of observables and sufficient amounts of data to overcome the inherent noise of the system. In contrast, the description of the discrete compressible fluid in Section 5.2 is approximate, as is expected of a mean field model. This reflects the large contributions of the unmodeled degrees of freedom of each individual particle to the overall coarse-grained fields, which effectively manifest as noise that cannot be modeled by mean field theory.

6.1 Future Work

Even though SPIDER is now positioned as a powerful and easy to use data-driven tool for model discovery, there still remain promising avenues for further improvement. First, there are several conceptual generalizations of SPIDER that we aim to pursue in the near future. As discussed briefly in Section 2.2, the symbolic algorithms in this dissertation can be extended from the simple case of systems with an $\mathrm{E}(n)$ global symmetry and a Euclidean spacetime metric to other symmetry groups and metrics, as well as tensors of rank higher than 2. In some cases, this may lead to more complex tensor symmetries and identities than have been considered in this work. These could be automatically addressed through symbolic computation using Cadabra [89, 90, 91], a computer algebra system specialized to tensor calculus that is able to perform much more complex canonicalization operations than the current version of `PySPIDER`. For instance, Cadabra can automatically simplify certain multi-term identities involving three or more terms, such as the Ricci and Bianchi identities in general relativity, as well as Lovelock-type identities [74] such as those discovered by SPIDER in Chapter 5.

While SPIDER has proven to be a powerful tool for model discovery, it is able to output simple, high-fidelity models only when the chosen set of observables and library definition admits a sparse representation of the dynamics in terms of PDEs. (For instance, the equations of fluid dynamics take the form of integro-differential equations rather than PDEs if formulated in terms of the velocity alone.) In scientific fields with well-established physics-based models, existing domain knowledge often provides a natural choice of observables (e.g., number density and momentum in the case of MBS). For more complex or less well understood problems, the best choice of features (model building blocks) is usually unknown. For example, it is unclear what is a proper set of variables for capturing the small scales in multiscale models such as large eddy simulation (LES) of fluid turbulence. Feature learning can enable SPIDER to handle a wider range of challenging problems in fields such as biology, where the data are often partially observed and represent highly nonlinear processes. Similarly, while coarse-graining currently uses a predefined kernel (such as a Gaussian or a specific compactly-supported polynomial) with a physically informed length scale, it may be better to learn optimal mean-field kernels from a parametrized family. Some problems may benefit from using variable bandwidth kernels [7] that depend on the local data.

At present, there is no comprehensive method for quantifying the uncertainty of the equations learned by SPIDER. There has been some recent work on incorporating Bayesian inference into sparse regression [47]. Besides allowing for uncertainty quantification in model discovery, the framework of information theory could be combined with feature learning in order to identify a methodical balance between flexibility and parsimony in the design of the features themselves.

Finally, not all of the conceptual machinery of SPIDER has been incorporated into the `PySPIDER` codebase at the time of writing. Unimplemented features include

automated learning of boundary conditions (Section 2.3) and regression using mixed-integer optimization (Section 4.4).

Additionally, program optimization deserves further attention, as future applications of SPIDER may need to construct much larger libraries with $O(1000)$ terms or more. For instance, these large libraries may arise when the models are built from a greater number of different observables that could all be physically relevant, rather than only 1 or 2 observables that are sufficient to describe the simple systems considered here. As an example, modeling magnetohydrodynamic systems may require simultaneously considering the electric and magnetic fields, electric charge density, and current in addition to pressure and velocity. Although library generation is computationally inexpensive, the evaluation process requires computing a number of multidimensional integrals equal to the number of entries in the library matrix G. Moreover, in the case of PDE models for MBS, the speed of evaluation is currently limited by an expensive coarse-graining step which scales with the total number of particles, the library size (via the number of distinct indexed primes), and the resolution of the coarse-graining grid. The computational cost of some regression methods and initializations also increases rapidly with the size of the library. These challenges may be addressed by optimizing the corresponding routines in `PySPIDER`. Almost all of the tasks involved in library evaluation are embarrassingly parallel and can be accelerated considerably by implementing multi-core processing or even GPU programming. As another example, greedy backward-forward regression in `PySPIDER` could be made more efficient by implementing SPRINT [49].

Appendix A

PySPIDER code overview

In this appendix, we summarize the structure of SPIDER's official Python implementation, `PySPIDER`, at the time of writing. `PySPIDER` was written in collaboration with Akash Gaonkar and Carlos de Oliveira e Silva Filho and is available on GitHub at `https://github.com/sibirica/PySPIDER`. While some parts of the implementation may evolve over time, we aim to keep the general structure consistent for ease of use while maintaining an up-to-date tutorial on the GitHub page. The package is organized into three groups of modules:

`commons` contains modules corresponding to general concepts that are not specific to the continuous or discrete symbolic language. These make up the bulk of the `PySPIDER` codebase and include

- `z3base.py`: definition of index types, irreducible representations, and the language-agnostic satisfiability modulo theory (SMT) problem for enumerating canonical indexings (see Section 3.2)
- `library.py`: general definitions of types (observable, prime, term, equation) and operations on types discussed in Section 3.1, as well as general utilities for library generation (Section 3.2)

- `process_library_terms.py`: definition of `AbstractDataset` template object for storing all attributes associated with a specific SPIDER dataset, `LibraryData` object for storing information associated with a specific irreducible representation, and general routines for library evaluation (Section 3.3)

- `weight.py`: helper functions for constructing and differentiating weight functions

- `identify_models.py`: automatic equation synthesis and implication (Section 3.4) by calling a regression oracle

