MATHEMATICAL APPROACHES TO IDENTIFICATION PROBLEMS – COUNTING, RNA FOLDING, AND PDE IDENTIFICATION

A Dissertation
Presented to
The Academic Faculty

By

Mengyi Tang

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Mathematics
Department of Mathematics

Georgia Institute of Technology

Dec 2023

MATHEMATICAL APPROACHES TO IDENTIFICATION PROBLEMS – COUNTING, RNA FOLDING, AND PDE IDENTIFICATION

Thesis committee:

Dr. Sung Ha Kang, Advisor
School of Mathematics
Georgia Institute of Technology

Dr. Luca Dieci
School of Mathematics
Georgia Institute of Technology

Dr. Wenjing Liao
School of Mathematics
Georgia Institute of Technology

Dr. Haomin Zhou
School of Mathematics
Georgia Institute of Technology

Dr. Rachel Kuske
School of Mathematics
Georgia Institute of Technology

Date approved: Nov 7, 2023

In my mathematical odyssey, beauty flourishes within the ceaseless joy of exploration.

For my parents and Miles

ACKNOWLEDGMENTS

First and foremost, I would like to express thanks to my esteemed research advisor, Dr. Sung Ha Kang. Her guidance and support have helped me reach where I am today. Dr. Kang has had a significant impact on how I approach research and my attitude towards it. Her mentorship has not only contributed to my academic growth but has also taught me how to think and do research more independently. She has also shown care and support in my personal life. I've learned valuable lessons from her that I will carry with me throughout my entire life.

In addition, I wish to express my gratitude to Dr. Liao and, in particular, to Dr. Kuske, Dr. Liu, my dedicated advisor, for their instrumental roles in our work within the differential equation identification project. Our countless discussions and the substantial time we invested in this project have had a significant impact on my research. I would also like to extend my heartfelt thanks to Dr. Yashtini, with whom I collaborated closely on the Counting Project. Her expertise and collaborative spirit were invaluable in the achievement of our collective goals.

Special recognition is also extended to my parents, whose unwavering support has been a constant presence in my life. Additionally, I am deeply grateful to my husband, Milosz Rajchel, for his exceptional support and encouragement. Miles has provided me with the strongest mental support, allowing me to grow both personally and academically. His belief in my abilities and his encouragement to pursue my passions have been instrumental in my success. Lastly, I wish to acknowledge all my cherished friends who have been my steadfast companions throughout this academic journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Mathematical algorithms have become an essential tool in uncovering hidden patterns and unraveling dynamic behaviors within complex datasets, aiding in gaining deeper insights and making informed choices in an era driven by data-driven decision-making. In this thesis, we present several works that utilize numerical algorithms in identification problems derived from mathematical models. These works place a specific emphasis on the identification and prediction of structures and patterns within various type of datasets, while also offering the capacity to forecast the behavior of future data. Chapter 2 illustrates our work in predicting the secondary structure of RNA sequences, where a novel idea of using Stem and the *Clique* structure is introduced to form a *Stem-graph* to represent the secondary structure of an RNA sequence. In Chapter 3, we propose a new method, Counting Objects by Diffused Index (CODI), which is an application of diffusion algorithm to count objects in digital images. An efficient algorithm is proposed based on an operator-splitting approach and the alternating direction minimization method. Chapter 4 and Chapter 5 focus on identifying differential equations in the physical domain and frequency domain, respectively. Both methods use Subspace Pursuit and weak formulation of features to stabilize the output support and handle noises. Chapter 4 proposes a general and robust framework to identify differential equations using a weak formulation with two new mechanisms, narrow-fit and trimming, for both ordinary and partial differential equations (ODEs and PDEs) in the physical domain. Chapter 5 explores the benefits and challenges of utilizing the frequency domain in differential equation identification. In this work, we introduce Fourier feature denoising and define the meaningful data region and the core regions of features to eliminate noise from the frequency domain. Additionally, We introduce a group trimming step to refine the support of new energy based on the core regions of features for coefficient identification. Comprehensive experiments are presented for both methods to demonstrate the benefits. Chapter 6 provides conclusions and discussion upon

all the works presented and potential future directions.

This thesis includes published work from three sources: [1] in chapter 2, [2] in chapter 3, and [3] in chapter 4.

CHAPTER 1
INTRODUCTION

Mathematical algorithms have gained increasing importance and are used in widespread applications across various domains. These algorithms integrate diverse aspects of mathematics, such as discrete mathematics, geometry, and differential equations, enabling enhanced data comprehension and the potential to forecast future or unseen data patterns. In this thesis, we focus on mathematical algorithms in identification problems.

First, we explore a problem related to RNA structure. RNA contributes to cellular life with its profound significance in human biology. Predicting RNA's cellular functions can be useful in understanding the interaction dynamics among nucleotide bases linked by phosphate groups and polysaccharide molecules. Here, each RNA sequence comprises nucleotide bases denoted by the characters A, U, C, and G, which can be meticulously folded into distinct structural components. These structural components are instrumental in various biological activities. A lot of methods have been proposed to predict the secondary structure of the sequences. This includes dynamic programming algorithm [4] with Minimum Free Energy (MFE), Maximum Expected Accuracy (MEA), combination of MFE and MEA [5] , single-sequence analysis, multiple-sequence analysis, and deep learning. In chapter 2, we define the concept of stem as a fundamental structural unit in RNA, defined by consecutive base pairs, and explore its significance and its relationship with other characteristics within the secondary folding structure of RNA. We further developed a novel deterministic methodology called StemP to predict the secondary structure of RNA sequences. This method works by treating stems as vertices in a graph and predicts a stem-based graph that connects these vertices based on their coexistence. The idea of using graph building blocks gives StemP the beauty of being simple and deterministic. To enhance the accuracy and stability of structure predictions, we further propose the utiliza-

tion of two key stem characteristics, the minimum stem length, and the Stem-Loop score, especially for short RNA and tRNA sequences. The main idea behind StemP is to consider all possible stems with certain stem-loop energy and strength to predict RNA secondary structure. A full Stem-graph presents all possible folding structures, from which we pick sub-graph(s) that yield the best energy match for structure prediction. Stem-Loop score adds structure information and speeds up the computation. In addition, we propose using Generalized-Stem-Loop score, which represents the total strength of basepair matching over the total length of the sequence enclosed several stems, to capture the complex folding structures in the longer sequences. The proposed method can predict secondary structure even with pseudo knots. We also provide numerical experiments on various sequences from Protein Data Bank and the Gutell Lab using a laptop, and results take only a few seconds.

Another interesting problem is about cells counting. As a fundamental building block of life, cells play an important role in various biological processes. Automating the process of quantifying the cells can aid biologists in tracking the growth and activities of specific biological behaviors. There are traditional methods, such as watershed [6] and Hough Transform [7], that work well for objects with uniform characteristics, clear shapes, and distinct background colors. This counting task can also be categorized into detection-oriented techniques and deep learning-based methods. For example, integrating representative methods [8] and Principal Component Analysis combined with histogram processing [9] are two types of detection-oriented methods. Deep learning can be used to create density maps from image patches, extracting features like texture and gradients for counting [10, 11, 12, 13, 14]. Most of the above works aim to identify the objects in a given image, which leads to certain additional work and limitations due to the texture of objects. In contrast, we aim to automate the process of quantifying objects, emphasizing the total number is our goal in chapter 3. We propose a novel diffusion-based, geometry-independent, and training-free method to count the number of objects in images, inspired by color inpainting problems. This method can be applied to various types of images, while most of the

existing methods, including learning-based methods, are problem-specific. The main idea is to represent each object by a unique index value regardless of its intensity or size and to simply count the number of index values. First, we place different vectors, referred to as seed vectors, uniformly throughout the mask image. Secondly, the seeds are diffused using an edge-weighted harmonic variational optimization model within each object. We propose an efficient algorithm based on an operator splitting approach and alternating direction minimization method. For computational efficiency, we stop the diffusion process before a full convergence and propose to cluster these diffused index values. We refer to this approach as Counting Objects by Diffused Index (CODI). We use Gaussian fitting in histograms, and a high-dimensional clustering method for the final step of counting via clustering. We present counting results in various applications such as biological cells, agriculture, concert crowds, and transportation. Some comparisons with existing methods are also presented in this chapter.

We also explore a different aspect of imaging in relation to the dynamic generated from differential equations. Consider taking a video, where images are captured at specific intervals, creating a series of signals. Within this series of signals, objects or patterns vary with respect to time. The changes in each pixel, akin to a single signal in the one-dimensional case, can be modeled by a mathematical differential equation and identified. There has been an increasing interest in discovering physical or biological dynamics from complex data. The discovery of differential equations can offer important insights into contemporary neuroscience [15], fluid mechanics, physical systems[16, 17], and biology [18]. Different methods in the area of identifying differential equations include symbolic regression [16, 17], optimization approach in [19, 20, 21], sparse regression [22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37]. Recent progress [35, 38, 36, 37] has shown the benefit of using a weak/integral formulation in addressing noise in sparse coefficient identification. The weak form of features involves using integral features instead of differential features to enforce denoising effects with a proper test function. In Chapter

4, we propose a new work, WeakIdent, to identify the differential equations governing specific behaviors given time-dependent sequences. This chapter is focused on the problem of identifying differential equations in the physical domain, especially when the given data are corrupted by noise. We model the identification problem as solving a linear system, with the feature matrix consisting of linear and nonlinear terms multiplied by a coefficient vector. This product is equal to the time derivative term and thus generates dynamical behaviors. Then, the goal is to identify the correct terms that form the equation to capture the dynamics of the given data. We propose a general and robust framework to recover differential equations using a weak formulation for both ordinary and partial differential equations(ODEs and PDEs). The weak formulation facilitates an efficient and robust way to handle noise, and two new mechanisms, narrow-fit and trimming, improve the coefficient support and value recoveries, respectively. For each sparsity level, Subspace Pursuit is utilized to find an initial set of support from the large dictionary. Then, we focus on highly dynamic regions (rows of the feature matrix) and error normalize the feature matrix in the narrow-fit step. The support is further updated by trimming the terms that contribute the least. Finally, the support set of features with the smallest Cross-Validation error is chosen as the result. A comprehensive set of numerical experiments are also presented for both systems of ODEs and PDEs with various noise levels. The proposed method gives a robust recovery of the coefficients and a significant denoising effect, which can handle up to 100% noise-to-signal ratio for some equations. We also compare the proposed method with several state-of-the-art algorithms for the recovery of differential equations.

In Chapter 5, we delve deeper into identifying differential equations by presenting FourierIdent, a method designed to identify a partial differential equation within the frequency domain. We illustrate how we address these more challenging tasks as it expands our investigation from studying physical characteristics (in WeakIdent) to analyzing frequency-based features. We investigate the benefits and challenges of utilizing the frequency domain in differential equation identification. Similar to WeakIdent, assuming

that a single noisy realization of space and time-dependent data is given, we consider the underlying differential equation to be a linear combination of various linear and nonlinear differential and polynomial terms. Identifying differential equations in the frequency domain is more challenging due to large magnitudes and sensitivity against noise. To address these challenges, we introduce a Fourier feature denoising technique and define the meaningful data region and the core regions of features to eliminate noise from the frequency domain. We use Subspace Pursuit on the core region of the time derivative of the given data and introduce a group trimming step to refine the support. We further introduce a new energy based on the core regions of features for coefficient identification. Utilizing the core regions of features serves two critical purposes. It enhances the accuracy of identified coefficients by eliminating low-frequency noise response regions and facilitates the identification of stable coefficients, which minimizes the residual error. The proposed method is tested on various differential equations with linear, nonlinear, and high-order derivative feature terms. Our results demonstrate the distinct advantages of the proposed method, particularly on complex and highly corrupted datasets.

In summary, this thesis offers novel mathematical insights and algorithms for modeling identification problems with broad applicability in physics, imaging, and biology. These contributions advance our understanding and problem-solving capabilities across a spectrum of scientific disciplines, bringing new possibilities for innovation and discovery.

CHAPTER 2

STEMP - PREDICTING RNA SECONDARY STRUCTURE

This chapter reproduces our previously published paper [1]. The author of this thesis personally contributed to Data curation, Investigation, Methodology, Software, Visualization, and Writing – original draft.

We propose Stem-graph based structure Prediction (StemP). The benefit of the proposed method is to explicitly study the main contributions of folding process, such as minimum stem length, Stem-Loop score and co-existence of stems. We first build the graph which represents all possible folding structure of the sequence, which we refer to as the full Stem-graph. Then, we extract a sub-graph or multiple sub-graphs which has the best matching energy for folding structure prediction. We introduce the Stem-Loop score to give structure information in vertices construction, and to make algorithm computationally more efficient.

The main contributions of this chapter is to propose a simple, flexible, efficient and deterministic method for folding structure prediction.

- This method can handle pseudo-knots and 3D structure naturally without any other modification.
- The proposed method is computationally efficient that for sequences of length smaller than 200, results can be computed within one second.
- The proposed method is flexible, and applicable to predict structures of short RNA sequences, tRNA sequences, and rRNA 5S sequences.

This chapter is organized as follows. In Section 2.2, we give a general outline and details of Stem-graph approach, including how to construct the vertices and edges utilizing Stem-Loop score. In Section 2.4.1, Section 2.4.2 and Section 2.4.3, we present details and

comparison results for RNA sequences from Protein Data Bank (PDB), tRNA sequences, and 5S rRNA sequences respectively.

2.1 Literature Review

There have been a wide range of literature on prediction of the secondary structure of a RNA sequence. Minimizing Free Energy (MFE) [39] is applicable for large molecules in terms of efficiency and is widely extended. For an efficient computation, dynamic programming is suggested in [40], which conquers numerous possible folding structures by dividing them into smaller sub-problems. A practical dynamic algorithm can be found in [41]. Mfold [42] is a well-known method dynamic programming method using thermodynamic parameters. A standard dynamic programming algorithm can suffer from time complexity $O(N^3)$ (N represents the length of a RNA sequence) [43] and accuracy. The time complexity of dynamic programming was improved in a Four-Russians method [44] exploiting the number of sub-sequences belonging to an optimal folding set and the maximum number base-pairs. A more recent work [45] on dynamic programming claims to be the first to achieve linear in time and space complexity and shows it's benefits in predicting long sequences such as 16s and 23s. In [46], an optimal structure is assembled based on Maximum Energy Accuracy (MEA) without using dynamic programming algorithm. A novel method based on both MFE and MEA is proposed in [47] for experimental probing data, and structural probing data is incorporated in related work such as [48] as supplementary information to enhance accuracy. Recent studies[49] suggests replacing MEA by a partition function-based algorithm Threshokont and shows that this thresholded version of ProbKnot can give accurate prediction. Another work in linear partition can be found in [50].

We explore single sequence approach to focus on the effect of stems in each sequence. Single sequence analysis approach includes, CONTRAfold[51] which uses fully-automated statistical learning algorithms to evolve model parameters instead of relying on thermody-

namics, and CyloFold[52], an approach based on reproducing the folding procedure in a coarse-grained manner by choosing folding structures based on free energy contribution. Related work on single sequence analysis includes [53, 54, 55, 56], and we refer to [57] for more details in various categories. Alternatively, comparative methods reduce the space of possible folding structure by using evolutionary approaches [58, 59, 60],[61], and recently, various machine learning techniques are developed, e.g. [62, 63, 64].

To represent the structure, we construct a Stem-graph, where any possible stems are represented as vertices of the graph, and edges represents all possible co-existences among the vertices. This setting is similar to [65], where a maximum clique finding algorithm is implemented to assemble compatible conserved stems among multiple sequences. Multiple possible optimal structures are assembled in topological order according to their compatibility among k sequences. This vertex-edge representation is considered as dual graph and analyzed in [66], which reveals the importance of such labeled dual graph in RNA structure identification and biotechnological applications. In [67], a graph based method formulated the structure prediction problem as a maximum weighted clique finding problem to predict RNA complex consisting multiple RNA sequences by considering the best the combination with respect to free energy and RNA-RNA interactions. In [68], the author employs the idea of tree graphs for archiving RNA tree motifs and dual graphs for general RNA motifs. A graph mining algorithm is proposed in [69] to detects all the possible motifs exhaustively. One important advantage of these graph methods is that it allows the existence of pseudo-knots in plausible folding structures [65, 68, 70]. In general, pseudo-knots is particularly difficult to identify efficiently since it leads to a structure with at least two helical stems. With the help of a graphic representation, a pseudo-knot can be efficiently considered as a possible folding structure without further annotation.

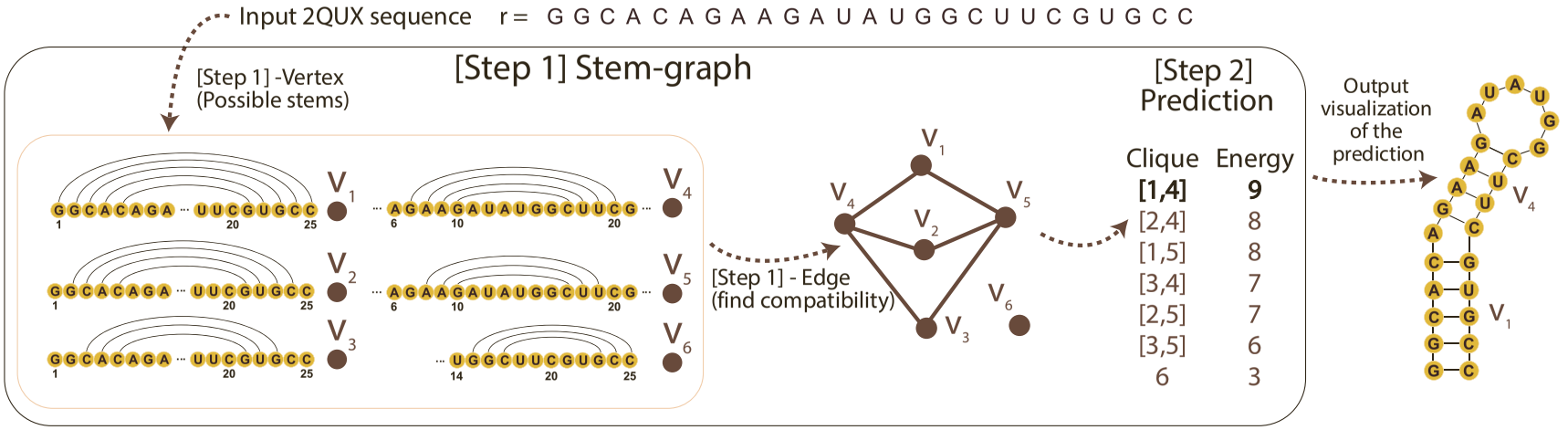


Figure 2.1: Outline of StemP with an example of sequence 2QUX from Protein Data Bank, From the input sequence, in [Step 1] the Stem-graph is constructed where vertex represents stem and edges represent co-existences. [Step 2] cliques from the full Stem-graph are ordered, then maximum energy one is picked as a prediction.

2.2 StemP Methodology

We briefly review some definitions. *A complete graph* is a simple undirected graph in which every pair of distinct vertices is connected by a unique edge. *A clique* in an undirected graph is a subset of vertices, such that every two distinct vertices are adjacent. It's induced subgraph is complete. *Maximal Clique* is a clique which cannot be extended by including any more adjacent vertices.

The proposed Stem-graph based Prediction algorithm has two steps: [Step1] Stem-graph construction which presents all possible folding structure from the given sequence, where Stem-Loop score is used in vertex construction, and [Step2] Prediction, where we pick an optimal folding structure among maximal cliques. Figure 2.1 represents the outline of StemP, using an example of 2QUX (structural protein/RNA) from Protein Data Bank :

$$\texttt{r = GGCAC AGAAG AUAUG GCUUC GUGCC.} \tag{2.1}$$

[Step1] Stem-graph construction. First, any possible stems are assigned as a vertex. We add two pieces of information for vertex construction, minimum stem length L and Stem-Loop score (Equation (2.3)). We consider the canonical base-pair matching, i.e., only `A-U` and `C-G`, unless mentioned otherwise in the later sections. A stem is constructed if there are at least L number of consecutive base pairs matching. The integer L represents the minimum stem length to consider for each vertex. In the case of (Equation (2.1)) we set $L = 3$, starting from the first base `G1`, we search for the first base `C` in this sequence after `G1` which matches it. If the pair `G1-C3` is matched, since `G2` is not match to any other bases to form a stem including `G1-C3`, the stem length is $L = 1$, and this is not considered as a vertex. A stem starting from `G1` with $L \geq 3$ is only possible for `C25`. In fact for this case, consecutive 5 base pairs, `G1-C25`, `G2-C24`, `C3-G23`, `A4-U22`, and `C5-G21`, match to form a stem with length 5. We consider the longest stem which can be constructed from the starting `G1` and ending bases `C25`, and this is assigned as the first vertex v_1. If the first

base G1 forms other different stems, they are assigned as separate vertices, e.g., v_2, etc. In this example, v_1 is the unique vertices starting from G1. Once all possible stems starting with the first base G1 are found and assigned as vertices, we move to the second base G2 and repeat the process. Then, we move to the third base C3 and repeat, i.e., sequentially consider each base one by one to find all vertices. Figure 2.1 **[Step 1]-Vertex** shows all vertex for 2QUX. We note that the stem formed from the first base G1 may include the stem formed from the second base G2, as a sub-stem, e.g. v_1, v_2 in Figure 2.1. We consider these to be two different vertices.