- `sparse_reg_bf.py`: major routines associated with regression (Sections 4.1–4.3, 4.5)

- `TInvPower.py`: implementation of truncated inverse iteration (TII) and truncated Rayleigh quotient iteration (TRQI) (Section 4.3)

- `sr_utils.py`: regression helper functions for finding least-squares solutions of linear problems with a fixed set of terms

- `utils.py`: miscellaneous helper functions, e.g., filtering terms in a library by part of their string representation

`continuous` contains code specific to the continuous language:

- `library.py`: continuous library generation

- `process_library_terms.py`: definition of continuous `SRDataset` object for storing all attributes associated with a specific continuous SPIDER dataset, evaluation of continuous primes and nondimensionalization

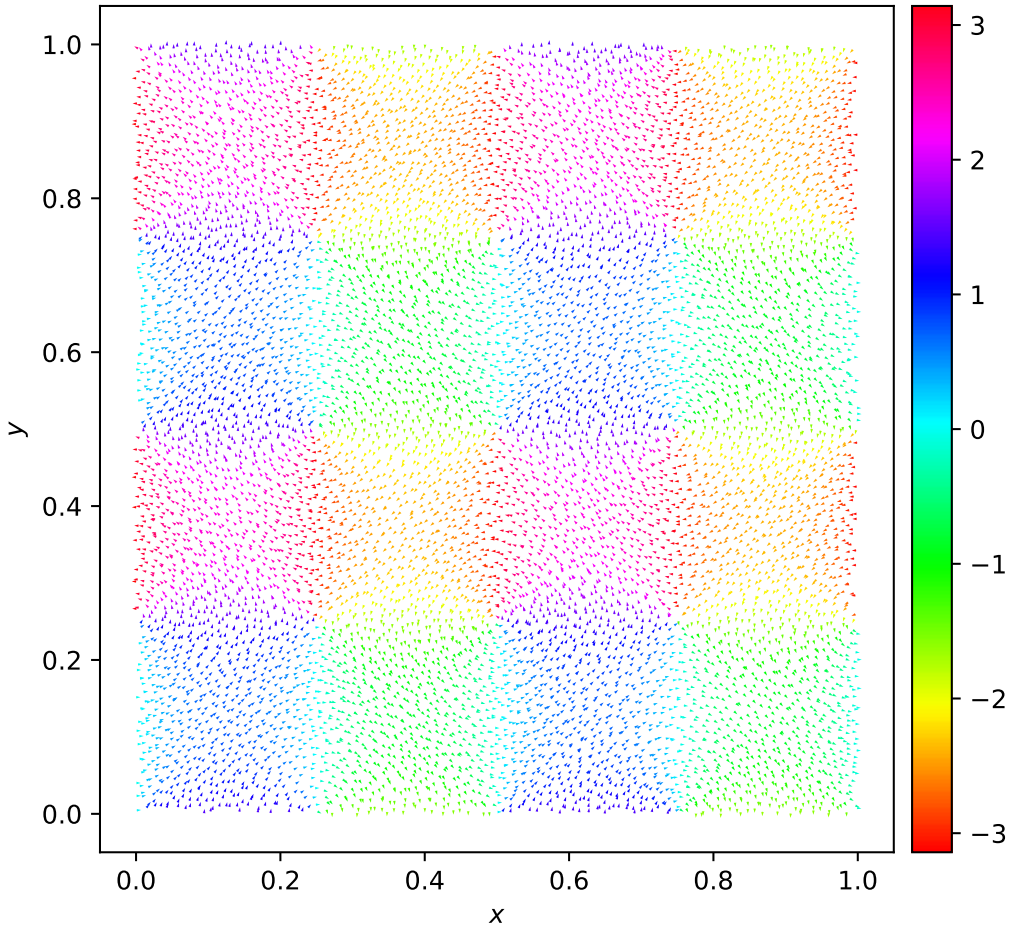
Similarly, `discrete` contains code specific to the discrete language and many-body systems:

- `library.py`: definition of coarse-grained prime (CGP) type, discrete library generation
- `process_library_terms.py`: definition of discrete `SRDataset` object for storing all attributes associated with a specific discrete SPIDER dataset, evaluation of discrete primes and nondimensionalization
- `coarse_grain_utils.py`: routines for coarse-graining corresponding to Gaussian or compactly-supported polynomial kernels

Running SPIDER is a two-step process. First, one should construct an `SRDataset` object summarizing the dataset and the irreducible representations to be analyzed. Functions associated with this object are used to construct in order the libraries, set of integration domains, weight functions, and finally library matrices G, which are all stored within the `SRDataset`. Then, one defines all of the regression hyperparameters associated with each irreducible representation. Together with the `LibraryData` associated with the irreducible representations, these are passed to the function `interleave_identify` defined in `identify_models.py`, which prints and returns all fundamental equations and their associated residuals, along with a summary of the regression process, the implied equations, and the terms excluded from further regression iterations due to logical implication. Each major routine can also be called with a Boolean `verbose` flag, which toggles printing of debugging information during execution.