Each vertex stores it's corresponding property, for example:

$$v_1 = (i_1, j_1, l_1, d_1) = (1, 25, 5, 24) \tag{2.2}$$

here i_1 is the starting base number, j_1 the ending base number, l_1 the length of the stem (the number of consecutively matched bases), and $d_1 = j_1 - i_1$ the distance between the starting and the ending bases. We use the ratio between l_1 and d_1 to define Stem-Loop score in Equation (2.3). For short sequences Stem-Loop score is not needed, while for longer sequence, we define a range of Stem-Loop score to only consider biological or physically meaningful stems for vertex construction. This also helps to reduce the number of vertex for an efficient computation.

After all possible stems are found and represented as vertices, in **[Step1]-Edges**, edges are constructed based on the co-existing possibilities between vertices. For example, v_2 and v_3 are sub-stems of v_1 that they cannot co-exists, but v_1 and co-exits with v_4. Figure 2.1 **[Step 1]-Edge** considers all such possibility and e_{14}, e_{24}, e_{34}, e_{15}, e_{25}, e_{35} are constructed in the full Stem-graph. Note that in this full Stem-graph (i) every sub-graph represents different folding structure, and that (ii) pseudo knots and 3D prediction comes naturally without any extra consideration.

[Step2] Prediction. With all possible structure presented in the full Stem-graph, struc-

ture prediction is to choose a sub-graph satisfying certain energy. Since plausible structure requires compatibility between every two vertices, for all sub-graphs with multiple vertices, only a complete sub-graph can be recognized as a possible structure.

> We consider *maximal cliques* of the full Stem-graph as possible folding structures. We sort all maximal clique by the total number of base-pair matching as a simple energy.

The principle idea of Stem-graph Prediction is to find the maximum matching among the stable structures, which is maximal cliques among complete subgraphs since it represent the most complete structure where at least one vertex is a part of.

For 2QUX (Equation (2.1)), as shown in Figure 2.1 [Step2], there are 7 cliques in total. The full Stem-graph of 2QUX only have 2-vertex cliques (six of them), and no 3-vertex clique, and one 1-vertex clique. The maximal clique constructed by v_1 and v_4 is the maximum base-pair matching with maximum energy 9. This is picked as the result of StemP, which matches with the true folding.

For numerical computation for finding maximal cliques, we employ a modified Bron-Kerbosch algorithm with pivoting [71] to find maximal cliques of the full Stem-graph. In this algorithm, depth-first search algorithm with pruning methods are implemented based on Bron and Kerbosch algorithm [72], which is a recursive backtracking algorithm. The time complexity of the worst-case time is $O(3^{\frac{n}{3}})$ for an undirected graph with n vertex. The problem of finding cliques is also studied in many research area such as social network, bioinformatics, computer vision and computational topology. Another improved methods based on Bron-Kerbosch algorithm can be find in [73].

StemP gives a deterministic folding structure prediction, while showing all possible folding structure in a compact full Stem-graph. In practice, to reduce the number of vertices for more efficient computation, we use the minimum stem length $L \geq 2$, and Stem-Loop score which we introduce in next subsection.

Stem-Loop score and Generalized Stem-Loop score When constructing Stem-graph, we further utilize structure information to reduce the number of vertices. We introduce the Stem-Loop score for each vertex v_k:

$$SL(v_k) = \frac{\text{the total length of vertex } v_k}{\text{the stem length of vertex } v_k} = \frac{d_k}{l_k}. \tag{2.3}$$

The main idea behind this score is to explore the energy balance between the stem length l_k and the size of the loop the stem encloses. The stem length l_k is not too small compared to the size of the loop d_k, since there will not be enough binding force to hold the loop stable. The stem length l_k is not too large compared to the size of the loop d_k, since it is somewhat unnatural. We observed that SL values among the same type of sequences are similar, and we can learn the range of this value from known sequences. This is in spirit similar to recent trend of active learning methodology such as Neural Network [62, 64] or Motif analysis[74, 75], that one can learn Stem-Loop score from a known sequence, and biological property can be added.

For a long sequence like rRNA 5S, Stem-Loop score itself may not be enough to capture the complex structure of the folding, such as stems enclosing another stems. We generalize SL to include such structures. We define a set of vertices, V_k, which is understood as a structure starting from a base stem leading up to one hairpin loop, including all internal loop and bulge in between. This set $V_k = \{v_{k_1}, ...v_{k_m}\}$ represents a set of several stems in which one encloses the rest of the stems. For this set V_k, we consider the following Generalized-Stem-Loop score:

$$GSL(V_k) = GSL(v_{k_1}, ...v_{k_m}) = \frac{d_{k_1}}{l_{k_1} + ... + l_{k_m}}. \tag{2.4}$$

The first vertex v_{k_1} encloses all the other vertices $v_{k_2}, ...v_{k_m}$ (i.e. $a_{k_1} < a_{k_j}, b_{k_1} > b_{k_j}$ for all $j > 1$). Here d_{k_1} denotes the total length of V_k, and $l_{k_1} + ... + l_{k_m}$ denotes the sum of stem length within V_k. GSL represents the total strength of basepair matching over the total

length of the sequence enclosed in V_k. We discuss details in Section 2.4.3 for 5s rRNA.

The case such as tRNA Acceptor, the sequences have a self closing form, that is there is a stem connecting starting and ending bases of the entire sequence. When the total length d_k of the stem-loop is sufficiently large to enclose the whole sequence, we consider the Acceptor-Stem-Loop score:

$$ASL(v_k) = \frac{\hat{l} - d_k + 2l_k - 2}{l_k}, \tag{2.5}$$

where $\hat{l}$ is the size of the given sequence. This is a way to account for the open end closing stems. We consider the open-end loop to be closed and compute the Stem-Loop score in the opposite direction of a normal stem.

2.3 StemP Algorithm

In this section, We present the outline of the StemP algorithm and sub-algorithms for tRNA and 5s rRNA. One of the best part of this algorithm is its simplicity: a simple code can be easily written and experiments can be done on a regular laptop for the sequence of length up to 150.

In Algorithm 1 [Step 1] constructs vertex and edges respectively. Then we employ Bron-Kerbosch algorithm with pivoting [71] to find the cliques of the full Stem-graph. Along with the bound of *Stem-Loop score*, Length Threshold L of a stem is introduced to narrow down the choice of possible vertex. This gives priority to overall skeleton of the folding structure, which is driven from big binding forces, i.e. by longer stems. The longer the sequence is, the longer this threshold can be. This Algorithm 1 is a general algorithm that can be applied to any sequences with unknown structure.

Algorithm 2 illustrates the details in finding partial stems when predicting tRNA sequences. An example of predicting Archaeal 5s rRNA sequence can be found in Algorithm 3. This algorithm is based on the results of vertex construction for 5 helix H_1, H_2, H_3, H_4, H_5

with minimum Stem Length L. Both Algorithm 2 and Algorithm 3 are an extension of Algorithm Algorithm 1 which utilized the known structure of a given type of RNA sequences to increase accuracy and computing efficiency. Example of finding particular structure $l_{1[n_1/n_2]}l_2$ vertex can be found in Algorithm 4.

2.4 Numerical Experiments

Comparison Measure for Numerical Experiments

In Section 2.4.1, Section 2.4.2 and Section 2.4.3, we present StemP results and compare with existing methods. To evaluate the prediction, we consider True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) base pair matching. We use Specificity/ Positivity Prediction Value (PPV), Sensititity, the Matthews Correlation Coefficient (MCC) [47, 76, 69], and F_1-score. For each predicted matched base pair, if it exists in the true folding structure, then it counts towards TP, otherwise, it counts towards FP. For each matched base pair in the true folding structure that doesn't exit in the prediction, it counts towards FN. $TP + FP$ gives the total number of base pairs in the prediction. $TP + FN$ gives the total number of base pairs in the true folding. Both MCC and F_1 takes values in between [0,1] and 1 represents 100 % matching without any additional nor missing basepair matching. Both MCC and F_1-score give an overall justification of the prediction with respect to the False prediction.

In practice, StemP prediction of the maximum matching maximal clique may not be unique. We use Standard Competition Ranking (SCR)("1224" ranking) to present the results. If there are multiple ranked 1 folding, we report with the value m in the parenthesis to indicate that there are multiple structures with the same maximum number of base pairs matched. For example, SCR(m) = 1(2) represents that there are total of 2 sequences in the top rank, the same maximum matching, and 3(4) represents that there are 4 sequences in the third rank.

Algorithm 1 Structure Predicting Algorithm (StemP)

Input: The sequence r of size n, the stem length threshold L, and bounds of SL : $SL_{\min}$ and $SL_{\max}$.

Step 1-1: full Stem-graph construction (Vertex Construction)

$k \leftarrow 1$
for $i = 1 : n$ **do**
  for $j = i + 3 : n$ **do**
    if IsBasePair(r_i, r_j) **then**
      $k \leftarrow k + 1$, $l \leftarrow 1$
      while $j - l > j + l$ and IsBasePair(r_{i+l}, r_{j-l}) **do**
        $l = l + 1$
      end while
      $l \leftarrow l - 1$,$d \leftarrow j - i$, $SL \leftarrow \frac{d}{l}$
      if $l > L$ and $SL_{\min} \leq SL \leq SL_{\max}$ **then**
        $v_k \leftarrow (i, j, l, d, SL)$,$k \leftarrow k + 1$
      end if
    end if
  end for
end for

Step 1-2: full Stem-graph construction (Edge Construction)

for $m = 1 : length(V)$ **do**
  for $n = m + 1 : length(V)$ **do**
    if (i) $j_m < i_n$ or (ii) $j_n < i_m$ or (iii) $i_m + l_m - 1 < i_n$ and $j_n < j_m - l_m + 1$ or (iv) $i_m + l_m - 1 < i_n$ and $i_n + l_n - 1 < j_m - l_m + 1$ and $j_m < j_n - l_n + 1$ **then**
      $e_{mn} = 1$
    end if
  end for
end for

Step 2: Choose a Subgraph

Find all the cliques, using [71].
Compute the total matching energy for each cliques.
Choose the maximum matching and/or maximal clique as the folding prediction.

Algorithm 2 tRNA algorithm

Input: (i)tRNA Sequence r of size $\hat{l}$, (ii)lower bound $SL_{\min,Acceptor}$ of SL for Acceptor stem, (iii)the stem length threshold L, (iv)bounds of SL : $SL_{\min}$ and $SL_{\max}$, and (v) bounds of distance $d_{\min}, d_{\max}$.

Ensure: secondary structure of tRNA

Step 1: vertex construction

Find all possible vertices v_k of $l_k \geq L$ and store them in V_{temp}

V $\leftarrow \phi$

for $v_k \in V_{temp}$ **do**

  if $d_k > \hat{l}/2$ **then**

    $SL_k \leftarrow (\hat{l} - d_k + 2l_k - 2)/l_k$

    if $SL_k \leq SL_{\min,Acceptor}$ **then**

      $V \leftarrow V \cup v_k$

    end if

  else

    $SL_k \leftarrow d_k/l_k$

    while $SL_k \leq SL_{\min}$ **do**

      $l_i \leftarrow l_k - 1, SL \leftarrow d_k/l_k$

    end while

    if $SL_{\min} \leq SL \leq SL_{max}andl_k \geq L$ and $d_{\min} \leq d_k \leq d_{\max}$ **then**

      $V \leftarrow V \cup v_k$

    end if

  end if

end for

Step 2: go to step 2 of Algorithm 1.

Algorithm 3 5S rRNA algorithm for Archaeal

Input: (i) five sets of possibles vertex in each Helix: H_1, H_2, H_3, H_4, H_5, (ii) lower/upper bounds of *Generalized-Stem-Loop score* in α domain and β domain: $GSL^{\alpha}_{\min}, GSL^{\alpha}_{\max}, GSL^{\beta}_{\min}, GSL^{\beta}_{\max}$.
Output: secondary structure of 5S rRNA
$V \leftarrow \phi, V = V \cup H_1$
for $v_m \in H_2$ **do**
  for $v_n \in H_4$ **do**
    if ExistEdge(v_m, v_n) and $GSL^{\alpha}_{\min} \leq GSL(v_m, v_n) \leq GSL^{\alpha}_{\max}$ **then**
      $V \leftarrow V \cup (\min(i_m, i_n), \max(j_m, j_n), l_m + l_n, d_m, GSL(v_m, v_n))$
    end if
  end for
end for
for $v_m \in H_3$ **do**
  for $v_n \in H_5$ **do**
    if ExistEdge(v_m, v_n) and $GSL^{\beta}_{\min} \leq GSL(v_m, v_n) \leq GSL^{\beta}_{\max}$ **then**
      $V \leftarrow V \cup (\min(i_m, i_n), \max(j_m, j_n), l_m + l_n, d_m, GSL(v_m, v_n))$
    end if
  end for
end for
Step2: go to step 2 of Algorithm 1.

Remark: ExistEdge can be found in Algorithm 1 Step 1-2. each vertex v in V has 5 attributes: (i, j, l, d, SL)

Algorithm 4 algorithm to construct $l_{1[n_1/n_2]}l_2$ vertex in 5S rRNA

Require: (i)5S rRNA Sequence r of size n, (ii)bound of SL: $SL_{\min}, SL_{max}$, (iii) subvertex size: l_1, l_2, and (iv) gap size n_1, n_2

Ensure: All possible vertex of structure $l_{1[n_1/n_2]}l_2$ with stem-loop score SL satisfying $SL_{\min} \leq SL \leq SL_{\max}$

```
l ← 0, V ← φ, k ← 0
for i = 1 : n do
  for j = i + 2 : n do
    d ← j − i
    idx ← 0, l ← 0
    while IsBasePair(r_i, r_i) and i < j and l < l_1 + l_2 do
      index ← idx+1, l ← l + 1
      if idx ≠ 1_1 then
        i ← i + 1, j ← j − 1
      else
        i ← i + 1 + n_1, j ← j − 1 − n_2
      end if
      if i > n or j ≤ i then
        break
      end if
    end while
    LS ← d/l
    if s_min ≤ SL ≤ s_max then
      k ← k + 1, v_k ← (i, j, l, d, SL), V ← V ∪ v_k
    end if
  end for
end for
```

Table 2.1: StemP parameters for short RNA sequences (length up to 50) from Protein Data Bank[77]. Wobble pair can be considered in addition.

StemP parameters for short RNA sequence	
Base-pair matching	Canonical base pair (Wobble pair)
Minimum Stem Length	$L = 3$ (or 2)
Stem-Loop score	$2 \leq SL \leq 20$ (optional)
Optimal Structure	Maximum base-pair matching (often)

2.4.1 Short RNA sequences

We first experiment with data from the Protein Data Bank (PDB) [77], which preserves structure information of a large number of biological molecules including proteins and nucleic acids. There are various experimental work on structure prediction of sequences from PDB. In [76], Nucleotide Cyclic Motif (NCM) is introduced to represent nucleotide relationships in structured RNA. Experiments were performed on 182 sequences from PDB of sizes from 8 to 35 and the corresponding result reached 0.87 of MCC on average. From the point of view of graph structure, base triples were explored in [78]. Related work includes Direct-Coupling Analysis[79], orientation and twisting of β-sheets [80] and multiple threading alignment approaches [81].

Parameters: We experiment with RNA sequences of length up to 50, and provide the accuracy of our results based on the folding structures formed by Parallel Watson-Crick/Watson-Crick (tww) and Anti-Parallel Watson-Crick/Watson-Crick (cww) based pairs retrieved from Nucleic Acid Database [82, 83]. Table 2.1 shows the parameters we chose for StemP. We set the minimum stem length to be $L = 3$, and set Stem-Loop score to be $2 \leq SL \leq 20$. These two mild conditions enhance the computing speed by reducing the number of vertices in the full Stem-graph. For example, for sequence 1KXK, MCC 0.96 is obtained in 11 seconds with $2 \leq SL \leq 20$, while it took 89 seconds to obtain the same accuracy without the conditions. We found that for a short sequence, the correct structure may be a single vertex, when there is no clique of size bigger than 2.

Table 2.2: StemP for short RNA sequences from Protein Data Bank (length up to 50). In the first column, superscript p indicates pseudo knots, superscript w indicates we allowed wobble base pairs. Superscript L indicates using $L = 2$, otherwise we set $L = 3$, and S indicates when SL is used. The second column shows the best MCC value among all maximal clique, the third column shows Standard Competition Ranking (SCR) of this best MCC in the form of SCR(m), and the forth column shows the CPU time in milliseconds (ms). 293^* denotes SCR(m)= 293(512) for 2OZB.

RNA	StemP	SCR	CPU	[84]	[85]	[46]	[76]	[86]
1RNG ^{w}	**1.00**	1 (1)	61	**1.00**	**1.00**	**1.00**		
2F8K	**0.91**	1 (1)	19	**0.91**	0.82	**0.91**	o	x
2KVN	**1.00**	1 (1)	6	**1.00**	**1.00**	0.91		
2AB4	**1.00**^{S}	1 (1)	4	**1.00**	**1.00**	0.93	o	x
361D	**1.00**	1 (1)	12	0.79	0.79	0.79	o	o
2ANN	**1.00**	4 (4)	12	0.65	0.71	0.77	x	o
1RLG	**0.91**^{L}	1 (3)	244	0.79	0.79	0.79	x	o
2QUX	**1.00**	1 (1)	25	**1.00**	**1.00**	**1.00**	o	x
387D	**0.77**	4 (3)	7	0	0	0.42	x	x
2L5Z ^{w}	**0.95**	1 (1)	41	**0.95**	**0.95**	**0.95**		
1MSY ^{w}	**0.91**^{S}	4 (1)	30	0.77	0.77	0.83	x	o
1L2X ^{p}	**0.94**^{S}	1 (1)	29	0.79	0.79	0.72	x	o
2AP5 ^{p}	**1.00**^{S}	1 (1)	13	0.79	0.79	0.79	x	x
1JID ^{w}	**0.80**	1 (2)	34	**0.80**	**0.80**	**0.80**	o	x
1OOA ^{w}	**1.00**	3 (3)	31	**1.00**	**1.00**	0.87	x	o
430D	**0.83**	1 (3)	6	**0.83**	**0.83**	**0.83**	x	o
2OZB ^{w}	**1.00**^{L}	293^*	1k	**1.00**	0.95	0.89	o	o
1MJI	**0.95**	1 (1)	7	**0.95**	**0.95**	**0.95**	x	x
1ET4 ^{p}	**0.47**	6 (1)	17	0.13	0.13	0.15	x	o
2HW8 ^{w}	0.96	9 (11)	97	**1.00**	**1.00**	0.89	o	o
1I6U ^{w}	**0.87**	3 (13)	150	**0.87**	**0.87**	**0.87**	o	o
1F1T	**0.88**^{L}	2 (3)	101	**0.88**	**0.88**	0.73	o	o
1ZHO	**1.00**	2 (4)	23	**1.00**	**1.00**	0.90	o	o
5NZ3 ^{p}	**0.82**	8 (4)	24	0.55	0.55	0.68		
1SO3	**1.00**^{L}	1 (18)	3k	0.89	0.89	0.92	o	o
1XJR ^{w}	**0.94**^{L}	6 (34)	27k	**0.94**	0.90	0.79	o	o
1U63	**0.97**	2 (3)	153	**0.97**	**0.97**	**0.97**	o	x
2PXB ^{w}	**0.97**	12(42)	389	**0.97**	**0.97**	**0.97**	o	o

StemP Results and Comparison for short sequences: Table 2.2 presents prediction results for sequences of length up to 50. The comparisons are presented between the proposed method StemP, FOLD[84], MaxExpert[85], ProbKnot[46], MC [76] and NAST [86] based on MCC value. The experimental results of those methods were performed on RNAstructure web server[87] with default parameters[1]. MC [76] and NAST [86] results are from [88], where O and X indicate success or failure of these methods, and if empty, experiments were not known.

Some variations are presented as superscripts in Table 2.2. The superscript p indicates existence of pseudo knots. For StemP, pseudo knots are naturally predicted without any a priori information, but for some methods it is important to indicate. The superscript w indicates that we allow wobble base `G-U` pairs in StempP. If a structure is known to have a Wobble base pair, it helps to add this possibility in vertex construction to identify the true folding. Typically short RNA sequences don't have a strong structure known a priori, that using Stem-Loop score is not necessary. We indicated with superscript S to denote that Stem-Loop score are imposed. For other sequences, the same results are obtain with or without SL condition.

StemP is computationally efficient. In Table 2.2 forth column, we present the CPU time in milliseconds (ms) for predicting structure with StemP for each sequence. The average CPU time in Table 2.2 is 1.18 seconds. We used MATLAB with Intel®Core i5-9600K processor with 3.7GHz 6 Core CPU and 16 GB of RAM.

In Table 2.2, StemP results in second column show the best MCC values among all maximal cliques. These best MCC values show that they are more accurate compared to other methods for all sequences except for one 2HW8. The SCR(m) in the third column shows, 10 out of 28 results are SCR(m)=1(1), that the unique top choice (maximum matching) is the structure prediction, and the matching accuracy is of MCC 1 or above 0.91 (i.e.

[1]Temperature = 310.15(K), Maximum Loop Size = 30, Maximum % Energy Difference = 10, Maximum Number of Structures = 20, Window Size = 3, Gamma = 1, Iterations = 1, Minimum Helix Length = 3, SHAPE Intercept = -0.6, SHAPE Slope = 1.8, Maximum probabilities to show = 2.

(a) MCC 0.98 (b) MCC 0.92 (c) MCC 0.95

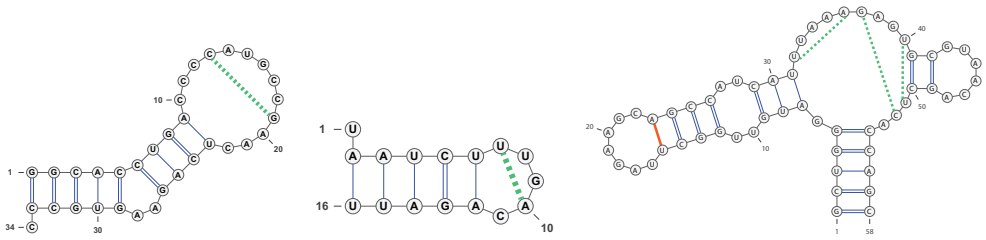


Figure 2.2: Examples of StemP of MCC above 0.9. Missing pairings are non-canonical or a single canonical pairing. (a) StemP of 1MJI (length 34). (b) StemP of 2F8K (length 16). (c) StemP of 1MMS (length 58). The green lines is missing base-pair matching (False Negative), which are either non-canonical pairing or a single canonical pairing. (VARNA[89] is adopted for visualization of secondary structures.)

100% or above 91% matching). Another 4 is the top choice, but with a multiple possibilities. We list the rank 1, MCC values for the case, when the best structure prediction is not ranked 1: in Table 2.2, 2ANN 12ms, 387D 7ms, 1MSY 30ms , 1OOA 31ms, 2OZB 1s, IET4 17ms, 2HW8 97ms, 1I6U 153ms, 1F1T 101ms, 1ZHO 23ms, 5NZ3 24ms, 1XJR 27s, 1U63 153ms, and 2PXB 389ms.

Figure 2.2 shows examples when the best matching is not MCC 1 (100 % matching). Typical examples are shown for MCC 0.95, 0.91 and 0.85. These mismatches are causes by (i) non-canonical basepair matching such as `G-G`, `A-G`, `C-U`, or (ii) a singe canonical base pair with length 1. In our current setting, vertex only considers the stem of length $L \geq 2$.

For the cases when the best MCC is not ranked 1, we give an example in Figure 2.3 with 1MSY (length 27). The best prediction result is found in SCR=4. The true structure is similar to (b) SCR=2(2) and (d) SCR=4, with one more basepair in (b), yet the true structure is closer to (d). This example shows, although there are similar folding structures with more matching, the true structure does not connect all the basepair matching. StemP with maximum matching ranking, it found (a) as a top choice.

(a) $SCR = 1$ (b) $SCR = 2(2)$ (c) $SCR = 2(2)$ (d) $SCR = 4$

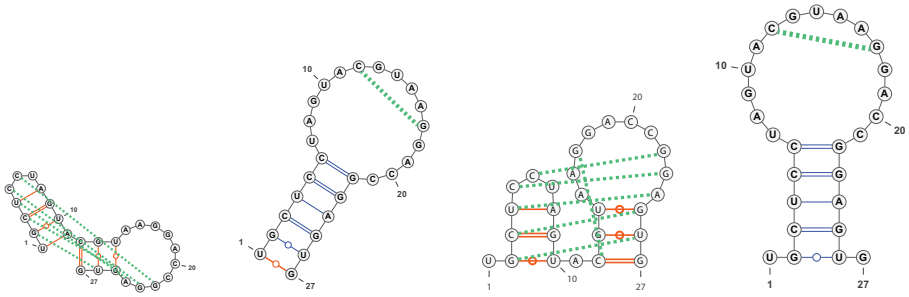


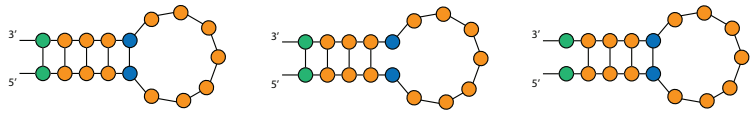
Figure 2.3: For 1MSY (length 27), the true structure is not maximum matching. The true folding is closest to StemP result SCR=4 shown in (d) with MCC=0.91. (a) is StemP result rank 1 with MCC 0, (b) SCR(m)=2(2) with MCC 0.83, and (c) another SCR(m)=2(2) with MCC 0. The green lines show the missing base pairs, and the red line wrong matching. One base pair `C(12) - G(16)` is missing in (d), and (b) also shows similar result.

2.4.2 tRNA sequence

The folding structure of tRNA is mostly standard as shown in Figure 2.5(b). There are 4 distinct regions, which are Acceptor, D loop, Anticodon loop and TC loop. The structure of transfer RNA (tRNA) was determined through comparative analyses of RNA structure, and various methods are developed [90]. Different types of RNA including tRNA were studied in [91] regarding secondary structure and tertiary structure. It was shown that the secondary structure is more significant and stable than the latter one. MC-fold [76] is a classical model to unify all basepair matching energetic contributions. Minimum Free Energy model was adopted by [92], [40] and [93] to discover tRNA sequences, where probing data is utilized in [40]. In [69], a possible RNA alignment sequence was represented by finding the most frequent stem patterns from a database, and experiments were performed on a variety of RNA sequences including tRNA. From the geometric point of view, the author of [68] employs the idea of tree graphs and dual graphs to represent RNA tree motifs and general RNA motifs which makes it possible to characterize the typologies of RNA structure. There are probability based methods[46, 94] which measures the probability of the structure of a nucleic or base pair based on a large learning group. Some methods provide multiple sequences alignments with probabilities and can be extended to different types of RNA

Table 2.3: StemP parameters for tRNA. For Cysteine, Glutamic Acid, Glutamine, Histidine, SL upper bound 4.7 is used. For Alanine, Asparagine, Aspartic Acid, Glutamic Acid, Glutamine, Glycine, Histidine, Isoleucine, Lysine, Methionine, Phenylalanine, Proline, Tryptophan, Tyrosine, SL upper bound 5.4 is used. $\hat{l}$ is the total length of the sequence.

StemP parameters for tRNA folding	
Basepair matching	Canonical and Wobble matching
Stem type	Partial Stems included
Minimum Stem Length	$L = 3$
Stem-Loop Distance	$12 \leq d_i \leq 18$
Stem-Loop score	$3 < SL \leq 4.7$ or 5.4
Stem-Loop Distance (Acceptor)	$\hat{l}/2 < d_i$
Acceptor-Stem-Loop score	$ASL \leq 3$
Optimal Structure	Maximum matching Maximal clique



(a) (b) (c)

Figure 2.4: Partial stem considered for tRNA folding. (a) and (c) are typical vertex constructed. (b) a new type of partial stem considered for tRNA in addition. Notice one interior basepair is not connected.

sequences.

Parameters: For StemP for tRNA, (i) Table 2.3 shows all the StemP parameter used for the tRNA folding structure prediction. To account for the standard shape of tRNA, we add following modifications: (ii) We use Acceptor-Stem-Loop score in (Equation (2.5)) to find the Acceptor stem, and consider (iii) Partial Stem. For tRNA, it is common that some of matching base pairs at the end of the stem do not pair. We consider these Partial Stem, which is a sub-stem of a typical vertex. Figure 2.4 (b) shows Partial stem considered in addition.

StemP Results for tRNA: We present an outline of typical result of StemP for tRNA in Figure 2.5 (tRNA Accession Number AB041850 from organism Alanine). Figure 2.5

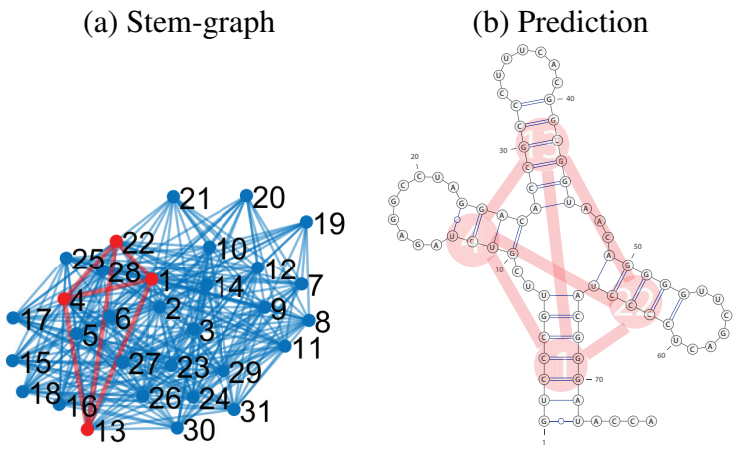


Figure 2.5: StemP for tRNA. (a) The full Stem-graph of sequence with Accession Number AB041850. (b) The maximum matching maximal clique (SCR=1) predicted by StemP, superposed with the folding structure. In this example, StemP reaches 100% correct matching (MCC = 1.00).

(a) shows the full Stem-graph and the red subgraph represents a maximal clique predicted by StemP. (b) shows the maximal clique superposed with the folding structure. In this example, StemP gives 100% matching (MCC=1).

In Figure 2.6, we present results of StemP for tRNA structure prediction for 27,010 different tRNA sequences containing 15 subset of tRNA from The Gutell Lab [95]. True folding structures are taken from the Gutell Lab. We consider sequences that satisfies the following two criteria as valid inputs: (i) The length larger than or equal to 50; (ii) There exists at least one base pair in the true folding structure. The typical number of vertices in full Stem-graph for tRNA ranged from 20 to 40, edges from 120 to 150, and cliques from 180 to 2000. In (a), we show the percentage of SCR in $[1, 1], (1, 5], (5, 10], (10, 15], (15, \infty)$ for the best MCC prediction. In (b), we show the percentage of MCC values for the top SCR=1 prediction. In (c), we show the percentage of the best prediction score (MCC) in $[0.95, 1], [0.90, 0.05), [0.85, 0.90), [0.80, 0.85), [0, 0.8)$ for each organism respectively.

StemP finds the shape of the tRNA structure well with a short computation time, in general in less than 0.3 seconds. Figure 2.6 shows that the majority of the sequences, 20,463 among 27,010 (75.8%), reach the maximum MCC as the top ranking (SCR=1)

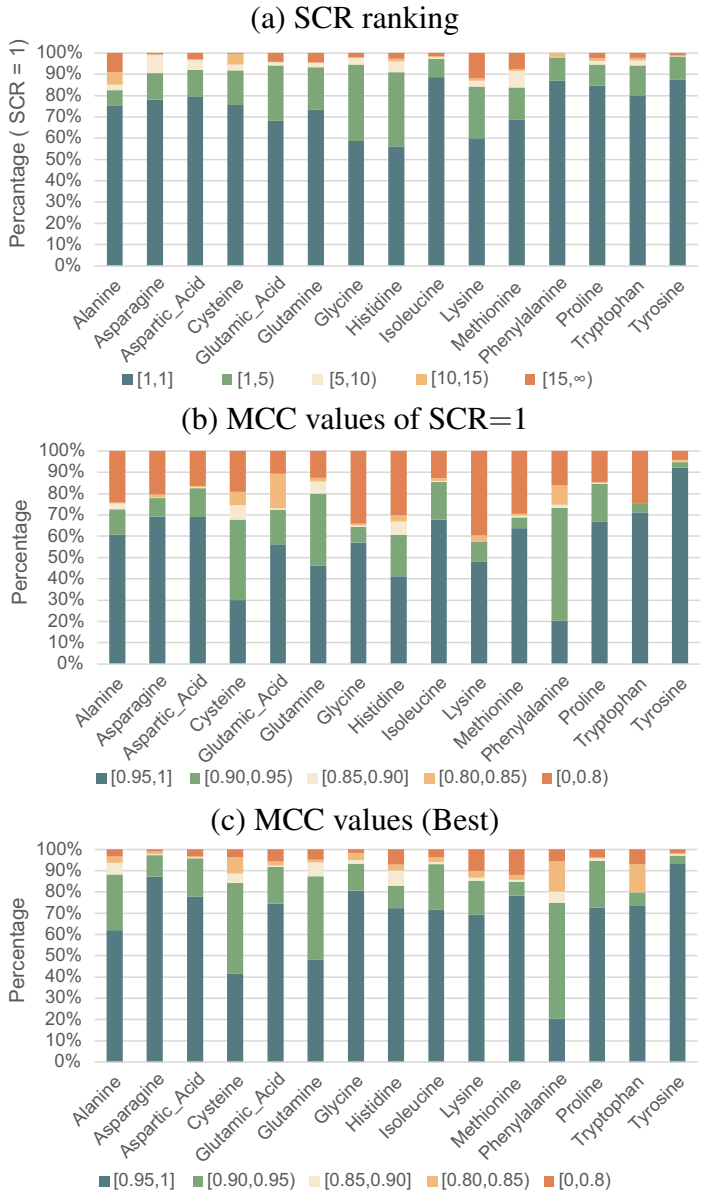


Figure 2.6: StemP result for 27,010 different tRNA sequences. (a) The percentage of sequences that have SCR in $[1, 1]$, $(1, 5]$,$(5, 10]$,$(10, 15]$,$(15, \infty)$ for the highest MCC prediction. (b) The percentage of sequences with SCR = 1 that have the prediction score (MCC) to be in $[0.95, 1]$, $[0.90, 0.05)$, $[0.85, 0.90)$,$[0.80, 0.85)$,$[0, 0.8)$ in each organism. (c) The percentage of sequences that have the best prediction score (MCC) to be in $[0.95, 1]$, $[0.90, 0.05)$, $[0.85, 0.90)$,$[0.80, 0.85)$,$[0, 0.8)$ for each organism.

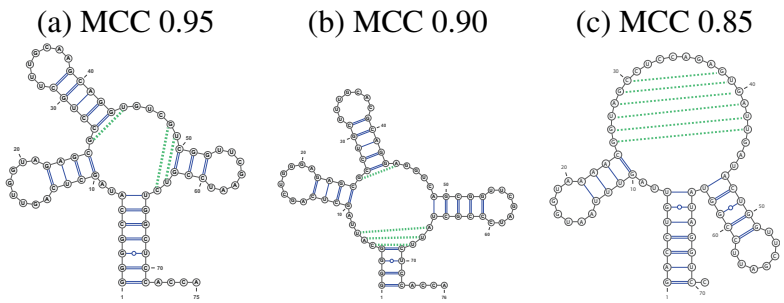
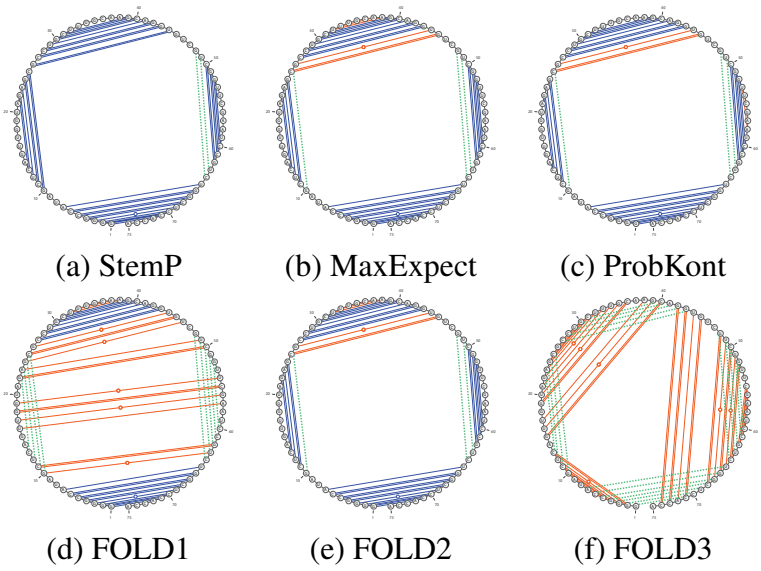


Figure 2.7: Examples of StemP results with MCC above 0.85 tRNA. (a) StemP (rank 1) and true structure of (L00194) superposed. Green dotted line shows the missing pairs (False Negatives), while all other base-pairs matching are correct. (b) StemP, SCR=1(9), and true structure of (AF146727.1) superposed. (c) AY653733. StemP, SCR = 7(3), with true folding.

using StemP, i.e., maximum base-pair matching. The second graph shows that for SCR=1, the majority of the sequences (74.8%), 20,202 among 27,010, give MCC above 0.9. The majority of the sequences (89.1%), 24,053 among 27,010, reaches MCC value in $[0.90, 1]$ in the best prediction, and 24,398 (90.3%) gave the maximum MCC within top 5 SCR.

When MCC is around 0.95, the prediction may miss base-pair matching, typically non-canonical pair like `U-U`, `G-A` or `C-U`. Figure 2.7 shows example of typical structure with MCC 0.95, MCC 0.90 and MCC 0.85. (a) with Accession Number L00194 and MCC 0.95, (b) with Accession Number AF146727.1 and MCC 0.90. (c) with Accession Number AY653733 and MCC 0.85. In (a), the mismatch (False Positive) is non-canonical pair `U(49)-U(63)`. Similarly, in (b), the mismatches are non-canonical pair `G(26)-A(44)` and `C(5)-U(68)`. In (c), the missing stem starting with `G(25)-U(43)` and ending with `C(30)-G(38)` is broken into two sub-stems of length 3 and 2 due to the non-canonical pair `A(28)-G(40)`, which fall out of the Stem-Loop score range and the stem length range. It is shown that these miss-match does not affect the general shape of the folding structure.

Comparison results for tRNA: Figure 2.8 shows comparison results using StemP, FOLD, MaxExpect, and Probknot for a tRNA sequence with Accession Number L00194. In Figure 2.8, (a) StemP has only 2 base pairs missing (False Negative) and no extra match-



(a) StemP (b) MaxExpect (c) ProbKont

(d) FOLD1 (e) FOLD2 (f) FOLD3

Figure 2.8: tRNA (Accession Number L00194) prediction comparison. False-Positive (Green), and False-Negative (Red). (a) StemP result of SCR=1(1) (MCC 0.95) (b) MaxExpect (MCC 0.86), (c) ProbKont (MCC 0.84), and (d)-(f) 3 possible FOLD predictions (MCC 0 - 0.86).

ing (False Positive). (b) MaxExpect and (c) ProbKont have only acceptor stem correctly identified, and in (c), there are 3 base pairs missing (False Negative) and 4 extra pairs (False Positive). (d)-(f) are multiple structures provided by FOLD, there are some base pairs missing (False Negative), and extra pairs (False Positive).

In Table 2.4, we present comparisons on 47 different tRNA sequences between StemP and [96]. We show the highest F_1 among the predictions with SCR=1 by F_1^{top}, the highest F_1 among all possible clique structure predictions by F_1^{best}. StemP has higher F_1^{top} and F_1^{best} for 46 sequences out of 47. There are 40 sequences reach the highest F_1-score when SCR = 1. StemP has 0.95 as an average for best prediction and 0.92 for top prediction on this test. For StemP, Table 2.3 parameters are used with $3 \leq SL \leq 5.4$ uniformly for all testing sequences.