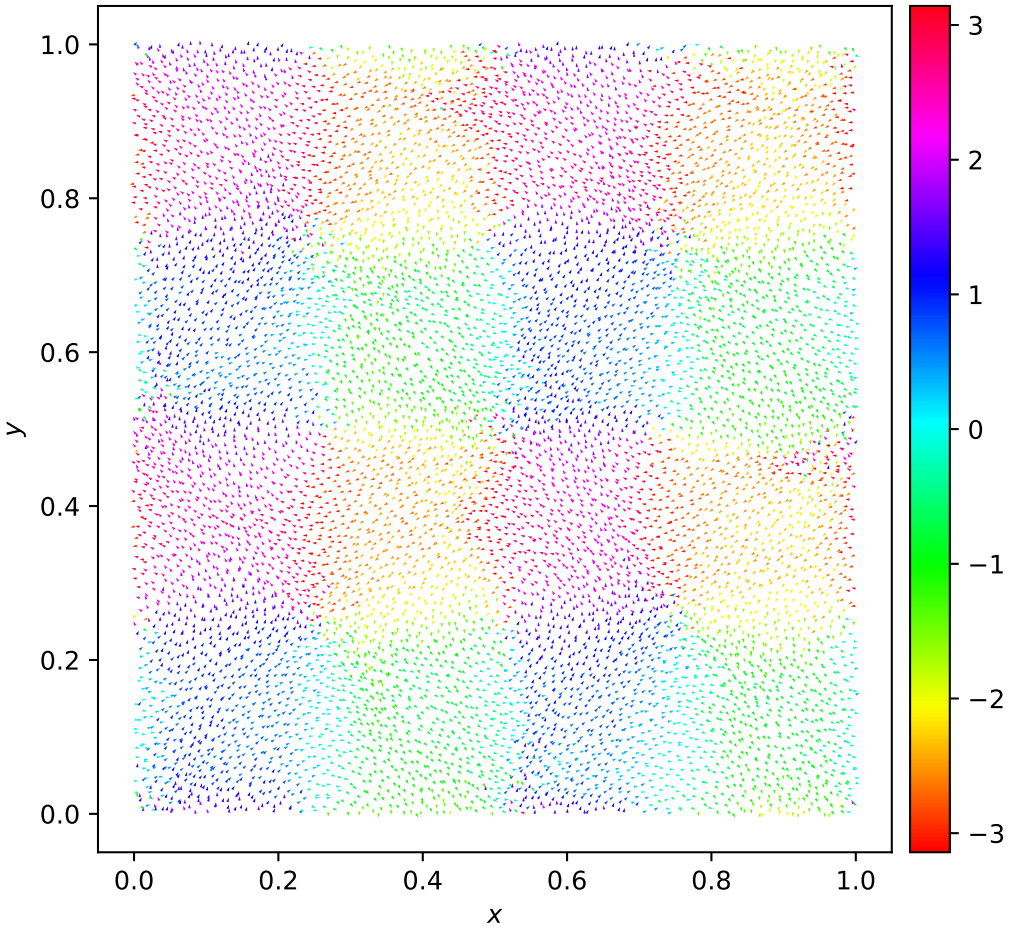
Appendix B

Supplementary Figures



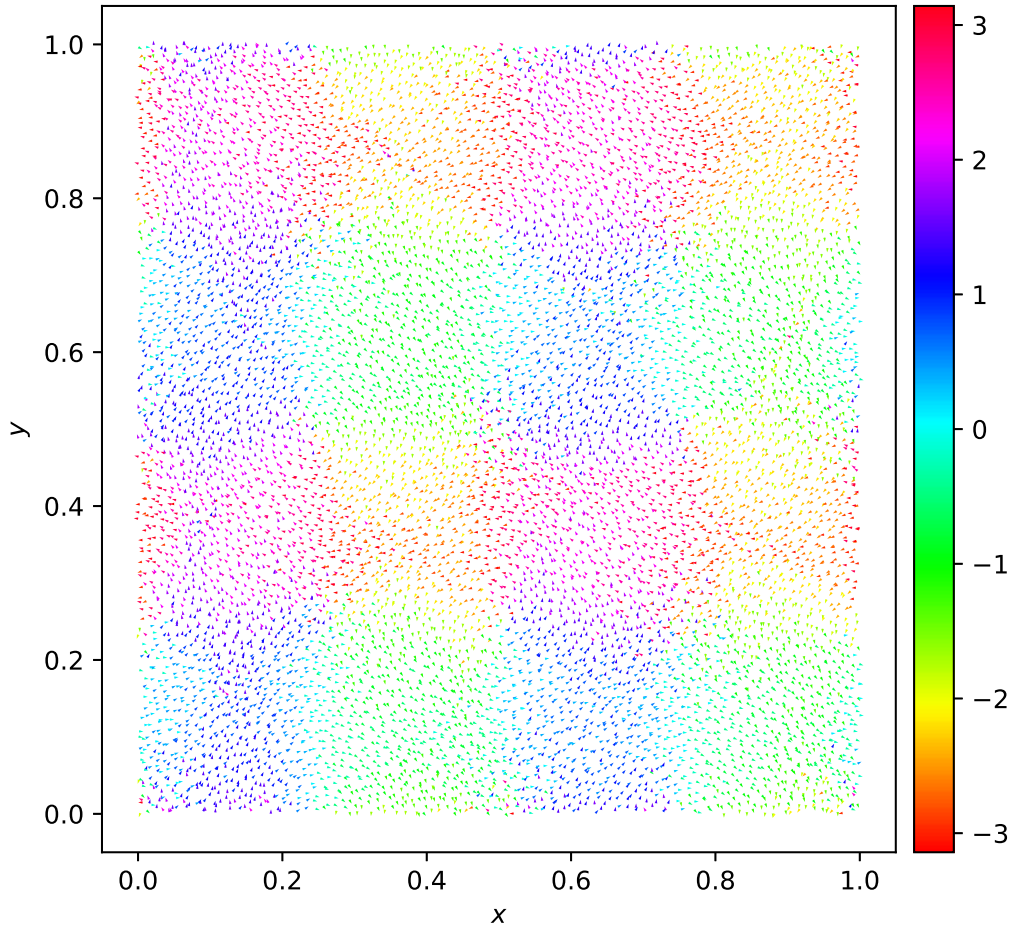
(a)

Figure B.1: Snapshots of the particle velocity data for the discrete compressible fluid simulation at times $t =$ (a) 0, (b) 0.2, (c) 0.4, (d) 0.6. Each arrow indicates the position and direction of the velocity for a single particle, with the color corresponding to angle in radians from the $+x$-axis, $\theta \in (-\pi, \pi]$.