	[96]	StemP			[96]	StemP	
Accn	F_1	F_1^{top}	F_1^{best}	Accn	F_1	F_1^{top}	F_1^{best}
AY934184	0.63	0.98	0.98	AE014184	0.73	0.98	0.98

AE013169	0.59	0.98	0.98	CP000473	0.62	0.98	0.98
AY934018	0.60	0.73	0.75	CP000492	0.58	0.77	0.98
AJ010592	0.47	0.98	0.98	CR382129	0.55	0.68	0.73
U41549	0.58	0.93	0.93	X03154	0.56	0.98	0.98
AE000520	0.70	0.57	0.78	CP000254	0.55	0.85	0.85
CP000143	0.68	0.98	0.98	CP000493	0.73	0.93	0.93
AY934254	0.67	0.98	0.98	BA000021	0.50	0.98	0.98
AE009952	0.54	0.93	0.93	CP000412	0.51	0.79	0.95
AE017159	0.48	1.00	1.00	AP006618	0.56	0.98	0.98
AY934351	0.60	0.98	0.98	CP000099	0.56	0.93	0.93
BA000023	0.70	0.95	0.95	CP000141	0.68	0.98	0.98
AY934387	0.55	0.98	0.98	AC006340	0.62	0.93	0.93
AY933864	0.53	0.98	0.98	BX569691	0.63	0.98	0.98
AJ248288	0.68	0.95	0.95	AF137379	0.51	0.98	0.98
CP000471	0.64	0.98	0.98	DQ396875	0.59	0.98	0.98
BA000011	0.64	0.98	0.98	CP000142	0.69	1.00	1.00
BX321863	0.61	0.98	0.98	BX640433	0.58	0.73	0.98
CP000423	0.55	0.55	0.98	AJ294725	0.43	0.98	0.98
CR936257	0.65	0.93	0.93	X04779	0.57	0.98	0.98
AC004932	0.53	0.93	0.93	AY934393	0.49	0.98	0.98
AY933788	0.58	0.98	0.98	AE008623	0.61	0.98	0.98
X04465	0.41	0.93	0.93	AE000657	0.73	0.98	0.98
DQ093144	0.62	0.98	0.98				
				Average	0.59	**0.92**	**0.95**

Table 2.4: Comparison of tRNA prediction between StemP and [96]. The 47 sequences in [96] from Gutell Lab [95]. Accn denotes the Accession number of the corresponding sequences. F_1^{top} is the best F_1-score among predictions with SCR = 1. F_1^{best} is the highest F_1-score of among all predictions of clique.

2.4.3 5s rRNA sequences

5S rRNA plays a critical role in ribosomes of organisms. The sequences typically contains about 120 nucleotides with a particular structure [97]. This structure has been recognized by comparative sequence analysis. The size and ubiquity of 5S rRNA, that enabled RNA sequencing using direct methods, made it an ideal candidate for a molecular phylogenetic marker[97]. Similar to tRNA, different methods such as Minimum Free Energy[92], constructing tree structure[68] and sequence analysis[98, 76] are available for predicting 5s rRNA structures. TurboFold II [94] is an extension of TurboFold [99], a comparative method that provide multiple sequence alignments by iteratively estimating the probabilities for nucleotide positions between all pairs of input sequences. A modified version of TurboFold II in [100] uses basepair probabilities from SHAPE experimental data. In [46], 309 sequences were tested by a probability based method ProbKnot where 69.2% of known pairs were correctly predicted in average.

5S rRNA has a particular structure, consisting of 5 stems (I-V) and 5 loops (A-E). We use the notation of Domain α, β, γ to help illustrate the structure of 5s rRNA. Domain α is identical to Helix I. Domain β and Domain γ can be understood as a structure starting from a base stem leading up to one hairpin loop, including all internal loop and bulge in between. Domain β has Helix II enclosing Helix IV while Domain γ has Helix III enclosing Helix V. Figure 2.10 (b) shows a typical example of full Stem-graph of a 5s rRNA sequence. We explore 5s rRNA sequences from the Gutell Lab [95] and the true foldings given in it.

Parameters: In 5S rRNA, often helix doesn't have consecutive basepair matches, but has one or two bases gap. We consider the stem variations to account for such cases, which counts a combination of shorter stems as one vertex. The structure $l_{1[n_1/n_2]}l_2$ or $l_{1[n_1/n_2]}l_{2[n_3/n_4]}l_3$ denotes such variation, where $l_i (i = 1, 2)$ the length of consecutive base pairs, and n_is denote the gaps between the shorter stems, as illustrated in Figure 2.9. Notice that, not considering the gaps, this vertex have stem length to be $l_1 + l_2$. We use Stem-

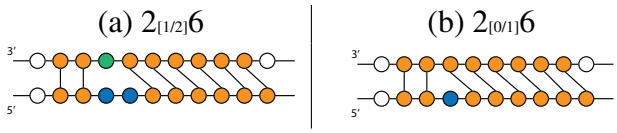


Figure 2.9: Two examples of 5s rRNA vertex variation. Both has stem length $L = 8$, but with gaps. We consider such cases as one vertex for 5S rRNA structure prediction.

Loop score (Equation (2.3)) to identify each Helix (I-V) and Generalized-Stem-Loop score (Equation (2.4)) to identify the Domain β, γ which contain more than one Helix.

For the computation, we add a step to use GSL for more efficient computation. From the input data,

1. find five different sets of vertices v_i with appropriate SL score (each set of candidates for Helix I-V).

2. Use GSL to find two (additional) sets of vertices V_k to find Domain β and γ.

3. Construct edges between each domain.

4. A maximal clique that represents the highest energy gives the prediction.

We summarize the StemP parameters for 5S rRNA in Table 2.6 for Archaeal, and general parameters in Table 2.8.

StemP Result for 5s rRNA: Figure 2.10 shows a typical result of StemP for a 5s rRNA sequence (Accession number AE000782). StemP result in (a) gives MCC 0.97 accuracy with only 2 non-canonical base pairs `C(28)-U(56)` and `A(81)-G(103)` matching missing. Not considering non-canonical pairs as positive pairs, StemP gives MCC 1 for this sequence.

In Table 2.5, StemP is tested on 53 sequences in organism Archaeal based on the refined parameters in Table 2.6. For some sequences such as AE010349, X07545, the cpu time is longer than 1 minute because of the number of allowed possible structures is very large. The average cpu time for each sequence is 15.5 seconds. Table 2.5 (a) SCR shows that

(a) Prediction (MCC 0.97) (b) True folding

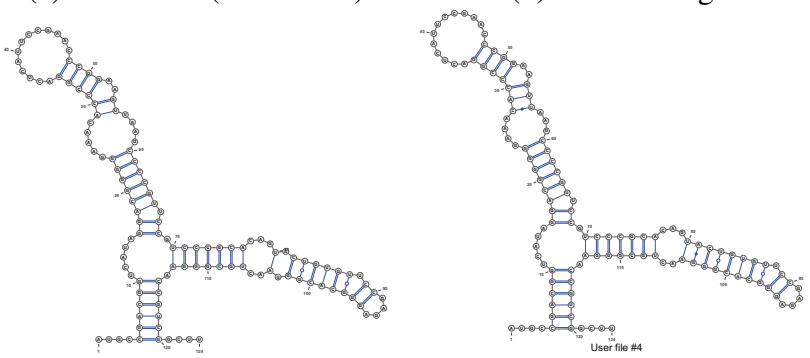


Figure 2.10: StemP for 5S rRNA (Accession number AE000782). (a) StemP which has MCC 0.97 (not considering non-canonical pairs, MCC=1). (b) True folding. There are 154 vertices, and 8986 cliques. The best predicted is SCR=1(16). The average MCC for these 16 structures is 0.89 (not considering non-canonical pairs).

79.2% of the sequences have rank 1, and 90.6% of the sequences have rank 5 or higher. For 5S rRNA, maximum matching and maximal clique seems to be a good choice for the prediction. Table 2.5 (b) shows that 90.6% sequences have top MCC to be more than 0.92. (c) shows that 66% of the sequences have MCC higher than 0.95 and 100% of the tested sequences has MCC higher than 0.92. (d) shows the top and the best MCC values for each Archaeal 5s rRNA sequences.

(a) SCR ranking of best MCC

Organism	#	$=1$	≤ 3	≤ 5	>5
Archaeal	53	42 (79.2)	+2 (83.0)	+4 (90.6)	5 (9.4)

(b) Top MCC values of StemP

Organism	#	≥ 0.97	≥ 0.95	≥ 0.92	<0.92
Archaeal	53	16 (30.2)	+9(50.9)	+ 21 (90.6)	5 (9.4)

(c) Best MCC values of StemP

Organism	#	≥ 0.97	≥ 0.95	≥ 0.92	<0.92
Archaeal	53	16 (30.2)	+19(66.0)	+ 18 (100.0)	0 (0.0)

(d) Prediction of StemP on 5s rRNA sequences

Accn	Top	Best	Accn	Top	Best
AE000782	0.97	0.97	X62859	0.78	0.95
AE006649	0.97	0.97	U67537	0.96	0.96
AE010349	0.89	0.96	U67518	0.96	0.96
AM180088_b	0.95	0.95	M34911	0.94	0.94
AP000006	0.95	0.96	X62860	0.96	0.96
AP000986	0.96	0.96	X62861	0.95	0.95
AP006878	0.97	0.97	M34910	0.97	0.97
Arc.fulgidus	0.97	0.97	X62862	0.97	0.97
BA000001_b	0.95	0.96	X62864	0.44	0.94
BA000002	0.83	0.97	M26976	0.97	0.97
BA000023_b	0.96	0.96	X15364	0.96	0.96
CNSPAX02	0.82	0.96	X72495	0.96	0.96
CNSPAX03	0.94	0.95	M21086	0.99	0.99
CP000254_b	0.95	0.95	X15329	0.96	0.96
CP000477_b	0.95	0.95	V01286	0.97	0.97
CP000493	0.99	0.99	U05019	0.96	0.96
CP000575	0.97	0.97	X01588	0.97	0.97
CR937011	0.95	0.95	Y08257	0.97	0.97
DQ314493_b	0.95	0.95	X05870	0.97	0.97
DQ314494	0.95	0.95	X07692	0.95	0.96
X07545	0.97	0.97	M12711	0.96	0.96
E.coli.ref	0.93	0.96	X02709	0.95	0.95
AF034620	0.95	0.95	M32297	0.95	0.95

L27343	0.97	0.97	AL445066	0.95	0.95
L27169	0.95	0.95	NC_002689	0.95	0.95
L27236	0.96	0.96	BA000011	0.95	0.95
X15364	0.93	0.95			
			Average	0.94	0.96

Table 2.5: StemP for 53 different 5S rRNA sequences. (a) The number and the percentage in parenthesis for SCR $\leq 1, 3, 5$ or > 5. The true structure is mostly within top 5 ranking. (b) The number of the StemP results (with SCR = 1) of MCC $\geq 0.97, 0.95,$ or 0.92. (c) The number of the best StemP results of MCC $\geq 0.97, 0.95, 0.92$ and < 0.92. (d) StemP results on 53 different Archaeal 5s rRNA sequences. Accn denotes the Accession number of each sequences. Top represents the highest MCC score among all predictions that has SCR = 1. Best represents the highest MCC score among all predictions with clique structure.

Archaeal	Stem Length Variation	Stem Loop score
Helix I*	6, 5, $4_{[1/0]}1$, 4	$17.82 \leq SL \leq 27.5$
Helix II	8, $2_{[0/1]}6$, $2_{[0/1]}5$, $2_{[1/2]}5$, $1_{[1/2]}6$, $2_{[0/1]}1_{[2/1]}4$	$6.37 \leq SL \leq 7.72$
Helix III	$3_{[0/2]}4$, $2_{[0/2]}4$, $2_{[2/4]}2$	$5.66 \leq SL \leq 10.76$
Helix IV	7, 6, 5, $3_{[1/1]}2$, $3_{[2/2]}1$, $2_{[1/1]}2$	$3.9 \leq SL \leq 6.6$
Helix V	8, $1_{[1/1]}5_{[2/1]}2$, $1_{[1/1]}6_{[1/0]}2$, $8_{[1/0]}2$, $1_{[1/1]}5$, $1_{[1/1]}8$, $1_{[1/1]}7$	$2.24 \leq SL \leq 3.1$
Domain β		$3.46 \leq GSL \leq 4.26$
Domain γ		$2.52 \leq GSL \leq 3.43$

Table 2.6: StemP parameters for 5S rRNA Archaeal. Canonical and Wobble basepair matching is considered, and Partial Stems are included. *: Helix I is Domain α.

Comparison on 5s rRNA: Figure 2.11 shows comparison on sequences AE000782 using StemP, MaxExpect, ProbKnot and Fold. Using the parameters in Table 2.6, StemP correctly found all base pairs except for two non-canonical base pairs `C(28)-U(56)` and

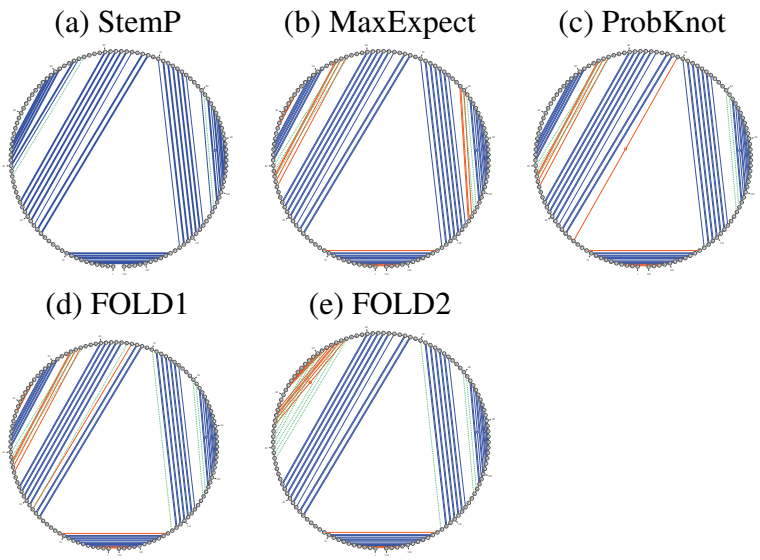


Figure 2.11: 5S rRNA(AE000782) prediction. (a) StemP (MCC 0.97), (b) MaxExpect (MCC 0.82), (c) ProbKnot (MCC 0.85), (d)-(e) FOLD (MCC 0.73 - 0.80). FalsePositive(Green), and FalseNegative (Red).

A(81)−G(103). MaxExpect, ProbKnot and FOLD all mismatched base pairs in the first branch. For Helix II, (a), (b), and (d) successfully identified the branch while (c) has one missing base pair (False negative) U(14)−G(69) and (d) has a missing base pair (False Negative) A(18)−U(65) as well as a False Positive base pair A(18)−U(66). Note that Helix IV has structure $3_{[0/2]}4$ with one non-canonical base pair C(28)−U(56), (b)-(e) all failed to identify at least 3 base pairs in Helix IV completely. For Helix III, (a),(b),(c) successfully recognized this branch while (d),(e) both lost one pair U(70)−A(113) as the first pair of the vertex. The special structure of Helix V with one non-canonical base pair A(81)−G(103) in the middle made it difficult for all methods to identify the complete structure. FOLD, MaxExpect, ProbKnot all lost two pairs: U(80)−A(104) and A(81)−G(103) and MaxExpect found two extra pairs in (b). StemP is robust in predicting structures of 5s rRNA sequences.

In Table 2.7, we present the comparison of StemP on 6 different 5s rRNA sequences with Genetic Algorithm (RNAPredict) [101], SA [102], two-level particle swarm optimization algorithm (TL-PSOfold)[103], RNAfold[104], Sarna-predict (SP) [56], Mfold[42],

Table 2.7: Comparison of 5s rRNA structure prediction between StemP, RNAPredict[101], SA [102], TL-PSOfold[103], RNAfold[104], SP[56], Mfold[42], and COIN[105]

Accn	Method	Spe %	Sen %	F1%	CPU(s)
X67579	StemP	**100.0**	**94.6**	**97.2**	**0.16**
	RNAPredict[101]	84.6	89.2	86.8	101.89
	SA [102]	89.1	89.1	89.1	401.34
	TL-PSOfold[103]	33	86.8	89.2	88.0
	RNAfold[104]	82.5	89.2	85.7	
	SP[56]	84.6	89.2	86.8	
	Mfold[42]	80.5	89.2	84.6	
	COIN[105]	97.1	89.2	93	
AF03462	StemP	**100.0**	**89.5**	**94.4**	**1.97**
	RNAPredict	90	71.1	79.4	102.66
	SA	90	71	79.4	479.08
	TL-PSOfold	81.6	86.1	83.8	
	RNAfold	81.6	86.1	83.8	
	SP	90	71.1	79.4	
	Mfold	85.3	76.3	80.6	
	COIN	**100**	86.8	93	
X01590	StemP (Top $1(1)$)	84.2	80.0	85.9	**1.18**
	StemP (Best $2(14)$)	**100.0**	**92.5**	**96.1**	
	RNAPredict	91.7	82.5	86.8	120.45
	SA	53	65	58.4	481.98
AJ251080	StemP (Top $1(8)$)	**94.3**	**86.8**	**90.4**	**0.14**
	RNAPredict	69.7	60.5	64.8	98.07
	SA	52.3	57.9	55.0	398.23
V00336	StemP	**100.0**	**92.5**	**96.1**	**0.27**
	RNAPredict	25.6	25	25.3	99.09
	SA	43.5	50	46.5	397.56
AE002087	StemP	**100.0**	**87.5**	**93.3**	**0.12**
	RNAPredict	75.8	62.5	68.5	123.55
	SA	46.2	45	45.6	490.00

and Two-level particle swarm optimization algorithm (COIN)[105]. The six sequences X67579 (S.cerevisiae), AF03462 (H.marismortui), X01590 (T.aquaticus), AJ251080 (G.stearothermophilu V00336 (E.coli), AE002087 (D.radiodurans) are obtained from Gutell Lab [95]. X67579 belongs to Archaeal organism, AF03462 belongs to Eukaryotic organism and the rest belong to Bacterial organism. The result of TL-PSOfold is from [103], and RNAPredict and Mfold from [101].

For each sequence, we provide the top prediction with Sensitivity, Specificity, and F_1-score listed across different methods. We use the general parameters in Table 2.8. For sequences X01590 and AJ251080, where the best prediction has SCR larger than 1, we show the highest measures among the unique or multiple sequences with SCR = 1. All sequences except for X01590, StemP's top prediction has highest Sensitivity, Specificity, and F_1 among all methods. For X01590, the best prediction, which has SCR = 2, has the highest top prediction among all other methods. In addition, in the best prediction of all 6 sequences, there is no incorrect base-pairs (False Positive) found by StemP, all the False Negative base pairs associated with the best prediction are non-canonical base pairs including `U-C, G-A, G-G, A-A, A-C, U-U, C-C`. Compared to other methods, there is significant improvement in cpu time, where the maximum of 2 seconds is needed for StemP while typically more than a minutes is needed for other methods.

In Table 2.9, we present comparison results of 5s rRNA sequences with PMmulti[106], RNAalifold[107] and Profile-Dynalign [108]. We implement StemP on a test set containing 12 different 5s rRNA Bacterial sequences from Gutell Lab [95] in [108]. The general parameters of Bacterial Sequences in Table 2.8 is used. In Table 2.9, we show the Sensitivity, PPV, and MCC of the top and the best prediction results of StemP. StemP has an average of MCC 0.922 for top predictions and an average of MCC 0.936 for the best predictions. PMmulti + RNAalifold has the highest Sensitivity, which means that fewer False Negative pairs while more incorrect pairs (False Positive pairs) were found. Overall, StemP has higher performance over the other methods when considering both False Negative rate and

Table 2.8: A general bound of parameters for 5S rRNA. This table is based on the true structure in Gutell Lab [95]. (Canonical and Wobble base-pair matching is considered, and Partial Stems are included.)

	Archaeal		Bacterial		Eukaryotic	
	StemLength	SL	StemLength	SL	StemLength	SL
Helix I	6,	17 $\leq$SL$\leq$ 19	8,9,10,$2_{[1/0]}$6	11 $\leq$SL$\leq$ 21	7,8,9,	12.5 $\leq$SL$\leq$ 13.5
Helix II	$2_{[0/1]}$6, $2_{[1/2]}$5, $1_{[1/2]}$6,8	$6 \leq$ SL ≤ 7	$2_{[0/1]}$6, $2_{[0/1]}$5,8,$2_{[0/1]}$8,$1_{[1/1]}$6	$6 \leq$ SL ≤ 7.5	$2_{[0/1]}$6, $2_{[0/1]}$5,$2_{[0/1]}2_{[1/1]}$3, 8	$6 \leq$ SL ≤ 7
Helix III	$2_{[0/2]}$4,$3_{[0/2]}$4	$3 \leq SL \leq 5$	$3_{[0/2]}$4,$2_{[0/2]}$4,$3_{[1/3]}$3,7	$3 \leq SL \leq 5$	$2_{[0/2]}$4,$2_{[0/3]}$4,$3_{[0/2]}$4, $2_{[1/3]}$3	$3 \leq SL \leq 5$
Helix IV	7,6,5	$6 \leq SL \leq 7$	$2_{[1/1]}$2, 5	$8 \leq SL \leq 10$	4,5	$8 \leq SL \leq 11$
Helix V	$6_{[1/0]}$2, $8_{[1/0]}$2, $5_{[2/1]}$2	$2 \leq SL \leq 4$	8, 7, 6, $2_{[1/1]}$4, $3_{[1/1]}$3,$2_{[1/1]}$5,$2_{[2/2]}$5	$2 \leq SL \leq 4$	$2_{[1/1]}2_{[1/0]}$3,$5_{[2/1]}$2,$5_{[1/0]}$3, $2_{[1/1]}2_{[1/0]}$2,7,8	$2 \leq SL \leq 4$
Domain β	$2 \leq GSL \leq 4$					
Domain γ	$2 \leq GSL \leq 4$					

Table 2.9: Comparison of 12 different Bacterial 5s rRNA sequences in [108] using StemP, PMmulti[106], RNAalifold[107] and Profile-Dynalign [108]. Both the top and the best predictions of StemP give highest MCC and PPV compared to other methods.