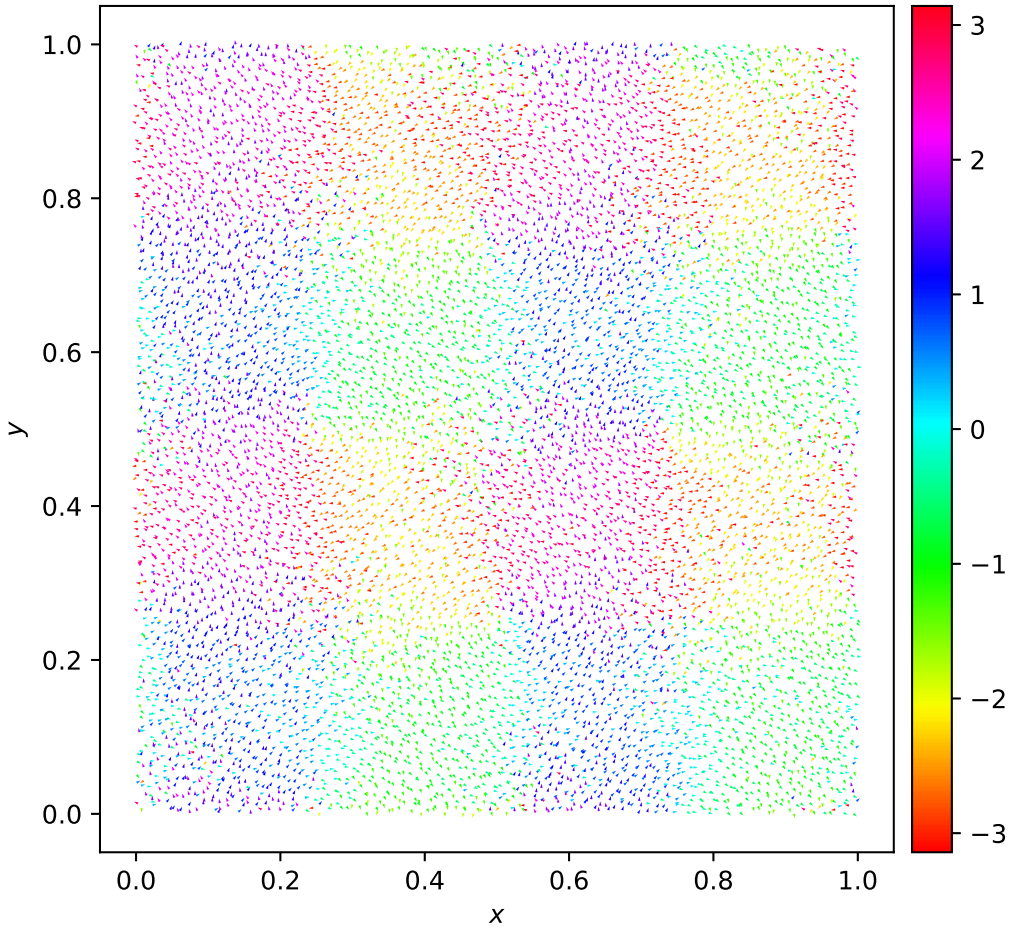
(b)

Figure B.1: Snapshots of the particle velocity data for the discrete compressible fluid simulation at times $t =$ (a) 0, (b) 0.2, (c) 0.4, (d) 0.6. Each arrow indicates the position and direction of the velocity for a single particle, with the color corresponding to angle in radians from the $+x$-axis, $\theta \in (-\pi, \pi]$.



(c)

Figure B.1: Snapshots of the particle velocity data for the discrete compressible fluid simulation at times $t =$ (a) 0, (b) 0.2, (c) 0.4, (d) 0.6. Each arrow indicates the position and direction of the velocity for a single particle, with the color corresponding to angle in radians from the $+x$-axis, $\theta \in (-\pi, \pi]$.



(d)

Figure B.1: Snapshots of the particle velocity data for the discrete compressible fluid simulation at times $t =$ (a) 0, (b) 0.2, (c) 0.4, (d) 0.6. Each arrow indicates the position and direction of the velocity for a single particle, with the color corresponding to angle in radians from the $+x$-axis, $\theta \in (-\pi, \pi]$.

Bibliography

[1] Hirotugu Akaike. A new look at the statistical model identification. *IEEE transactions on automatic control*, 19(6):716–723, 1974.

[2] E. P. Alves and F. Fiuza. Data-driven discovery of reduced plasma physics models from fully kinetic simulations. *Physical Review Research*, 4(3):033192, September 2022.

[3] Joseph Bakarji and Daniel M. Tartakovsky. Data-driven discovery of coarse-grained equations. *Journal of Computational Physics*, 434:110219, June 2021.

[4] M. Ballerini, N. Cabibbo, R. Candelier, A. Cavagna, E. Cisbani, I. Giardina, V. Lecomte, A. Orlandi, G. Parisi, A. Procaccini, M. Viale, and V. Zdravkovic. Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study. *Proceedings of the National Academy of Sciences*, 105(4):1232–1237, January 2008.