Method	% Sens	% PPV	MCC
StemP (Top)	92.0	**96.9**	**0.922**
StemP (Best)	87.7	**100.0**	**0.936**
PMmulti	36.8	88.9	0.572
Profile-Dynalign	35.9	94.7	0.583
Clustal W + RNAalifold	86.5	80.3	0.833
PMmulti + RNAalifold	96.6	85.3	0.908
Profile−ynalign + RNAalifold	66.1	80.5	0.729

False Positive rate.

In Table 2.10, we present comparison results on 50 different 5s rRNA sequences in [96]. For StemP, we used the parameters in Table 2.8 (for Archaeal, Bacterial and Eukaryotic) on sequences 1-15, 16-21 and 22-50 respectively. Here, for sequences 1-15, we present results not using any GSL for domain β and γ. This is due to the absent of similar sequences in the learning set of StemP, which is where the parameter bounds are learned. For these sequences 1-15, not using GSL (only using the top Helix I-V parameter) was enough to find the structure prediction, some even giving higher accuracy. This allows more possible helix in each domain to construct maximal cliques. It is shown that 33 out of 50 sequences, StemP's top prediction (with or without GSL) with SCR=1 has higher F_1-score. Overall, StemP has 0.77 as an average of best prediction and 0.73 as highest prediction on this test set, which is higher than 0.635 in [96].

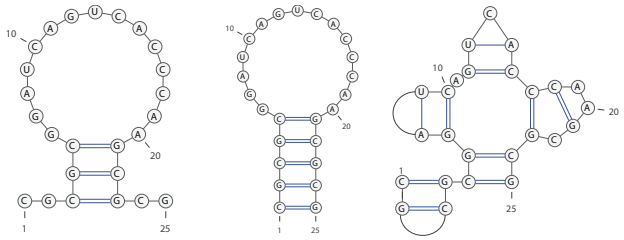
2.4.4 StemP comparison

We present details on three examples of maximum matching (i.e. $L = 1$) compared with StemP with $L = 3$ in predicting RNA sequences. All the sequences are from Protein data bank.

In Figure 2.12, we show (a) the true folding of sequence 2ANN with length 20, and (b)

Table 2.10: Comparison of 5s rRNA sequences between StemP and [96]. The 50 sequences from [96]. We adopted the general bound of parameters for 5S rRNA for Archaeal, Bacterial and Eukaryotic on sequences 1-15, 16-21 and 22-50 respectively. Overall, StemP has higher F_1-score in both the top and the best prediction.

	[96]	StemP			[96]	StemP	
Accn	F_1	F_1^t	F_1^b	Accn	F_1	F_1^t	F_1^b
X07545	0.90	0.85	0.85	K02343	0.85	0.86	0.86
X14441	0.19	0.68	0.84	AB015590	0.67	0.97	0.97
X72588	0.20	0.66	0.84	X06102	0.74	0.86	0.86
M10691	0.47	0.69	0.69	M25016	0.72	0.86	0.86
M36188	0.77	0.00	0.00	X13718	0.70	0.41	0.56
M26976	0.73	0.85	0.86	X06996	0.86	0.94	0.96
X62859	0.63	0.60	0.66	U31855	0.49	0.93	0.94
U67518	0.76	0.67	0.67	M74438	0.84	0.31	0.70
M34911	0.86	0.26	0.26	Z75742	0.36	0.86	0.86
X62864	0.55	0.41	0.45	X01004	0.81	0.33	0.33
X72495	0.94	0.94	1	X00993	0.61	0.86	0.86
AE009942	0.89	0.62	0.62	D00076	0.82	0.59	0.59
M21086	0.88	0.89	0.89	Z93433	0.38	0.86	0.86
X05870	0.88	0.90	0.90	V00647	0.15	0.93	0.94
X07692	0.87	0.89	0.89	M10432	0.31	0.86	0.86
X02627	0.33	0.93	0.95	L49397	0.29	0.93	0.94
V00336	0.27	0.96	0.96	M18170	0.58	0.86	0.86
AJ251080	0.75	0.90	0.93	X00996	0.24	0.69	0.70
M24839	0.25	0.50	0.67	Y14281	0.68	0.86	0.86
M25591	0.79	0.90	0.93	Z33604	0.75	0.30	0.59
U39694	0.72	0.80	0.80	AJ242949	0.83	0.69	0.70
X99087	0.89	0.43	0.75	M24954	0.17	0.93	0.94
X13035	0.74	0.44	0.70	X13037	0.77	0.70	0.70
Y00128	0.79	0.70	0.70	K00570	0.87	0.86	0.86
AB015591	0.60	0.91	0.91	X06094	0.69	0.99	0.99
				Average	0.64	**0.73**	**0.77**



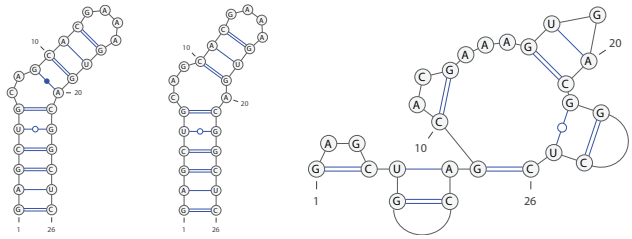
(a) True folding (b) StemP MCC =0

Figure 2.12: StemP vs. maximum matching. (a) The true folding of 2ANN (length 25). (b) The unique StemP top prediction using $L = 3$ without SL condition (Top MCC=0.77). (c) Maximum matching; one of the 3 predictions (all with MCC $= 0$) with top energy 10.

the prediction using $L = 3$ without SL condition with/without pseudoknot enforcement. The MCC of the top prediction is 0.77. In (c), we show one of 3 predictions (all $MCC = 0$) of the top energy 10 cases by considering maximum base pairs (using $L = 1$). In Figure 2.13, we show (a) the true folding of a sequence 2L5Z with length 26 and the unique top prediction using $L = 3$ without SL restriction and with/without pseudonot in (b) with $MCC = 0.95$ where only one non-canonical pair is missing. (c) presents one of the 63 predicted structures ($MCC = 0$) with maximum base pairs using $L = 1$ without SL condition and pseudoknot. Note that 34 out of these 63 structures with maximum base pairs have $MCC = 0$. Although the folding in (b) is also one of these 63 structures, for StemP this is the unique result. In Figure 2.14, we show (a) the true folding of a sequence 2AP5 of length 28, and the unique StemP top prediction using $L = 3$ with $2 \leq SL \leq 20$ and with pseudonot in (b) with $MCC = 1$. (c) gives one ($MCC = 0.89$) of the 56924 (average $MCC = 0.1$) top predictions with energy 10 with 2 false positive base pairs using $L = 1$ with the same SL, pseudoknot condition. These examples show benefits of using proper Stem length condition and Stem-loop score, compared to only considering maximum base pair matching.

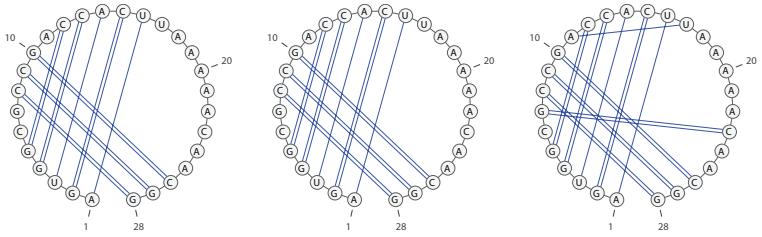
2.5 Further considerations of StemP

In Table A.6, we present experimental results of StemP on general sequences with un-



(a) True (b)StemP (c) MCC=0

Figure 2.13: StemP vs. maximum matching. (a) The true folding of 2L5Z (length 26) (b) StemP using $L = 3$ without SL condition. (c) one of the 63 predicted structures with maximum base pairs using $L = 1$.



(a) True Folding (b) StemP (c) Maximum

Figure 2.14: StemP vs. maximum matching. (a) True fold of 2AP5 (length 28) (b) unique top prediction using $L = 3$ with $2 \leq SL \leq 20$ and with pseudonot with $MCC = 1$. (c) one ($MCC = 0.89$) of the 56924 (average $MCC = 0.1$) top predicted structures with energy 10 with 2 false positive base pairs using $L = 1$ with the same SL, pseudoknot condition. (For this example, we use circular plot to show the difference between structures better.)

known structures. The first dataset is TS0 from [109]. We experimented on a subset of 468 sequences of length ≤ 90. TS0 was selected from the bpRNA-1m dataset[110] which includes sequences from the Comparative RNA Web (CRW), Protein Data Bank (PDB), and the RNA Family database (RFAM) which covers 2495 families of RNA sequences. The family of each sequence is not considered to blind test StemP. We experimented on 316 sequences in TS0 with length in $[50, 81]$ using vertex algorithms for tRNA in Table 2.3, among which 119 sequences has the best prediction with $F_1 \geq 0.95$ while the rest of the sequences has $F_1 \in [0, 0.95]$, with a total average 0.55.

The second dataset in Table 2.11 is a subeset of bpRNA-new containing sequences from 1500 new RNA families extracted by [109] originally from [110]. The subset we experimented on include 2483 sequences of length ≤ 87. In Table 2.11, we show the best prediction of StemP for both subsets from bpRNA-1m and bp-RNA-new. These results can be compared to [109] where comparison results of MXfold2 [109], SPOT-RNA[63], TORNADO[111], ContextFold[51] are presented showing F1 scores ranging around 0.5-0.6, and [4] showing comparison with Ufold [4], RNAstructure[39], RNAsoft[112], e2efold[113], Eternafold[5], Linearfold[45], and Mfold[42] F1 ranging from 0.1 to 0.65. We present the table in Appendix. While many recent methods require training of networks, this method explores deterministic approach giving insight into explicit process of folding, with a simple computation.

For StemP computatin, we consider open ended modification in (Equation (2.5)) for vertex with $d > \hat{l}/2$ for all sequences. For short sequence with length ≤ 65, we use $L = 3$, $2 \leq SL \leq 20$ to find vertex. For longer sequences, we add a stronger condition $d \leq 12$ and $3 \leq SL \leq 6$ to reduce computation cost. If a sequence has a potential to be close to certain structure such as tRNA and 5s rRNA, then StemP is able to predict with high accuracy using corresponding vertex construction algorithm. When family structure is known, we recommend using a general condition of $L = 3$ and $2 \leq SL \leq 20$ as in Table 2.1 along with an open ended modification in (Equation (2.5)). For longer sequences with length > 64,

Table 2.11: StemP result on a TS0, a subset of bpRNA-1m dataset[110], used as a test set in [109] and bpRNA-new from [109, 110]. This can be compared to results in [109] and [4]. We consider open ended modification in (Equation (2.5)) for vertex with $d > \hat{l}/2$ for all sequences. For short sequence with length ≤ 65, we use $L = 3$, $2 \leq SL \leq 20$ to find vertex. For longer sequences, we add a stronger condition $d \leq 12$ and $3 \leq SL \leq 6$ to reduce computation cost.

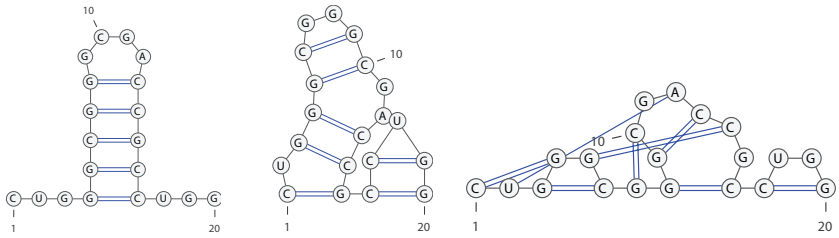
TS0	Count	%F1	%Sen	%PPV
StemP(Best)$[1, 90]$	468	**0.714**	**0.775**	**0.701**

bpRNA-new	Count	%F1	%Sen	%PPV
StemP(Best)$[1, 87]$	2483	**0.737**	**0.771**	**0.726**

we enforce a stronger condition, such as $d \leq 12$ and $3 \leq SL \leq 6$ to reduce computation cost.

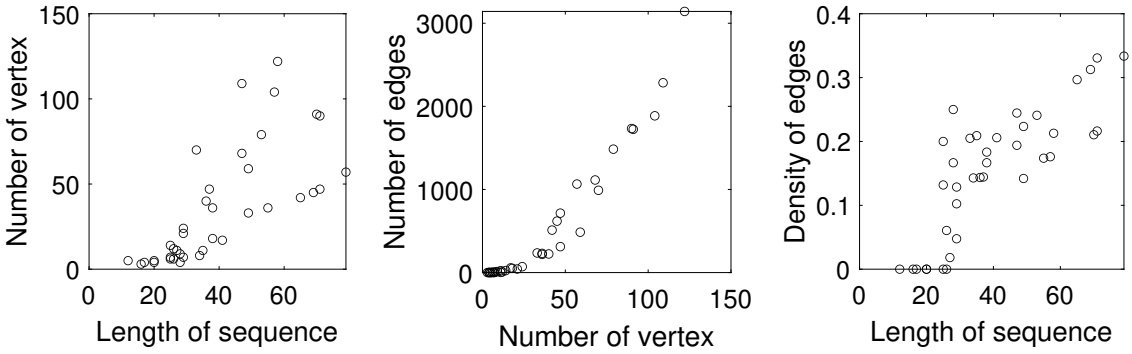
Figure 2.15 shows an comparison of a simple dynamic programming schemes that aim at maximizing the number of base-pairs, compared to StemP and the true folding. We show true folding and identical prediction results from StemP ($L = 3$, with/without SL condition) in (a). Considering maximum base pair matching (i.e. $L = 1$) is presented in (b) and (c). We obtained 3 different prediction by considering maximum base pairs with $L = 1$ without pseudoknot with all $MCC = 0$. We show one of theses prediction in (b). (c) gives one of the 24744 predictions with top energy 8 by considering maximum base pairs using $L = 1$ with pseudoknot. In addition to being able to find pseudo-knots, StemP using SL helps with prediction. More examples of how prediction can be improved using a proper L and SL are shown in Section 2.4.4.

StemP gives a good prediction typically within a few seconds of CPU time. Typically algorithms can be time consuming, e.g., as expensive as $O(N^6)$ [114], and it is difficult to quantify the time complexity across different methods. The computation cost of StemP depends on the number of vertex n and the density of edges in the Stem-graph, which we measure by # edges$/(N(N-1)/2)$. StemP has a computational advantage over a standard dynamic programming for free energy minimization, given by limiting the total number of all possible base pairs: from $\frac{N(N-1)}{2}$ combinations which cost $O(N^3)$[43], to considering



(a) StemP (MCC=1) (b) MCC =0 (c) MCC=0

Figure 2.15: (a) StemP gives unique top prediction, MCC =1 (with $L = 3$), which is the true folding of 361D (length 20). (b) one of the 3 predictions (all with MCC $= 0$) with top energy 7 via maximum base pairs (i.e. $L = 1$) without pseudoknot. (d) one of 24744 predictions with top energy 8 via maximum base pairs ($L = 1$) with pseudoknot.



(a) (b) (c)

Figure 2.16: (a) The length of sequence (x-axis) vs the number of vertex (y-axis). (b) The number of vertex (x-axis) vs the number of edges (y-axis). (c) The length of sequence (x-axis) vs the density of edges (y-axis). The experiment is done on 33 sequences from Protein Data Bank.

n vertex via imposed Stem-Loop Score and minimum stem length L. In Figure 2.16, we show how the computation scales as the length of RNA increases. (a) and (b) show in general longer sequence gives more number of vertex and more edges in the Stem-graph. Otherwise, (c) shows that the cost of computation is not directly correlated to the length of the sequence, but to the combination of the number of vertex and edges, i.e. complexity of Stem-graph which depends on each sequences.

In the Appendix, we present details of the algorithms and results, e.g., additional results, statistical result of tRNA in Figure 2.6, prediction results of 5s rRNA in Table 2.7 and Table A.1, and prediction of 15 5s rRNA in Table 2.10 without GSL.

CHAPTER 3

COUNTING OBJECTS BY DIFFUSED INDEX - IDENTIFYING THE QUANTITY OF OBJECTS IN DIGITAL IMAGES

This chapter reproduces our previously published paper [2]. The author of this thesis personally contributed to the Methodology, Investigation, Software, Visualization, and Writing – original draft.

In this chapter, we propose a diffusion-based geometry-free and training-free counting method. The main idea is to give a unique index to each object regardless of its intensity or size, and to simply count the number of indexes. First, we place different-value vectors, i.e. seed vectors, uniformly through out the given image. The seed values are independent from requiring precise prior knowledge about the image and objects to be counted. Secondly, these seed vectors are diffused using an edge-weighted harmonic variational optimization model to give a unique index to each object. Our edge-weighted harmonic variational optimization model is motivated by [115, 116] which was used for color image inpainting [117]. Inspired by recent developments on solving structured optimization models [118, 119, 120, 121, 122, 123, 124, 117, 116], we exploit variable splitting, alternating direction method of multipliers, as well as periodic boundary condition to develop a fast algorithm to solve this optimization model. We refer this part as Diffusion Algorithm. An optimal solution of the model is reached when the uniformly distributed seeds are diffused and reached different gray-level intensities. At this point, each object in the image has a unique index. For efficiency and more flexibility, we cluster the index values of each pixel before the Diffusion Algorithm is fully converged. We investigate both scalar and multi-dimensional seed vectors. For scalar seed vectors, we count the number of peaks in the Gaussian fitted curve of the histogram. For multi-dimensional seed vectors, we use high dimensional density based clustering algorithm. The main contribution of this paper is outlined below:

1. We introduce new simple geometry-free and training-free counting methodologies.

2. We propose a fast diffusion algorithm to count number of objects.

3. The proposed method works well for various applications without being depended on geometry or training. This method can count objects without clear or closed boundaries. We propose a simple extension to counting different size objects separately.

3.1 Literature Review

Counting object is an important problem in various applications such as biological cells [125], production line items [126], vehicles [127, 128], plant organs [129, 130, 131] , animals [12, 128], crowd [12, 128, 132, 133] counting and others. In literature, various different approaches have been explored. Studies such as watershed [134, 6] and floodfill [135, 125] consider cases where the objects to be counted have uniform intensity, similar shapes and sizes, and are disconnected from each other by distinct background color. For these classical techniques, the counting results are highly dependent on the quality of segmentation result of a given image. Utilizing geometrical features of objects can be useful in such cases. Hough Transform is often implemented to segment objects with a similar circular shape [126, 136, 7], and aid the segmentation stage. If the objects have overlapping boundaries, more preprocessing is required. For instance, in [137], the authors split the blood cell clumps by finding the maximum curvature on object boundaries and use Delaunay triangulation. In [138], the authors first detect concavity at the edge of a cluster to find the points of overlaps between two nuclei, then use the ellipse-fitting technique for segmentation. There are other detection oriented segmentation methods, such as, integrating representative [8], hough transform technique in detection [139, 140, 141], principle component analysis combined with histogram processing [9], low-rank decomposition[142], and saliency diffusion [143].

In machine learning, quantification problems is solved by segmentation, i.e., a detec-

tion based method. For example, in [144], the authors present an ImageJ plugin which is an adapted U-Net for single-cell segmentation and quantification based on a pre-trained model. In the scenario of counting by segmentation, object detection models such as Mask R-CNN [145] can be useful for counting. There are certain limitations for the cases of a lot of overlapping objects, objects with varying textures, or in lack of large enough data set with accurate boundary annotations. More recent works focus on learning from a density map of which the integration gives the estimate of number of objects by regression [146, 13, 147, 148, 149, 150, 151]. SAU-Net [152] incorporates a self-attention module with a segmentation network, U-Net, to learn the density map. Such regression based methods only require dot annotations as ground truth in the training set. There are other works that directly learn from image patches without generating a density map. The authors of [12] adapt a convolutional neural network to produce a patch-based regressor to count people, animals, vehicles and cells. The authors of [153] propose an architecture adapted from the Count-ception network to perform redundant counting based on receptive field to average over errors. In [154], the author formulates the counting task as an image classification problem and takes the counts as class labels. In [155], the authors consider mapping objects to blob-like structures and applied a Laplacian of Gaussian filter to localize objects. Overall, learning based methods can be time consuming, and a large dataset with ground truth is always necessary. An accurate count is usually learned by a particular network for a certain type of objects such as cells or crowd. Learning to count general objects is still a challenging task.

3.2 Proposed Model

Let us consider a given image in which there are objects to count. We aim to give a unique index to each object regardless of its intensity, shape or size, then we simply count the number of indexes to provide the quantity of the objects. There are three simple steps to this method:

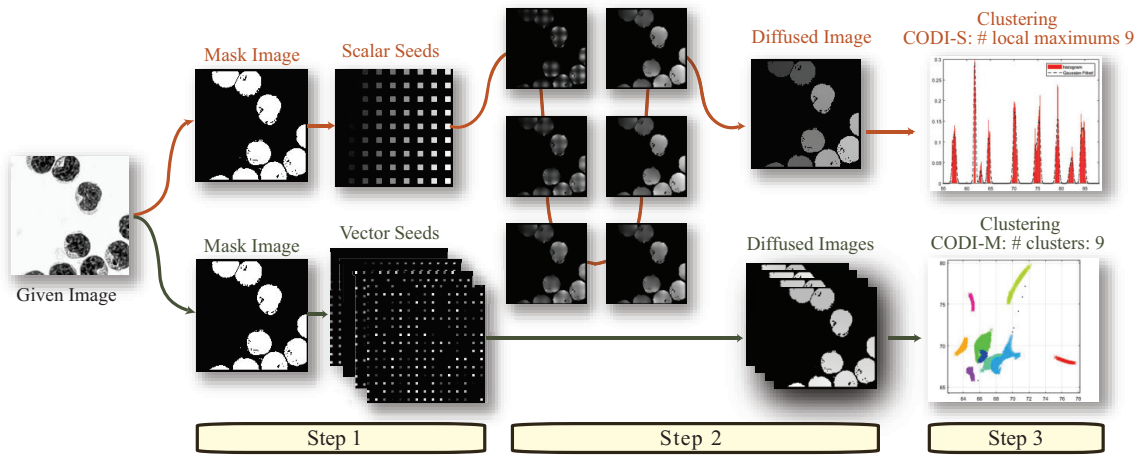


Figure 3.1: Outline of two proposed counting methods (scalar or vectorial seeds). Given image with 9 cells. **[Step 1]** Uniformly distributed seed (Scalar or multi-dimensional seeds). **[Step 2]** Diffusion of seeds to find unique index for each object. **[Step 3]** Counting stage: the number of indexes is counted using clustering methods. Both methods give 9 objects.

- **[Step 1]** Place different gray-value seeds uniformly though out the given image;
- **[Step 2]** Diffuse the seeds to obtain different index values within each object;
- **[Step 3]** Counting the different indexes to obtain the number of objects. We can further cluster objects based on their size.

Outline of the proposed method is presented in Figure 3.1. Based on the given image, in [Step 1] we put uniformly distributed seed onto a corresponding mask image. We choose the seeds to be all different from each other. In [Step 2], the seeds, whether scalar or multi-dimensional, are diffused within each object. The diffusion process is done by an iterative algorithm where after the decay rate reaches to a certain level, each object is reached to a different gray-intensity value. This is shown in [Step 3] via histogram of the diffused image. Each object is associated to a peak in the histogram. In [Step 3], we provide two counting methods for scalar and vectorial seeds.

3.2.1 Ingredients: seed, mask, and edge images [Step 1]

Let $\Omega \subset \mathbb{R}^2$ be the image domain with Lipschitz boundary and $\Phi_0 : \Omega \to \mathbb{R}$ be the given image. We place different gray-value seeds through out the given image. Let $U_0 : \Omega \to [0, 255]$ denote the seed image with M different seeds $s_{i,j} : \Omega_{i,j} \to v_{i,j}$, where $\Omega_{i,j} \subset \Omega$, $i = 1, 2, \ldots, n_1$, $j = 1, 2, \ldots, n_2$, $M = n_1 n_2$, and $n_1, n_2 \in \mathbb{N}$. We explore both scalar value seed as $v_{i,j} \in (0, 255]$ and multi-dimensional seed as $v_{i,j} \in \mathrm{R}^N$. For multi-dimension seeds, we use superscript to represent each dimension, e.g., U_0^j with $j = 1, 2, \ldots, N$. Different seeds are placed on a small region $\Omega_{i,j} \subset \Omega$ such that $D = \cup_{i=1}^{n_1} \cup_{j=1}^{n_2} \Omega_{i,j} \subset \Omega$, and $D^c = \Omega \backslash D \subset \Omega$. In practice, $\Omega_{i,j}$ are considered to be square shape, all with the same size, and dimension $d \times d$, and the distance between two adjacent seeds to be l. Outside of the seeded region D^c, U_0 is set to be zero. For scalar seeds, we set a constant gray-scale values $v_{i,j} \in (0, 255]$ on each seed domain $\Omega_{i,j}$. Thus, the seed image U_0 has $M + 1$ gray values $\{0, v_{1,1}, \ldots, v_{n_1,n_2}\}$ such that for any $x \in \Omega$, $U_0(x) = v_{i,j}$ if $x \in \Omega_{i,j}$, $i = 1, 2, \cdots, n_1$, $j = 1, 2, \cdots, n_2$, and $U_0(x) = 0$ otherwise. Typically, we picked $v_{i,j} = \frac{255}{M}[(i-1)n_2 + j]$ for $i = 1, 2, \cdots, n_1, j = 1, 2, \cdots, n_2$ as uniformly distributed value in $(0, 255]$.

Depending on the image and the objects, to stabilize the small separation between objects and to avoid having the same index for different objects, we also utilize multi-dimensional seeds. Figure 3.2 shows a multi-dimensional seed, where each seed dimension is shown separately in (a)-(d). In the first dimension, we increase the seed values in x-direction (horizontally) then $y-$direction (vertically) such that the lowest value is located on the upper-left corner and highest value is on the bottom-right corner. This is identical to the scalar seeds, i.e., $U_0^1 = U_0$. In the second dimension, we start with the bottom-left corner, increase the values in y-direction first then increase in x-direction, where the lowest gray-value is on the bottom-left corner while the highest gray value is on the uppper-left corner: $U_0^2(x) = v_{i,j}$ if $x \in \Omega_{i,j},\ i = 1, \ldots, M$ where seeds are assigned in the same logic: $v_{i,j} = \frac{255}{M}[n_1 n_2 - i n_2 + j]$ for $i = 1, 2, \cdots, n_1, j = 1, 2, \cdots, n_2$. We add two additional dimensions with random seeds given by a random permutation of the set $\{v_1, ..., v_M\}$. The

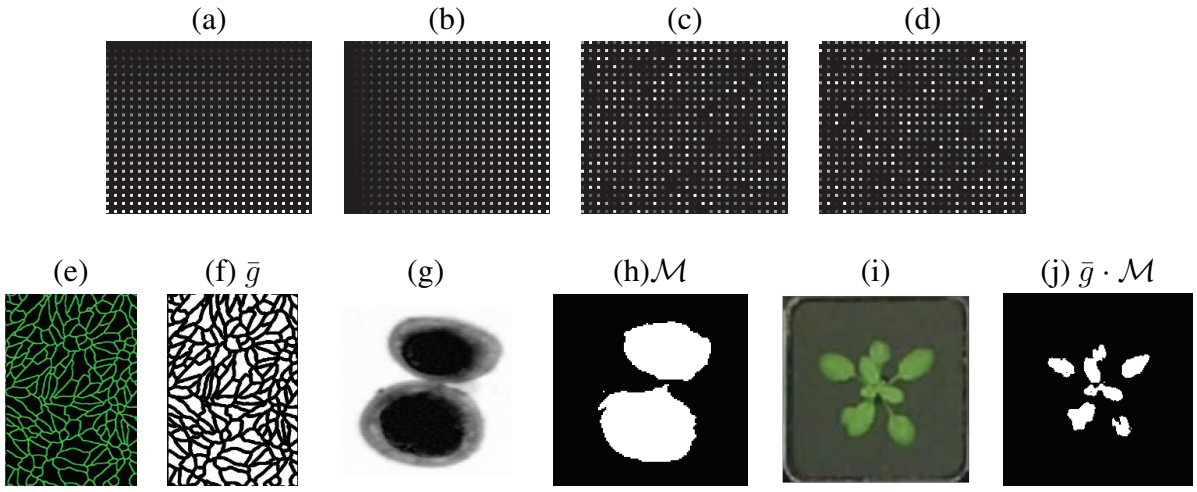


Figure 3.2: [4 dimensional seed, edge function and mask image] (a)-(d) shows an example of multi-dimensional seed. (a) U_0^1 (horizontal), (b) U_0^2 (vertical), (c) U_0^3 (random) (d) U_0^4 (random). (e), (g) and (i) are three given images, (f), (h) and (j) show the corresponding edge function $\bar{g}$, a mask image $\mathcal{M}$ and a mask and edge function $\mathcal{M} \cdot \bar{g}$ used for each image respectively.

values v_i of seeds in different dimensions are identical, but the order of placement is different in each dimension. We recommend $p \geq 3$ for multi-dimensional seeds, where p denotes the number of seed dimension, i.e. $U_0 = [U_0^1, \ldots, U_0^p] \in \mathbb{R}^p$. Through out this paper, we consider $p = 4$.

The most important geometric features of any image are the edges. When objects in a given image Φ_0 are separated by edges, we define a continuous monotone decreasing function $\hat{g}(t) : \mathbb{R} \to [0, 1]$ such that $\hat{g}(|\nabla \Phi_0|)$ gives the edge information. Here ∇ denotes the gradient operator and $|\cdot|$ represents the ℓ_2 norm. Some examples of $\hat{g}$ includes

$$\tilde{g}(t) = e^{-\tau t^2}, \quad \text{and } \bar{g}(t) = (1 + \tau t^2)^{-1} \tag{3.1}$$

with $\tau > 0$. With the monotone decreasing property of $\hat{g}(t)$ with respect to t, the diffusion process stops close to the object boundaries. To eliminate the coarse boundary features and remove noise, we consider $g(\Phi_0) = \hat{g}(|\nabla(G_\sigma * \Phi_0)|)$, where G_σ denotes the two dimensional Gaussian function with the variance σ.

If objects are separated with a different background color, we utilize a mask image

$\mathcal{M} : \Omega \to [0, 1]$ that is a binary image having zero values on the background and one on the objects. In this case, we set $g(\Phi_0) = \mathcal{M}$ in the model (Section 3.2.2). Both $\hat{g}$ and $\mathcal{M}$ can be considered at the same time when it is required to distinguish the objects from background as well as edges from objects. Figure 3.2 gives three examples of using edge function and mask images, where $\hat{g}$, $\mathcal{M}$ and $\mathcal{M} \cdot \hat{g}$ are suggested for image (e), (g) and (i) respectively.

3.2.2 Diffusion Phase [Step 2]

The weighted harmonic variational diffusion model is given by

$$\min_{U} \ \Big\{ F_\eta[U] \Big| \ U \in BV(\Omega; \mathbb{R}^2) \quad a \leq U(x) \leq b \Big\}, \text{ where} \tag{3.2}$$

$$F_\eta[U] = \int_\Omega g(\Phi_0) |\nabla U|^2 dx + \frac{\eta}{2} \int_{D^c \cap \mathcal{M}} |U - U_0|^2 dx.$$

The first term is the regularization term and the second term is the data fidelity term, $\eta > 0$ is the fidelity parameter, $0 < a < b < 255$, ∇U denotes the gradient of the image U defined by $\nabla U := (\partial_x U, \partial_y U)$, where ∂_x and ∂_y are the partial derivatives along the horizontal and vertical directions, and the function g is described near (Equation (3.1)) in Section 3.2.1. The parameter η in the fidelity term enforces the solution to stay close to the seed image U_0 on the regions $\Omega_{i,j}$, $i = 1, \ldots, n_1$, $j = 1, \ldots, n_2$. To obtain the unique index in each object, η should not chosen too large.

We propose the diffusion algorithm to solve (Section 3.2.2) efficiently, by exploiting variable splitting and alternating direction method of multiplier [118, 121, 122, 123, 117, 116]. We let the auxiliary variable $V \in L^2(\Omega; \mathbb{R})$ and we define $\Gamma := \{V \in L^2(\Omega; \mathbb{R}) | a \leq V(x) \leq b\}$. We rewrite (Section 3.2.2) equivalently as the following constrained optimiza-

tion problem

$$\min_{U,V} \Big\{ \int_\Omega g(\Phi_0)|\nabla U|^2 \, dx + \frac{\eta_D}{2} \int_\Omega |V - U_0|^2 dx \Big\} \tag{3.3}$$
$$\text{subject to} \quad V = U \ \text{ and } \ V \in \Gamma,$$

where $\eta_D : \Omega \to (0, +\infty)$ is given by

$$\eta_D(x) = \begin{cases} 0, & x \in D \\ \eta, & x \in D^c \cap \mathcal{M}. \end{cases} \tag{3.4}$$

We let λ be the Lagrange multiplier associated with the linear constraint $V - U = 0$. The augmented Lagrangian functional associated to (Equation (3.3)) is given by

$$\mathcal{L}_\mu(U, V, \lambda) = \int_\Omega \Big\{ g(\Phi_0)|\nabla U|^2 + \frac{\eta_D}{2}|V - U_0|^2 + \langle \lambda, V - U \rangle + \frac{\mu}{2}|V - U|^2 + \chi_\Gamma(V) \Big\} dx,$$

where $\mu > 0$ penalty parameter, $\chi_\Gamma(V)$ is the indicator function given by

$$\chi_\Gamma(V) := \begin{cases} 0, & V \in \Gamma \\ +\infty, & \text{otherwise.} \end{cases}$$

The algorithm to solve (Equation (3.3)) is given as follows. We initially set $k = 0$, and let $V^{(0)} = 0$ and $\lambda^{(0)} = 0$ be the initial values. For any $k \geq 1$, given $V^{(k)}$ and $\lambda^{(k)}$, we compute $U^{(k+1)}$ by solving

$$U^{(k+1)} = \arg\min_U \mathcal{L}(U, V^{(k)}, \lambda^{(k)}).$$

More precisely, we compute $U^{(k+1)}$ by solving

$$\min_U \int_\Omega \Big\{ g(\Phi_0)|\nabla U|^2 + \langle \lambda, V^{(k)} - U \rangle + \frac{\mu}{2}|V^{(k)} - U|^2 \Big\} dx. \tag{3.5}$$

To find the close-form solution and to encourage a fast diffusion, we modify this energy functional as follows. We let $G_0 = \max\big\{g(\Phi_0(x)) \ \big| x \in \Omega\big\}$, $\mathcal{H}(U) = \big(g(\Phi_0(x)) - G_0\big)|\nabla U|^2$, and write $g(\Phi_0)|\nabla U|^2 = G_0|\nabla U|^2 + \mathcal{H}(U)$. We exploit the second order Taylor polynomial approximation of $\mathcal{H}(U)$ about $U^{(k)}$ to get

$$\begin{aligned}\mathcal{H}(U) &\approx \mathcal{H}(U^{(k)}) + \Big\langle \nabla\mathcal{H}(U^{(k)}), U - U^{(k)} \Big\rangle + \frac{\theta}{2}|U - U^{(k)}|^2 \\ &= \Big(g(\Phi_0) - G_0\Big)|\nabla U^{(k)}|^2 + \Big\langle 2\nabla\cdot\Big((G_0 - g(\Phi_0)\nabla U^{(k)}\Big), U - U^{(k)}\Big\rangle + \frac{\theta}{2}|U - U^{(k)}|^2,\end{aligned}$$

where $\theta > 0$ is a scalar. With this approximation, the U-minimization subproblem (Section 3.2.2) becomes

$$\begin{aligned}U^{(k+1)} = \arg\min_U & \int_\Omega G_0|\nabla U|^2\, dx + \int_\Omega \Big\langle 2\nabla\cdot\Big(G_0 - g(\Phi_0)\nabla U^{(k)}\Big), U\Big\rangle\, dx \\ & + \frac{\theta}{2}\int_\Omega |U - U^{(k)}|^2 dx + \frac{\mu}{2}\int_\Omega |U - V^{(k)} - \mu^{-1}\lambda^{(k)}|^2 dx.\end{aligned}$$

The first-order optimality conditions of this problem is given by

$$\big((\theta+\mu)\mathcal{I} - 2G_0\Delta\big)U^{(k+1)} = \theta U^{(k)} + 2\nabla\cdot\Big((g(\Phi_0) - G_0)\nabla U^{(k)}\Big) + \mu V^{(k)} + \lambda^{(k)}.$$

We exploit the Fast Fourier Transform (FFT) to obtain the closed form solution of this problem. Since $\mathcal{F}\mathcal{F}^{-1} = \mathcal{I}$ we then obtain

$$U^{(k+1)} = \mathcal{F}^{-1}\Big[\mathcal{F}\Big(\theta U^{(k)} + 2\nabla\cdot\Big((g(\Phi_0) - G_0)\nabla U^{(k)}\Big) + \mu V^{(k)} + \lambda^{(k)}\Big)/\mathcal{D}\Big],$$

where $\mathcal{D} = (\theta+\mu)\mathcal{I} - 2G_0\mathcal{F}(\Delta)\mathcal{F}^{-1}$.

Next, we compute $V^{(k+1)}$ given $U^{(k+1)}$ and $\lambda^{(k)}$ by solving

$$V^{(k+1)} = \arg\min_V \mathcal{L}(U^{(k+1)}, V, \lambda^{(k)}), \quad \text{where}$$

$$\mathcal{L}(U^{(k+1)}, V, \lambda^{(k)}) = \int_\Omega \Big\{ \frac{\eta_D}{2}|V-U_0|^2 + \langle \lambda^{(k)}, V - U^{(k+1)} \rangle + \frac{\mu}{2}|V - U^{(k+1)}|^2 + \chi_\Gamma(V) \Big\} dx.$$

The objective function is the sum of a qudratic functional and an indicator function, so the solution is given in a closed form in form of projection as follows

$$V^{(k+1)}(x) = \mathrm{Proj}_\Gamma(\gamma), \quad \text{where} \quad \gamma = \frac{\eta_D U_0(x) + \mu U^{(k+1)}(x) - \lambda^{(k)}(x)}{\eta_D(x) + \mu}.$$

By the definition of Γ, then we have

$$V^{(k+1)}(x) = \begin{cases} a & \gamma \le a \\ \gamma & a \le \gamma \le b \\ b & \gamma \ge b \end{cases}.$$

Finally, we update the multiplier $\lambda^{(k+1)}$ by

$$\lambda^{(k+1)} = \lambda^{(k)} + \mu(V^{(k+1)} - U^{(k+1)}).$$

This algorithm is referred as Diffusion Algorithm, summarized in Algorithm 5.

Algorithm 5 The Diffison Algorithm

Data: A digital image Φ_0, mask image $\mathcal{M}$, seed image U_0
Output: Diffused Image U_*
Initialization: $k = 0$; $V^{(0)} = \mathbf{0}$, $\lambda^{(0)} = \mathbf{0}$
Parameters: $\mu > 0, \theta > 0, \eta > 0$
Set $\mathcal{D} = (\theta + \mu)\mathcal{I} - 2G_0\mathcal{F}(\Delta)\mathcal{F}^{-1}$
For $k = 1, 2, \ldots$ do

$U^{(k+1)} = \mathcal{F}^{-1}\Big(\mathcal{F}\big(\theta U^{(k)} + 2\nabla \cdot \big((g(\Phi_0) - G_0)\nabla U^{(k)}\big) + \mu V^{(k)} + \lambda^{(k)}\big)/\mathcal{D}\Big)$;
$V^{(k+1)}(x) = \mathrm{Proj}_\Gamma(\gamma(x)), \quad \gamma(x) = (\eta_D U_0(x) + \mu U^{(k+1)}(x) - \lambda^{(k)}(x))/(\eta_D(x) + \mu)$;
$\lambda^{(k+1)} = \lambda^{(k)} + \mu(V^{(k+1)} - U^{(k+1)})$;

If stopping criteria satisfied, set $U_* = U^{(k+1)}$.

For multi-dimensional seeds, the minimization problem (Section 3.2.2) is solved for each seed dimension separately and can be performed in parallel for efficiency. As a result,

the diffused image is also multi-dimensional.