[5] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[6] Lauren Berk and Dimitris Bertsimas. Certifiably optimal sparse principal component analysis. *Mathematical Programming Computation*, 11(3):381–420, September 2019.

[7] Tyrus Berry and John Harlim. Variable bandwidth diffusion kernels. *Applied and Computational Harmonic Analysis*, 40(1):68–96, 2016.

[8] Dimitris Bertsimas and Ryan Cory-Wright. On polyhedral and second-order cone decompositions of semidefinite optimization problems. *Operations Research Letters*, 48(1):78–85, January 2020.

[9] Dimitris Bertsimas, Ryan Cory-Wright, and Jean Pauphilet. Solving large-scale sparse PCA to certifiable (near) optimality. *Journal of Machine Learning Research*, 23(13):1–35, 2022.

[10] Dimitris Bertsimas and Wes Gurnee. Learning sparse nonlinear dynamics via mixed-integer optimization. *Nonlinear Dynamics*, 111(7):6585–6604, 2023.

[11] Dimitris Bertsimas, Angela King, and Rahul Mazumder. Best subset selection via a modern optimization lens. *The Annals of Statistics*, 44(2):813 – 852, 2016.

[12] Dimitris Bertsimas and Bart Van Parys. Sparse high-dimensional regression: Exact scalable algorithms and phase transitions. *The Annals of Statistics*, 48(1):300–323, February 2020.

[13] Dimitris Bertsimas and Robert Weismantel. *Optimization over integers*. Dynamic Ideas, 2005.

[14] Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales. In *International Conference on Machine Learning*, pages 936–945. PMLR, 2021.

[15] Josh Bongard and Hod Lipson. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104(24):9943–9948, June 2007.

[16] Lorenzo Boninsegna, Feliks Nüske, and Cecilia Clementi. Sparse learning of stochastic dynamical equations. *The Journal of Chemical Physics*, 148(24):241723, March 2018.

[17] Ahmad Borzou, Alison E. Patteson, and J. M. Schwarz. A Data-Driven Statistical Description for the Hydrodynamics of Active Matter. *New Journal of Physics*, 23(10):103004, October 2021.

[18] Antoine Bricard, Jean-Baptiste Caussin, Debasish Das, Charles Savoie, Vijayakumar Chikkadi, Kyohei Shitara, Oleksandr Chepizhko, Fernando Peruani, David Saintillan, and Denis Bartolo. Emergent vortices in populations of colloidal rollers. *Nature Communications*, 6(1):7470, June 2015.

[19] Antoine Bricard, Jean-Baptiste Caussin, Nicolas Desreumaux, Olivier Dauchot, and Denis Bartolo. Emergence of macroscopic directed motion in populations of motile colloids. *Nature*, 503(7474):95–98, November 2013.

[20] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, April 2016.

[21] D Burnett. The distribution of molecular velocities and the mean motion in a non-uniform gas. *Proceedings of the London Mathematical Society*, 2(1):382–435, 1936.

[22] Gregory Butler. *Fundamental algorithms for permutation groups*. Springer, 1991.

[23] Markus Bär, Rainer Hegger, and Holger Kantz. Fitting partial differential equations to space-time dynamics. *Physical Review E*, 59(1):337–342, January 1999.

[24] J. L. Callaham, J.-C. Loiseau, G. Rigas, and S. L. Brunton. Nonlinear stochastic modelling with Langevin regression. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 477(2250):20210092, June 2021.

[25] Andrea Cavagna, Alessio Cimarelli, Irene Giardina, Giorgio Parisi, Raffaele Santagati, Fabio Stefanini, and Massimiliano Viale. Scale-free correlations in starling flocks. *Proceedings of the National Academy of Sciences*, 107(26):11865–11870, June 2010.

[26] Sydney Chapman and Thomas George Cowling. *The mathematical theory of non-uniform gases: an account of the kinetic theory of viscosity, thermal conduction and diffusion in gases*. Cambridge university press, 1990.

[27] Shixiang Chen, Shiqian Ma, Lingzhou Xue, and Hui Zou. An alternating manifold proximal gradient method for sparse principal component analysis and sparse canonical correlation analysis. *INFORMS Journal on Optimization*, 2(3):192–208, 2020.

[28] Zhijun Chen and Hayden Schaeffer. Conditioning of random fourier feature matrices: double descent and generalization error. *Information and Inference: A Journal of the IMA*, 13(2):iaad054, 2024.

[29] Miles Cranmer. Interpretable machine learning for science with PySR and SymbolicRegression.jl. *arXiv preprint arXiv:2305.01582*, 2023.

[30] James P Crutchfield and BS McNamara. Equations of motion from a data series. *Complex systems*, 1:417–452, 1987.

[31] Alexandre d'Aspremont, Francis Bach, and Laurent El Ghaoui. Optimal solutions for sparse principal component analysis. *Journal of Machine Learning Research*, 9(7), 2008.

[32] Alexandre d'Aspremont, Laurent El Ghaoui, Michael I. Jordan, and Gert R. G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. *SIAM Review*, 49(3):434–448, 2007.

[33] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[34] David L Donoho, Arian Maleki, and Andrea Montanari. Message-passing algorithms for compressed sensing. *Proceedings of the National Academy of Sciences*, 106(45):18914–18919, 2009.

[35] Renáta Dubčáková. Eureqa: software review. *Genetic Programming and Evolvable Machines*, 12(2):173–178, June 2011.