The converged image U_* has a diffused value of nearby seeds. The parameter η in (Section 3.2.2) controls how close the value in the domain $\Omega_{i,j}$ to the given seed $s_{i,j}$, and it is not necessary to keep η very large. Since the edge function g gives information about the boundary of the objects, the diffusion will stop or slow down near the boundary, and give unique index to each object. The value of the diffused image U gives unique index d_i to each object for $i = 1, \ldots, K$, i.e., we refer to this as the diffused index image.

3.2.3 Clustering and Counting [Step 3]

During the diffusion process, seeds within each object start to converge to an unique index d_i, for $i = 1, \ldots, J$. Considering the histogram $H(U)$ of image U, the number of local maximum J in $H(U)$ starts from the total number of seed plus zero value on D^c, i.e., $M + 1$, and converges to K, the number of objects. Since different values of seeds are placed uniformly, especially when multidimensional seeds are used, it is highly unlikely for two objects in different locations to converge to the same index. Local seeds converge to one unique index d_i as long as they are within one object.

In **[Step 3]**, we count the number of local maximum by clustering the histogram $H(U)$ of U. Each local maximum represents d_i, a unique index for an object, and the number of such local maximums K is the number of the objects in the image and bigΦ_0.

For one-dimensional scalar seed image, we consider Gaussian fitting on $H(U)$ and we refer to it as Counting Objects by Diffused Index - Scalar seed clustering (CODI-S). The histogram can be described as a discrete function $h(r_k) = n_k/N$ where r_k is the kth gray level intensity within the range $[0, 255]$ in U, n_k is the number of pixels having the intensity value r_k, and N is the total number of pixels in the image. A discrete Gaussian filter $p(i) = \frac{1}{\sigma\sqrt{2\pi}} e^{-i^2/2\sigma^2}$ for $i = -r, \cdots, r$, (where $\sigma > 0$ denotes the variance) is considered onto $H(U)$ to obtain a smooth fitting curve. The number K of local maximums is counted by implementing binary search recursively. A larger r and bigger σ results in smoother

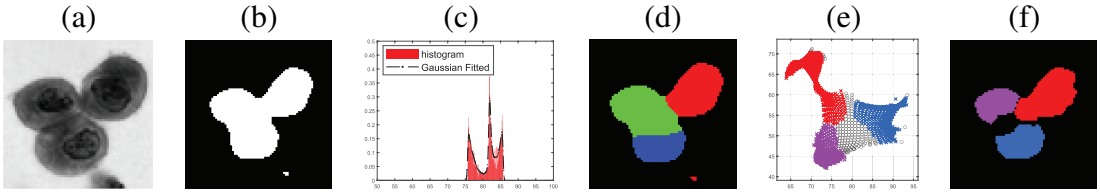


Figure 3.3: [CODI-S and CODI-M] (a) Given image with three cells (b) The mask image showing open boundaries between the objects. (c) The histogram and Gaussian fitted curve of CODI-S. (d) The visualization of CODI-S clustering in image domain. (e) The clustering results of CODI-M, projected onto two dimension for visualization. (f) The visualization of CODI-M clustering in image domain. Both methods counts three cells.

curve which gives a fewer number of local minimums. A smaller r and smaller ϵ involved more details from the labeled pixels that it can give larger number K. CODI-S has a large stable range of optimal parameters, due to smoothing $H(U)$ with the Gaussian convolution. We used $\sigma \in [0.05, 1.2]$, and $r = 5$ though out this paper. Figure 3.3 demonstrates the result obtained by the CODI-S on the cell image shown in (a). The mask image described in [Step 1] is shown in (b). Notice the open boundaries between the three cells. (c) demonstrates the histogram and Gaussian fitted curve after the diffusion process [Step 2]; we observe three peaks where each peak corresponds to each cell. The visualization of the clustering by CODI-S is also shown in (d), where the histograms in (c) are split into 3 sets at the two minimum values between the local maximums.

For multi-dimensional seeds, we use high dimensional density method, such as DBSCAN, and refer to as Counting Objects by Diffused Index - Multi-dimensional seed clustering (CODI-M). Using DBCAN [156], the seed vector similarity is tracked by the density, defined by ϵ and MinPts via l_2 Euclidian distance norm. Here ϵ defines ϵ-neighborhood, $N_\epsilon(\boldsymbol{x}) = \{\boldsymbol{y} \in \mathbb{R}^p : dist(\boldsymbol{x}, \boldsymbol{y}) \leq \epsilon\}$, and MinPts is the minimum number of points required within ϵ-neighborhood to be connected as one cluster. This property is called *direct density reachability* of $\boldsymbol{x}$ from $\boldsymbol{y}$. For points $\boldsymbol{x}$ that does not have density reachable points in its ϵ-neighborhood, they are classified as noise. To find a cluster of 4-dimensional histogram $\boldsymbol{H(U)}$ of $\boldsymbol{U}$, we start with an arbitrary point $\boldsymbol{U}(x)$ and retrieves all density-reachable points from $\boldsymbol{U}(x)$ with respect to given $\epsilon > 0$ and $\mathrm{MinPts} > 0$. The reaching

procedure ends when all points in a cluster has been visited and all the neighboring points in distance ϵ from any of the point in this cluster have been included in this cluster. The next step is to move onto the next unvisited point. The accuracy of the method depends on the two parameters ϵ and MinPts. A relatively small MinPts and large ϵ gives fewer and bigger clusters, while a relatively large MinPts and small ϵ leads to more and smaller clusters. In this paper, we use $\epsilon \in [1, 1.2]$, MinPts $\in [12, 18]$ as the optimal range.

In CODI-M with DBCAN, clusters $\{C_i | i = 1, \ldots, K\}$ are formed, where the centroid is the unique index $\boldsymbol{d_i}$ for each object. In Figure 3.3 (e) is a projection in two directions for visualization. Each object is visualized in multi-dimensional space with a different color. The number of different colors accurately gives the number of cells K. For each data $x \in \Omega$, it is associated with the diffused index value and a cluster

$$\{(x, \boldsymbol{U}(x), C_i) : \quad U(x) \in C_i, \quad i = 0, 1, \ldots, K\},$$

where C_i denotes the i-th cluster in the high dimensional histogram domain, and K denotes the count. Let C_0 stores x which is considered as noise, and in Figure 3.3 (e), C_0 is marked as black circles. (f) shows a visualization of each cluster C_i in Ω for $i = 1, 2$ and 3.

3.3 Properties of the proposed methods

In this section, we focus on a few interesting properties of the proposed methods. Since the proposed CODI counts the diffused index before the full convergence of the diffusion algorithm, we utilize this aspect to count objects that have open boundaries and explore this aspect. Secondly, since we use clustering methods to count the diffused index, we can further extend this idea and count different-sized objects separately. By using regularized k-means [157], we show how different-sized objects can be separately counted just by one simple additional step.

3.3.1 Open boundary and counting accuracy

The proposed CODI counts the diffused index before the full convergence of diffusion algorithm is reached. These method can handle not fully closed boundaries in the objects, and we present the effect of such cases. In Figure 3.4, three synthetic images with different open boundaries are presented: (a) thick and narrow boundary opening, (b) thin and narrow boundary opening, and (c) thin and wide boundary opening. The given image size is 47×91 with the sizes of gaps as (a) 9×15, (b) 9×3 and (c) 21×3. Identical seeds distribution U_0^1 is used for CODI-S and the first dimension of CODI-M. For CODI-M, we use U_0^2 for second dimension, and two random seeds similar to the idea in Figure 3.2 for third and forth dimensions. The third and forth columns demonstrate CODI-S and the fifth and sixth columns demonstrate CODI-M after 40 and 80 iterations of the diffusion process respectively.

We observe that even after short iterations for images (a) and (b), both CODI-S and CODI-M find two objects clearly, even with partially opened boundary. (a1)- (a4) and (b1)-(b4) all finds two objects. When the boundary opening is large and separation between the objects are not very clear like image (c), it is better for CODI-S to have smaller number of iteration while CODI-M needs a larger number of iterations.

3.3.2 Further grouping counts of similar sized objects

Since the proposed method utilize clustering of $\boldsymbol{H}(\boldsymbol{U})$, we can further distinguish different sizes after the clusters C_is are found. The clustering of $\boldsymbol{H}(\boldsymbol{U})$ gives data $x \in \Omega$ in the form of

$$\{(x, U(x), C_i) : \quad U(x) \in C_i, \quad i = 0, 1, \ldots, K\},$$

where C_i denotes the i-th cluster in the multidimensional histogram domain, and K denotes the counting result. Now, we consider the size of each clusters $S = \{|C_i||i = 1, \ldots, K\}$ and use the Regularized k-means algorithm [157] to further cluster this set S. The Regu-

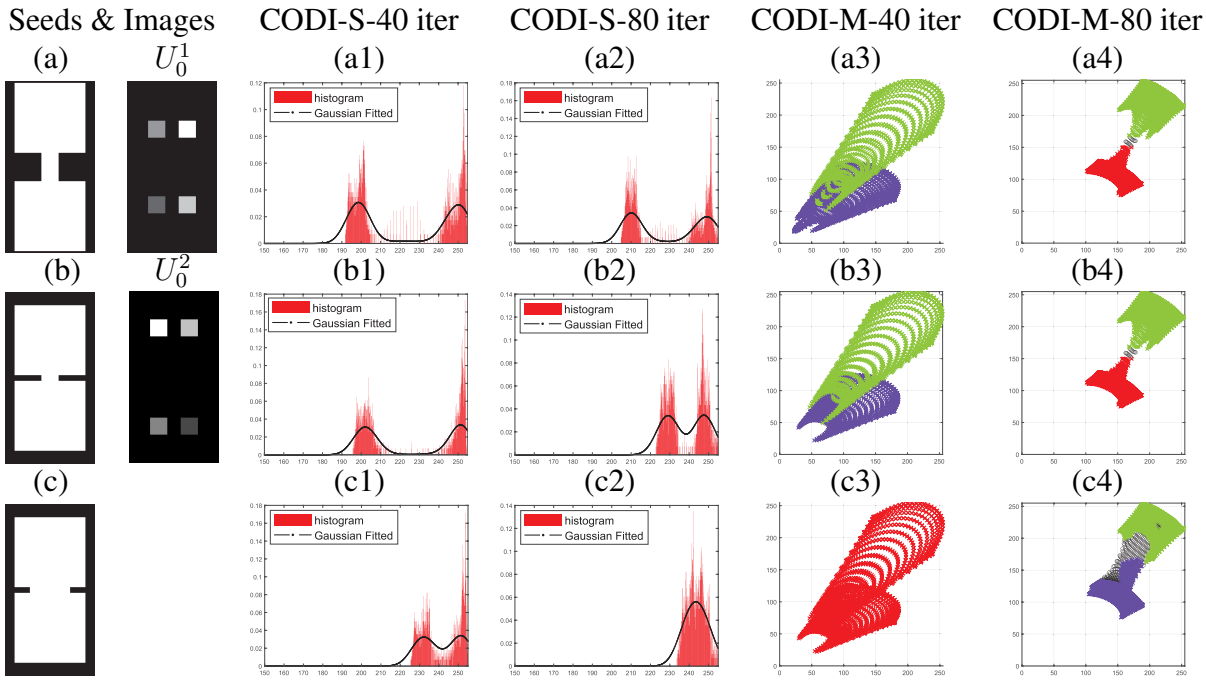
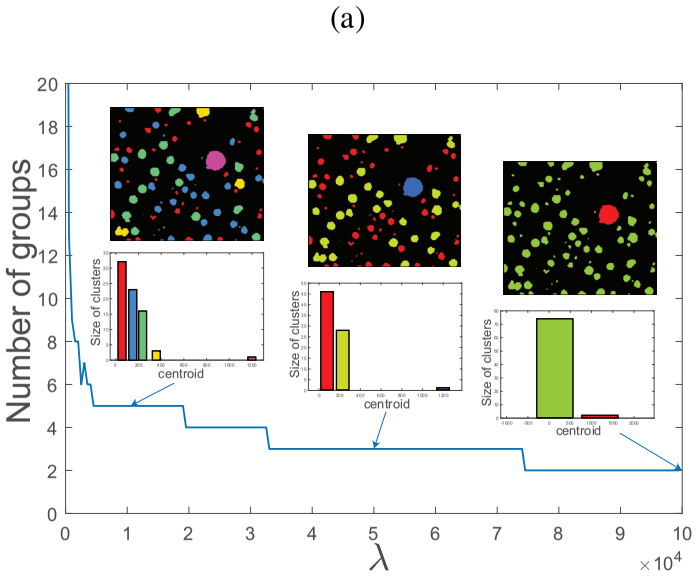
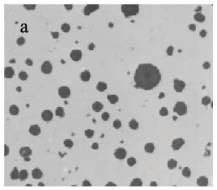


Figure 3.4: [Open boundary experiments] (a), (b) and (c) show three synthetic images where two square objects are separated with various size of gaps. An identical seed image U_0^1 is used for CODI-S and the first dimension of CODI-M. U_0^2 is used for second, and two random seed images for third and forth dimensions. The third and forth columns show CODI-S, and the fifth and sixth columns CODI-M after 40 and 80 iterations respectively. When the gaps between objects are wide and thin, it is helpful to have diffusion iteration small for CODI-S and large for CODI-M.





(a2) $\lambda = 1 \times 10^4$

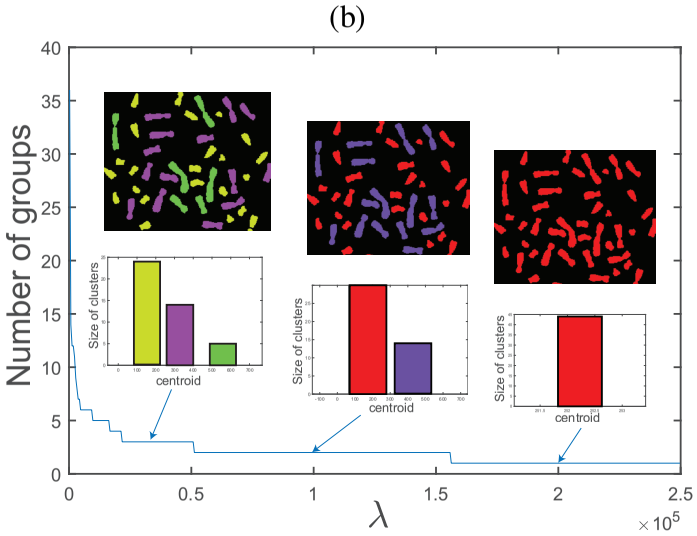
I_i	c_i	$\lvert I_i \rvert$
1	56.18	33
2	156.74	23
3	242.14	14
4	357.33	3
5	1199	1

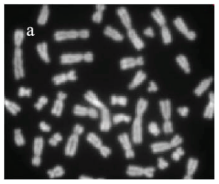
(a3) $\lambda = 5 \times 10^4$

I_i	c_i	$\lvert I_i \rvert$
1	78.82	45
2	229.25	28
3	1199.00	1

(a4) $\lambda = 1 \times 10^5$

I_i	c_i	$\lvert I_i \rvert$
1	135.9	73
2	1199	1





(b2) $\lambda = 3 \times 10^4$

I_i	c_i	$\lvert I_i \rvert$
1	153.17	24
2	327.71	14
3	558.00	5

(b3) $\lambda = 1 \times 10^5$

I_i	c_i	$\lvert I_i \rvert$
1	173.90	29
2	429.36	14

(b4) $\lambda = 2 \times 10^5$

I_i	c_i	$\lvert I_i \rvert$
1	257.07	43

Figure 3.5: [Counting similar size objects] (a1) and (b1) are given images from [135], and CODI-M found $K = 74$ and $K = 43$ cells respectively. (a) and (b) are graphs of λ vs. the number of groups. Three λ values for Regularized k-mean (Equation (3.6)) is picked from plateaus $\lambda = 1 \times 10^4, 5 \times 10^4, 1 \times 10^5$ for (a1) and $\lambda = 3 \times 10^4, 5 \times 10^4, 1 \times 10^5$ for (b1). Each λ shows different grouping depending on the size of objects from S. (a3) shows grouping to three different sizes, while (a4) shows grouping to two groups: one with one big object and another with all others. (b3)-(b5) also show different grouping possibilities.

larized k-mean energy is given by

$$E[k, \{I_i\}, \{c_i\}|S] = \lambda \left(\sum_{i=1}^{k} \frac{1}{n_i} \right) + \sum_{i=1}^{k} \sum_{|C_j| \in \{I_i\}} ||C_j| - c_i|^2, \tag{3.6}$$

which is minimized for given size of each cluster $|C_i|$. Here k is the number of groups found in the grouping process, $n_i = |I_i|$ is the number of objects that are contained in the group I_i, and $c_i = \{\frac{1}{k} \sum_{j=1}^{k} |C_j| : |C_j| \in I_i\}$ is the average object size in the group I_i. This l_i represents the group with similar size objects, and this similarity of the sizes are determined by the λ. This model automatically picks a reasonable number of cluster k with a parameter λ. A large λ gives fewer clusters while a small λ gives more number of clusters.

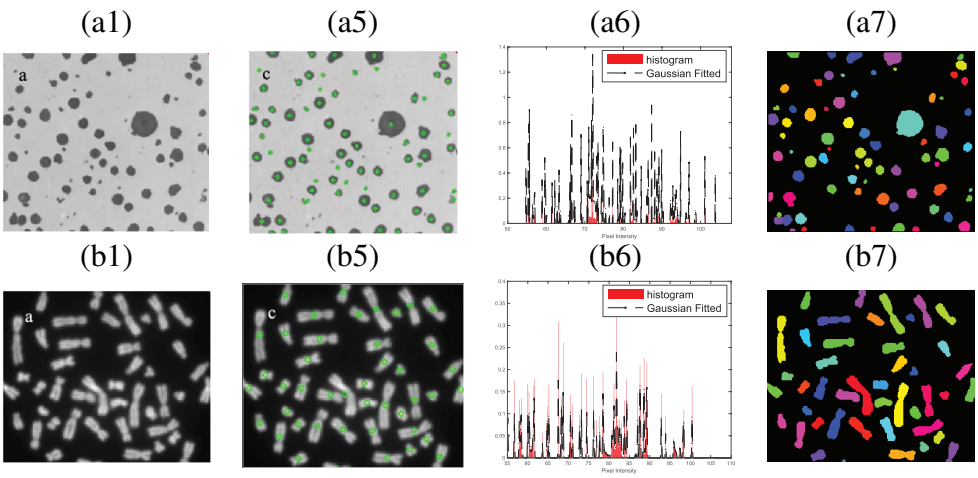
In Figure 3.5, (a1) is the given image from [135] where we used the edge function as $g(t) = \chi_{t<130}$ and $g(t) = \chi_{t>125}$, with $\chi_{t\in\Omega}(t) = \begin{cases} 1 & t \in \Omega \\ 0 & o.w. \end{cases}$ to threshold the given image. As a counting result, CODI-M identifies $K = 74$ objects. From the given image (a1) and its counting result $\{(x, U(x), C_i) : \quad U(x) \in C_i, \quad i = 0, 1, \ldots, K\}$, (a) shows a graph of λ vs. the number of groups. Notice that the Regularized k-mean (Equation (3.6)) has large plateaus showing the clustering result (the number of k) is not very sensitive against the choice of λ. We picked three λ values around different plateaus $\lambda = 1 \times 10^4, 5 \times 10^4$, 1×10^5 for (a1) and $\lambda = 3 \times 10^4, 5 \times 10^4, 1 \times 10^5$ for (b1). The colored image shows different size objects identified by different colors, and the histogram of S and tables below show more details. In each histogram, each bar denotes a group of different size objects. The horizontal axis – centroid size of each group – is the average size of objects in each group. The height of bars denote the number of objects that belongs to the same group. In the table (a2)-(a4), I_i shows how many different kinds of sizes are identified, c_i represents the average size in that group, and $|I_i|$ represents how many of such object exists in each group. For example in (a2), the table shows there are 5 different size of objects in image (a), with

32 number of the size around 54 objects, 23 of bigger objects of size 150, 16 of bigger ones of size 236, 3 of bigger ones of size 357, and one very big one of size 1199. As λ gets bigger the grouping gets simplified: (a3) separates objects to three, two of smaller sizes (46 of size around 78, and 28 number of size around 230), and one big one of size 1199. (a4) shows it can be also separated to two different sizes one big one and all other smaller of average size 135.