[36] Marco A Duran and Ignacio E Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical programming*, 36:307–339, 1986.

[37] Jerald L Ericksen. Conservation laws for liquid crystals. *Transactions of the Society of Rheology*, 5(1):23–34, 1961.

[38] Michael D. Ernst, Jeff H. Perkins, Philip J. Guo, Stephen McCamant, Carlos Pacheco, Matthew S. Tschantz, and Chen Xiao. The Daikon system for dynamic detection of likely invariants. *Science of Computer Programming*, 69(1):35–45, December 2007.

[39] Charles Fefferman. A sharp form of Whitney's extension theorem. *Annals of mathematics*, 161(1):509–577, 2005.

[40] Charles Fefferman. Extension of C^m, ω-smooth functions by linear operators. *Revista Matemática Iberoamericana*, 25(1):1–48, 2009.

[41] Charles Fefferman, Arie Israel, and Garving Luli. Sobolev extension by linear operators. *Journal of the American Mathematical Society*, 27(1):69–145, 2014.

[42] Charles Fefferman and Bo'az Klartag. Fitting a C^m-smooth function to data I. *Annals of Mathematics*, pages 315–346, 2009.

[43] E. A. Feigenbaum, B. G. Buchanan, and J. Lederberg. On generality and problem solving: A case study using the DENDRAL program. Technical Report NASA-CR-123182, August 1970.

[44] Oliver Y Feng, Ramji Venkataramanan, Cynthia Rush, Richard J Samworth, et al. A unifying tutorial on approximate message passing. *Foundations and Trends® in Machine Learning*, 15(4):335–536, 2022.

[45] Candida Ferreira. Gene expression programming: a new adaptive algorithm for solving problems. *arXiv preprint cs/0102027*, 2001.

[46] Daan Frenkel and Berend Smit. *Understanding molecular simulation: from algorithms to applications*. Elsevier, 2023.

[47] Lloyd Fung, Urban Fasel, and Matthew Juniper. Rapid bayesian identification of sparse nonlinear dynamics from scarce and noisy data. *Proceedings A*, 481(2307):20240200, 2025.

[48] Robert A Gingold and Joseph J Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly notices of the royal astronomical society*, 181(3):375–389, 1977.

[49] Matthew Golden. Scalable sparse regression for model discovery: The fast lane to insight. *arXiv preprint arXiv:2405.09579*, 2024.

[50] Matthew Golden, Roman O. Grigoriev, Jyothishraj Nambisan, and Alberto Fernandez-Nieves. Physically informed data-driven modeling of active nematics. *Science Advances*, 9(27):eabq6120, July 2023.

[51] Matthew Golden, Kaushik Satapathy, and Dimitrios Psaltis. Scalable Discovery of Fundamental Physical Laws: Learning Magnetohydrodynamics from 3D Turbulence Data, January 2025.

[52] Matthew Ryan Golden. *Physics-Inspired Machine Learning of Partial Differential Equations*. PhD thesis, Georgia Institute of Technology, July 2023.

[53] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.

[54] Harold Grad. On the kinetic theory of rarefied gases. *Communications on pure and applied mathematics*, 2(4):331–407, 1949.

[55] Sumit Gulwani, Oleksandr Polozov, Rishabh Singh, et al. Program synthesis. *Foundations and Trends® in Programming Languages*, 4(1-2):1–119, 2017.

[56] Daniel R. Gurevich, Matthew R. Golden, Patrick A.K. Reinbold, and Roman O. Grigoriev. Learning fluid physics from highly turbulent data using sparse physics-informed discovery of empirical relations (SPIDER). *Journal of Fluid Mechanics*, 996:A25, October 2024.

[57] Daniel R. Gurevich, Patrick A. K. Reinbold, and Roman O. Grigoriev. Robust and optimal sparse regression for nonlinear PDE models. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(10):103113, October 2019.

[58] Brian C Hall. *Lie groups, Lie algebras, and representations*. Springer, 2013.

[59] Jiequn Han, Chao Ma, Zheng Ma, and Weinan E. Uniformly accurate machine learning-based hydrodynamic models for kinetic equations. *Proceedings of the National Academy of Sciences*, 116(44):21983–21991, October 2019.

[60] Chaitanya Joshi, Sattvic Ray, Linnea M. Lemma, Minu Varghese, Graham Sharp, Zvonimir Dogic, Aparna Baskaran, and Michael F. Hagan. Data-Driven Discovery of Active Nematic Hydrodynamics. *Physical Review Letters*, 129(25):258001, December 2022.

[61] Logan M Kageorge, Roman O Grigoriev, and Michael F Schatz. Data-driven detection of drifting system parameters. *arXiv preprint arXiv:2111.12114*, 2021.

[62] Kadierdan Kaheman, J. Nathan Kutz, and Steven L. Brunton. SINDy-PI: a robust algorithm for parallel implicit sparse identification of nonlinear dynamics. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 476(2242):20200279, October 2020.

[63] Jacob Klein. *Greek mathematical thought and the origin of algebra*. Courier Corporation, 1992.

[64] Ken Kobayashi and Yuich Takano. A branch-and-cut algorithm for solving mixed-integer semidefinite optimization problems. *Computational Optimization and Applications*, 75(2):493–513, 2020.

[65] Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.

[66] Ellis Robert Kolchin. *Differential algebra & algebraic groups*. Academic press, 1973.

[67] Nikos Komodakis and Jean-Christophe Pesquet. Playing with duality: An overview of recent primal-dual approaches for solving large-scale optimization problems. *IEEE Signal Processing Magazine*, 32(6):31–54, 2015.

[68] Pat Langley. BACON: A production system that discovers empirical laws. In *IJCAI*, page 344. Citeseer, 1977.

[69] Frank M Leslie. Some constitutive equations for liquid crystals. *Archive for Rational Mechanics and Analysis*, 28:265–283, 1968.

[70] C David Levermore. Moment closure hierarchies for kinetic theories. *Journal of Statistical Physics*, 83:1021–1065, 1996.

[71] Michael James Lighthill. On sound generated aerodynamically i. general theory. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 211(1107):564–587, 1952.

[72] Yuxuan Liu, Jingmin Sun, Xinjie He, Griffin Pinney, Zecheng Zhang, and Hayden Schaeffer. PROSE-FD: A Multimodal PDE Foundation Model for Learning Multiple Operators for Forecasting Fluid Dynamics, September 2024.

[73] Edward N. Lorenz. Deterministic Nonperiodic Flow. *Journal of Atmospheric Sciences*, 20(2):130 – 141, 1963.

[74] David Lovelock. Dimensionally dependent identities. *Mathematical Proceedings of the Cambridge Philosophical Society*, 68(2):345–350, 1970.

[75] Leon B Lucy. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal, vol. 82, Dec. 1977, p. 1013-1024.*, 82:1013–1024, 1977.

[76] Wenjun Ma, Jun Zhang, Kaikai Feng, Haoyun Xing, and Dongsheng Wen. Dimensional homogeneity constrained gene expression programming for discovering governing equations. *Journal of Fluid Mechanics*, 985:A12, April 2024.

[77] Suryanarayana Maddu, Quentin Vagne, and Ivo F Sbalzarini. Learning deterministic hydrodynamic equations from stochastic active particle dynamics. *arXiv preprint arXiv:2201.08623*, 2022.

[78] Niall M Mangan, J Nathan Kutz, Steven L Brunton, and Joshua L Proctor. Model selection for dynamical systems via sparse regression and information criteria. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2204):20170009, 2017.

[79] L. R. U. Manssur and R. Portugal. Group-theoretic Approach for Symbolic Tensor Manipulation: II. Dummy Indices. *International Journal of Modern Physics C*, 13(07):859–879, September 2002.

[80] Georg Martius and Christoph H Lampert. Extrapolation and learning equations. *arXiv preprint arXiv:1610.02995*, 2016.

[81] José M. Martín-García. *xPerm*: fast index canonicalization for tensor computer algebra. *Computer Physics Communications*, 179(8):597–603, October 2008.

[82] Daniel A. Messenger and David M. Bortz. Weak SINDy for partial differential equations. *Journal of Computational Physics*, 443:110525, October 2021.

[83] Daniel A. Messenger and David M. Bortz. Weak SINDy: Galerkin-Based Data-Driven Model Selection. *Multiscale Modeling & Simulation*, 19(3):1474–1497, January 2021.

[84] Daniel A. Messenger and David M. Bortz. Learning mean-field equations from particle data using WSINDy. *Physica D: Nonlinear Phenomena*, 439:133406, November 2022.

[85] Kaisa Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 2012.

[86] Anders Miltner, Saswat Padhi, Todd Millstein, and David Walker. Data-driven inference of representation invariants. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2020, pages 1–15, New York, NY, USA, June 2020. Association for Computing Machinery.

[87] Baback Moghaddam, Yair Weiss, and Shai Avidan. Spectral Bounds for Sparse PCA: Exact and Greedy Algorithms. In *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2005.

[88] Benjamin E Niehoff. Faster tensor canonicalization. *Computer Physics Communications*, 228:123–145, 2018.

[89] Kasper Peeters. A field-theory motivated approach to symbolic computer algebra. *arXiv preprint cs/0608005*, 2006.

[90] Kasper Peeters. Introducing cadabra: a symbolic computer algebra system for field theory problems. *arXiv preprint hep-th/0701238*, 2007.

[91] Kasper Peeters. Cadabra2: computer algebra for field theory revisited. *Journal of open source software*, 3(32), 2018.

[92] Roger Penrose and Wolfgang Rindler. *Spinors and space-time*, volume 1. Cambridge university press, 1984.

[93] F. J. Perdreauville and R. E. Goodson. Identification of Systems Described by Partial Differential Equations. *Journal of Basic Engineering*, 88(2):463–468, June 1966.

[94] JF Peters, M Muthuswamy, J Wibowo, and A Tordesillas. Characterization of force chains in granular material. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 72(4):041307, 2005.

[95] James C. Phillips, David J. Hardy, Julio D. C. Maia, John E. Stone, João V. Ribeiro, Rafael C. Bernardi, Ronak Buch, Giacomo Fiorin, Jérôme Hénin, Wei Jiang, Ryan McGreevy, Marcelo C. R. Melo, Brian K. Radak, Robert D. Skeel, Abhishek Singharoy, Yi Wang, Benoît Roux, Aleksei Aksimentiev, Zaida Luthey-Schulten, Laxmikant V. Kalé, Klaus Schulten, Christophe Chipot, and Emad Tajkhorshid. Scalable molecular dynamics on CPU and GPU architectures with NAMD. *The Journal of Chemical Physics*, 153(4), July 2020.

[96] Anthony Ralston and Philip Rabinowitz. *A first course in numerical analysis*. Courier Corporation, 2001.

[97] Patrick A. K. Reinbold, Daniel R. Gurevich, and Roman O. Grigoriev. Using noisy or incomplete data to discover models of spatiotemporal dynamics. *Physical Review E*, 101(1):010203, January 2020.

[98] Patrick A. K. Reinbold, Logan M. Kageorge, Michael F. Schatz, and Roman O. Grigoriev. Robust learning from noisy, incomplete, high-dimensional experimental data via physically constrained symbolic regression. *Nature Communications*, 12(1):3219, May 2021.

[99] Samuel H. Rudy, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, April 2017.

[100] Subham Sahoo, Christoph Lampert, and Georg Martius. Learning equations for extrapolation and control. In *International Conference on Machine Learning*, pages 4442–4450. PMLR, 2018.

[101] Hayden Schaeffer. Learning partial differential equations via data discovery and sparse optimization. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2197):20160446, January 2017.

[102] Michael Schmidt and Hod Lipson. Distilling Free-Form Natural Laws from Experimental Data. *Science*, 324(5923):81–85, April 2009.

[103] Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, pages 461–464, 1978.

[104] Yoshio Sone. *Kinetic theory and fluid dynamics*. Springer, 2002.

[105] Robert Stephany and Christopher Earls. Weak-PDE-LEARN: a weak form based approach to discovering pdes from noisy, limited data. *arXiv preprint arXiv:2309.04699*, 2023.

[106] Robert Stephany and Christopher Earls. PDE-LEARN: Using deep learning to discover partial differential equations from noisy, limited data. *Neural Networks*, 174:106242, June 2024.

[107] Weijie Su, Małgorzata Bogdan, and Emmanuel Candes. False discoveries occur early on the lasso path. *The Annals of statistics*, pages 2133–2150, 2017.

[108] Jingmin Sun, Yuxuan Liu, Zecheng Zhang, and Hayden Schaeffer. Towards a foundation model for partial differential equations: Multi-operator learning and extrapolation. *arXiv preprint arXiv:2404.12355*, 2024.

[109] Rohit Supekar, Boya Song, Alasdair Hastewell, Gary P. T. Choi, Alexander Mietke, and Jörn Dunkel. Learning hydrodynamic equations for active matter from particle simulations and experiments. *Proceedings of the National Academy of Sciences*, 120(7):e2206994120, February 2023.

[110] John Lighton Synge and Alfred Schild. *Tensor calculus*, volume 5. Courier Corporation, 1978.

[111] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

[112] John Toner and Yuhai Tu. Long-Range Order in a Two-Dimensional Dynamical XY Model: How Birds Fly Together. *Physical Review Letters*, 75(23):4326–4329, December 1995.

[113] John Toner and Yuhai Tu. Flocks, herds, and schools: A quantitative theory of flocking. *Physical Review E*, 58(4):4828–4858, October 1998.

[114] Lloyd N Trefethen and JAC Weideman. The exponentially convergent trapezoidal rule. *SIAM review*, 56(3):385–458, 2014.

[115] Xavier Trepat, Michael R. Wasserman, Thomas E. Angelini, Emil Millet, David A. Weitz, James P. Butler, and Jeffrey J. Fredberg. Physical forces during collective cell migration. *Nature Physics*, 5(6):426–430, June 2009.

[116] Wu-Ki Tung. *Group theory in physics*, volume 1. World Scientific, 1985.

[117] Harsha Vaddireddy, Adil Rasheed, Anne E. Staples, and Omer San. Feature engineering and symbolic regression methods for detecting hidden physics from sparse sensor observation data. *Physics of Fluids*, 32(1):015113, January 2020.

[118] Marco Virgolin and Solon P Pissis. Symbolic regression is np-hard. *arXiv preprint arXiv:2207.01018*, 2022.

[119] C. J. Wareing, A. T. Roy, M. Golden, R. Grigoriev, and S. M. Tobias. Data-driven discovery of the equations of turbulent convection. *Geophysical and Astrophysical Fluid Dynamics*, 2025. Under review.

[120] Hassler Whitney. Analytic extensions of differentiable functions defined in closed sets. *Transactions of the American Mathematical Society*, 36:63–89, 1934.

[121] Hassler Whitney. Differentiable functions defined in closed sets I. *Transactions of the American Mathematical Society*, 36:369–389, 1934.

[122] Hassler Whitney. Functions differentiable on the boundaries of regions. *Annals of Mathematics*, 35:482–485, 1934.

[123] James Hardy Wilkinson. *The algebraic eigenvalue problem*. Oxford University Press, Inc., 1988.

[124] Yuan Xia, Jyotirmoy V Deshmukh, Mukund Raghothaman, and Srivatsan Ravi. Data-driven template-free invariant generation. *arXiv preprint arXiv:2312.17527*, 2023.

[125] Haoyun Xing, Jun Zhang, Wenjun Ma, and Dongsheng Wen. Using gene expression programming to discover macroscopic governing equations hidden in the data of molecular simulations. *Physics of Fluids*, 34(5):057109, May 2022.

[126] Daolin Xu and Omid Khanmohamadi. Spatiotemporal system reconstruction using Fourier spectral operators and structure selection techniques. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 18(4):043122, December 2008.

[127] Chen Yao and Erik M. Bollt. Modeling and nonlinear parameter estimation with Kronecker product representation for coupled oscillators and spatiotemporal systems. *Physica D: Nonlinear Phenomena*, 227(1):78–99, March 2007.

[128] Haruo Yoshida. Construction of higher order symplectic integrators. *Physics letters A*, 150(5-7):262–268, 1990.

[129] Xiao-Tong Yuan and Tong Zhang. Truncated power method for sparse eigenvalue problems. *The Journal of Machine Learning Research*, 14(1):899–925, 2013.