The sizes of cells in (b1) are similar size. Table (b2) shows that when using $3\times\lambda = 10^4$, 3 groups are formed, where the largest group has 24 objects of average size 327.71 pixels, and the smallest group contains 5 objects of size 558 pixels. As shown in table (b3), as 3λ increases to as large as 10^5, 2 groups are formed, where the larger group has 30 objects with averages size to be 169.57 pixels, which distinguishes the longer cells and shorter cells. When $\lambda = 2 \times 10^5$, all objects move into one single group of average size to be 252.33 as shown in table (b4). In conclusion, a smaller λ gives more groups and the plateaus of $k-\lambda$ curves in Regularized K-means provide meaningful justification about the number of groups of objects with respect to the distribution of size of objects in a given image.

3.4 Numerical Experiments and Comparisons

In this section, we demonstrate the effectiveness and efficiency of the proposed methods on various examples. All experiments are performed on MATLAB using Intel®Core i5-9600K processor with 3.7GHz 6Core CPU and 16 GB of RAM. In all experiments, we fix $\mu = 5\times10^{-5}$, $\theta = 1$, and $\eta = 0.0001$ in Diffusion Algorithm. In some cases, downsampling of original image is used for computational efficiency. An artificial outline is added on the boundary of the image domain Ω to prevent merging of objects near the boundary due to the effect of Fast Fourier transform. For CODI-S, we use a horizontal seed and for CODI-M we use a 4-dimensional seed involving one vertical, one horizontal, and two random seeds, as shown in Figure 3.2. Due to the two dimensions with random seeds, multiple tests are performed.



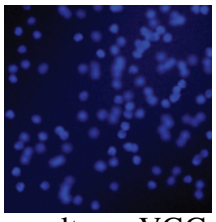
Given image	CODI-S	CODI-M	Existing result
(a1)	70 (1.68s)	74 (0.82s)	74 from [135]
(b1)	45 (1.87s)	43 (0.77s)	43 from [135]

Figure 3.6: [Cell counting] (a1) and (b1) are two cells images in Figure 3.5 from [135]. (a5) and (b5) are results from [135]. (a6) and (b6) are results of CODI-S. (a7) and (b7) are results of CODI-M. CODI-M experiments are performed 20 times, with the counting results between $[72, 74]$ for (a1) where 74 cells are found in 18 out of 20 trials. For (b1), the counting between 42 and 46 among 16 out of 20 trials. The average cpu time is 0.82 second and 0.77 second for (a1) and (b1) respectively. CODI is geometry-independent, and able to count cells of various sizes and shapes, very efficiently.

Cell counting: We experiment on cells images in Figure 3.5 (a1) and (b1). The counting results are illustrated in Figure 3.6. (a5) and (b5) show the results from [135]. (a6)-(a7) and (b6)-(b7) are results by CODI-S and CODI-M respectively. The method in [135] counted 74 cells in (a5) and 43 cells in (b5). The CODI-S count 70 cells in (a6) and 45 in (b6). The CODI-M count 73 cells in (a7) and 43 cells in (b7). For CODI-M, experiments are performed 20 times, and the counting results varies between $[72, 74]$ for (a1) where 74 cells are found in 18 out of 20 trials. For (b1), the counting result locates between 42 and 46 among 16 out of 20 trials. The average cpu time is 0.82 second and 0.77 second for (a1) and (b1) respectively. This shows that the CODI-M and CODI-S are both comparable to [135], and geometry-independent, and able to count cells of various sizes and shapes very efficiently.

Counting synthetic cell images: In Figure 3.7, we show the comparative counting results on the VGG-cell dataset [158], which is one of the public benchmark datasets. This dataset consists of images of size 256×256 with blue synthetic cells with blurry and overlapping boundaries. For CODI-S and CODI-M, we stop the diffusion iteration at ratio defined in (Equation (3.7)) to be 0.2 to 0.3 and 0.01 respectively. Since the data set has many overlapping or connected cells, for CODI-M, we further utilize the grouping strategy described in Section 3.3 for more accurate counting result. From the result of regularized k-means with $\lambda = 10,000$, consider the cluster for the smallest cells. We take the average size of objects in this cluster to represent the average cell size in the image. We multiply this value to the cluster with bigger cell to get a good counting result. We tested on 10 images (No.1, 48, 79, 84, 96, 127, 140, 155, 175, 196), and performed CODI-S once, and CODI-M 15 times (for random initial condition) for each image. The MAE of CODI-S is 1.78 with standard deviation 1.40. For g, we use the threshold $\chi_{t>137}$. We let $0.1 \leq \sigma \leq 1.2$ and $2 \leq r \leq 3$ for CODI-S and $\epsilon = 0.08$ and $\mathrm{MinPts} = 17$ for CODI-M method. The MAE of CODI-M of 150 experiments has average 1.73 with standard deviation 1.42. We compared proposed methods against some benchmark learning based methods in cell counting. The results of [158, 145, 144, 159, 146, 13, 147] are from [147] and the results of [160, 161, 148, 149, 155] are from [155].

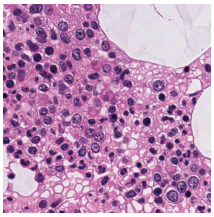
Counting cells in human bone marrow in MBM dataset: In Figure 3.8, we present statistical comparison results of proposed methods on MBM dataset. This dataset was first released by [162] and modified by [153] into a set of 44 images of 600×600 size with 126 ± 33 identified nuclei in each image. We tested on 10 images (No.4, 10, 14, 23, 26, 28, 29, 31, 32, 44) and performed CODI-S once, and CODI-M 15 times for each image. For g, we use the threshold $\chi_{t>150}$. We let $0.01 \leq \sigma \leq 0.7$ and $r = 3$ for CODI-S and $\epsilon = 0.7$ and $\mathrm{MinPts} = 10$ for CODI-M. The MAE of CODI-S is 3.10 with standard deviation 2.29. The MAE of CODI-M is 6.84 with standard deviation 4.22. We compare the proposed methods to some benchmark learning based methods [13, 12, 150, 146, 153, 147, 151, 152, 155] in



Comparison results on VGG Cell Dataset

Methods	MAE	Methods	MAE
Lempitsky's method [158]	3.52 ± 2.99	Artea's method [160]	4.5 ± 0.6
Mask R-CNN [145]	36.92 ± 19.73	Fiaschi's method [161]	3.2 ± 0.1
U-Net [144]	27.77 ± 25.48	Count-Ception [148]	2.3 ± 0.4
StructRegNet[159]	9.80 ± 8.68	Jiang's method [149]	2.2 ± 0.5
Cell-Net [146]	2.2 ± 0.5	Rodriguez's method[155]	2.2 ± 0.5
FCRN [13]	2.75 ± 2.47		
C-FCRN+Aux [147]	2.37 ± 2.27		
CODI-S	$\mathbf{1.78 \pm 1.40}$	CODI-M(5)	$\mathbf{1.86 \pm 1.59}$
		CODI-M(15)	$\mathbf{1.73 \pm 1.42}$

Figure 3.7: [VGG cell counting] Comparison results on cell images in VGG dataset. We let $0.1 \leq \sigma \leq 1.2$ and $r = 2$ in CODI-S, and $\epsilon = 0.08$ and $\mathrm{MinPts} = 17$ in CODI-M. We tested on 10 images (No.1, 48, 79, 84, 96, 127, 140, 155, 175, 196), and performed CODI-M 15 times and CODI-S once. Their corresponding mean and standard deviation of MAE are presented in the table. We observe that CODI is comparable to the existing methods without any training process.



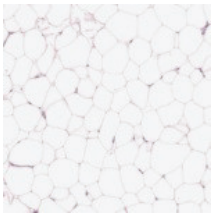
Comparison results on MBM Cell Dataset

Methods	MAE	Methods	MAE
FCRN-A [13]	21.3 ± 9.4	Marsden's method [12]	20.5 ± 3.5
Jiang's method [150]	14.5 ± 0.4	Cell-Net [146]	9.8 ± 3.2
CountCeption [153]	8.8 ± 3.2	He's method [147]	6.6 ± 5.3
Jiang's method [151]	6.0 ± 2.2	SAU-Net [152]	5.7 ± 1.2
Rodriguez's method [155]	4.2 ± 2.4		
CODI-S	$\mathbf{3.10 \pm 2.29}$	CODI-M(5)	6.58 ± 3.38
		CODI-M(15)	6.84 ± 4.22

Figure 3.8: [Counting human marrow MBM dataset] (a) a cell image from MBM-Cell dataset [153]. CODI-S and CODI-M are tested on 10 images (No.4, 10, 14, 23, 26, 28, 29, 31, 32, 44).

cell counting. Both CODI-S and CODI-M give relatively low counting loss, and CODI-S achieves the lowest MAE among all the methods.

Counting Non-isometric Cells in ADI dataset: In Figure 3.9, we show statistical comparison results on Adipocyte Cells. This dataset contains human subcutaneous adipose tissue and was originally released by [163]. We used the downsampled version from [153] where each image has size 150×150 and contains 165 ± 44.2 cells. The cells in this image have light boundaries with different sizes and shapes, which makes this dataset to be difficult to learn for some learning based methods [153, 152, 155, 149, 151], as shown in the table. For the proposed methods, we apply a simple histogram equalization follow by the mask g with the threshold $\chi_{t>70}$. We let $\sigma = 0.1, r = 2$ for CODI-S and $\epsilon = 0.1, \mathrm{MinPts} = 7$ for CODI-M. CODI-S and CODI-M are tested on 32 images, and performed CODI-S once and CODI-M for 15 time. The comparison results in Figure 3.9 are from [155]. It is shown that the proposed methods achieves the lowest MAE and they are robust in counting the isometric cells.



Comparison results on ADI Cell Dataset

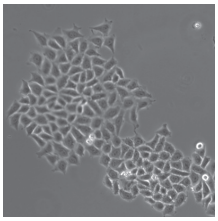
Methods	MAE	Methods	MAE
CountCeption[153]	19.4 ± 2.4	SAU-Net[152]	14.2 ± 1.6
Rodriguez's method [155]	17.3 ± 3.6	Jiang's method [149]	10.6 ± 0.3
Jiang's method [151]	10.1 ± 0.2		
CODI-S	$\mathbf{9.62 \pm 5.70}$	CODI-M(5)	$\mathbf{5.21 \pm 3.41}$
		CODI-M(15)	$\mathbf{5.03 \pm 3.45}$

Figure 3.9: [Counting non isometric cells - ADI dataset] (a) a cell image from ADI-Cell dataset [153]. CODI-S and CODI-M are tested on 32 images. Experiments are performed for 15 times for CODI-M.

Counting Hela Cells: In Figure 3.10, we present the statistical comparison results on Hela Cell Data set introduced in [164]. The Hela Cell images have a low percentage of overlapping cells where cells are separated by the bright edge boundaries.

Figure 3.10 compares the statistical results obtained by StructRegNet[159], Log [165], ITCN [166], IRV [167], FCRN-A, FCRN-B [13], CNN-SR [168], which were from [159]. In [169], a tree-structured discrete graphical model is used to classify non-overlapping regions by optimizing of a classification score. The detection is learned within the structured output SVM framework through dynamic programming on a tree structured region graph. In [127], the problem is formulated as a matching problem and the image self similarity property is used. Then a Generic Matching Network is trained using a few labeled examples. Figure 3.10 shows that both CODI-S and CODI-M methods are comparable to [169, 127, 165, 166, 167, 13] without any need of learning.

The statistical counting results on the whole Hela Data set, containing 11 test images, are given in Figure 3.10. The comparisons are made with Singletons [169], Full system w/o surface [169], and Class Agnostic method [127], which their data are taken from [127]. All these methods require training and learning procedure. The CODI-S and CODI-M



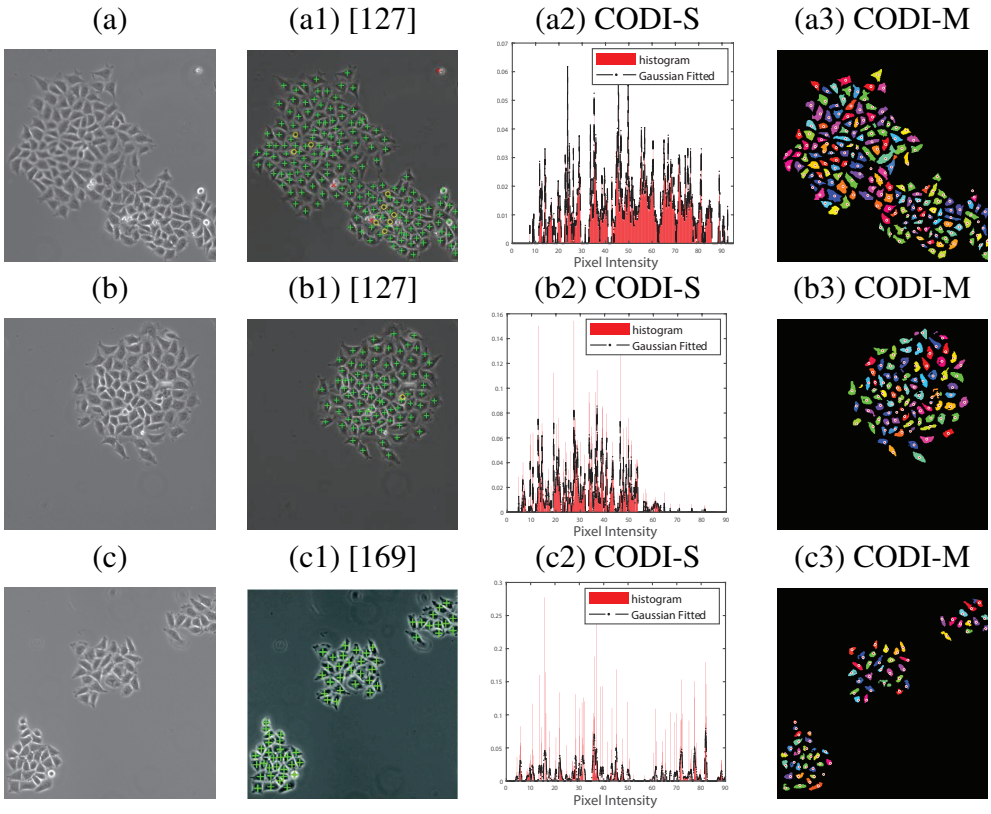
Comparison results on Hela Cell Dataset

Methods	MAE	Methods	MAE
Singletons [169]	2.35 ± 0.67	ITCN [166]	71.72 ± 41.63
Full system w/o surface [169]	3.84 ± 1.44	IRV [167]	54.36 ± 40.06
Class Agnostic [127]	3.53 ± 0.18	FCRN-A [13]	32.9 ± 21.9
StructRegNet[159]	$\mathbf{1.36 \pm 1.67}$	FCRN-B [13]	1.38 ± 1.91
Log [165]	20.82 ± 13.91	CNN-SR [168]	2.18 ± 3.82
CODI-S	2.36 ± 2.25	CODI-M(5)	3.33 ± 3.11

Figure 3.10: [Hela cell counting] Comparison results on 11 Hela images. We let $0.1 \leq \sigma \leq 2.7$ and $1 \leq r \leq 10$ in CODI-S, and $\epsilon = 1.5$ and $\mathrm{MinPts} = 20$ in CODI-M. CODI is comparable to the existing methods without any training process. CODI-M experiments are performed 5 times, and the mean and standard deviation of MAE are presented in the table. The results from FCRN, Log, ITCN, IRV , CNN-SR, and NERS are adopted from [159].

do not require any training thus to obtain some statistics, we exploit CODI-S once and CODI-M five times on each image in the training data set, containing 11 images. For g, we implemented a threshold with $\chi_{t\leq 100}$, contrast enhancement [170], a threshold with $\chi_{t\leq 70}$, and a dilation step with structuring element parameters to be (disk,1,4). We let $0.1 \leq \sigma \leq 2.7$ and $1 \leq r \leq 10$ for CODI-S and $\epsilon = 1.5$ and $\mathrm{MinPts} = 20$ for CODI-M method.

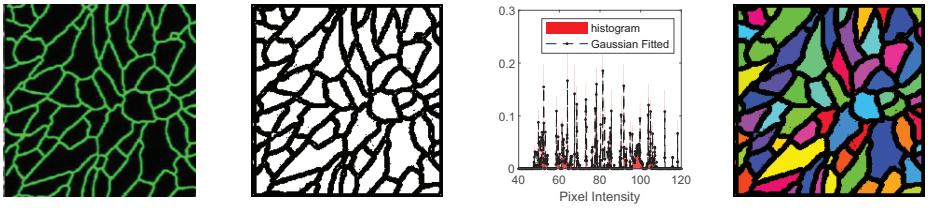
For numerical comparison measures, we use Mean Average Error (MAE) = $\frac{1}{n}\sum_{i=1}^{n} |y - y^*|$ for the number of objects. Here y^* represents the ground truth counting number, y is the computed number, and n is the number of images in the test set. Note that a lower MAE is preferable. In Figure 3.10, we observe that CODI is comparable to the exsiting method, but without any training. CODI-M is also able to track the objects location in the image. Additional visualization results between the proposed methods, Class Agnostic method [127], and Singletons [169] are shown in Figure 3.11.



Hela Cells	Size	Ground truth	CODI-S	CODI-M	Others
(a)	400×400	177	177 (0.95s)	175 (5.34s)	171 [127]
(b)	400×400	85	85 (2.06s)	88 (3.07s)	84 [127]
(c)	400×400	67	65 (1.95s)	68 (1.39s)	67 [169]

Figure 3.11: [Hela cell counting] Hela cell images from [164]. CODI-S and CODI-M give comparable counting results to [127] and [169]. In the parenthesis, we show the CPU times for each computation. CODI-M experiments are performed 5 times, and the best results are presented here, while all comparisons are given in Table Figure 3.10.

(a) Given image (b) Edge function (c) CODI-S (d) CODI-M



Given image	Manual Count	CODI-S	CODI-M
(a)	[70,72]	72 (2.16s)	70 (2.81s)

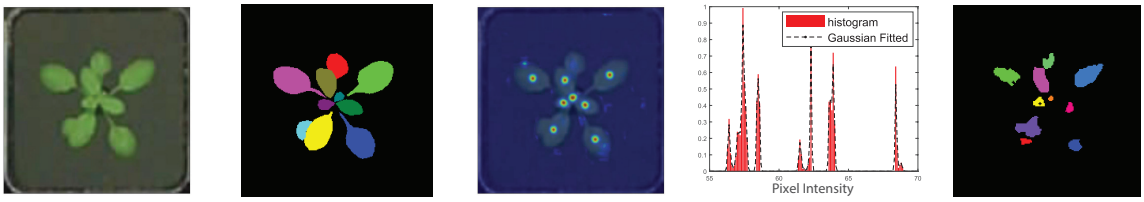
Figure 3.12: [Counting seamless leaf patterns] (a) Given image of where manual counting is given between 70 and 72. (b) The edge function $\tilde{g}$. (c) CODI-S counts 72 leafs. For 20 CODI-M experiments, the counts varies between [68,70] and 13 out of 20 trials results in count 70. The subtle uncertainty comes from the small objects in the original image. The average cpu time is 2.81 second. Figure (d) shows one representative result of CODI-M.

Counting seamless leaf patterns: We consider a seamless leaf image with lace veins patterns, in Figure 3.12 for visualization. The manual counting give the number between $[70, 72]$. For g, we use the edge detecting function $\bar{g}$ in Equation (3.1) where $\bar{g} > 0.7$ is considered as 1 as the binary output. In this example, CODI-S and CODI-M find 72 and 70 objects respectively.

Arabidopsis plant leafs: We consider an image of Arabidopsis plant from MSU-PID dataset [171] shown in Figure 3.13 (a). In the ground truth image in (b) shows 10 leafs. We compare our methods with [129], a Domain-Adversarial Neural Network (DANN) where the counting is done by the density map estimation shown in Figure 3.13(c). For g, we used a threshold with $\chi_{t>137}$. To further separate the edges between the overlapping leaves, an edge detecting function $\bar{g}$ in (Equation (3.1)) where $\bar{g} > 0.8$ is considered as 1 as the binary output. It finds 8 leafs, while CODI-S and CODI-M find 9 and 10 leafs, respectively.

Agriculture and fruits: We consider agriculture images in Figure 3.14: (a) an apple tree (594×800), (b) a bunch of cherries (800×800), (c) a sparse apple tree (585×768) from [172], and (d) a tomato plant (214×181). These are color images where the fruits are red or orange and the rest of image is roughly green. For g, we subtracted the green

(a) Given image (b) Ground truth (c) [129] (d) CODI-S (e) CODI-M



Given image	Manual Count	CODI-S	CODI-M	Other
from [171]	10	9 (1.33s)	10 (0.07s)	8 [129]

Figure 3.13: [Arabidopsis plant leaf counting] (a) Given image of Arabidopsis plant [171]. (b) The ground truth image in [171] showing 10 leafs. (c) The density map estimation [129], showing 8 leafs. (d) CODI-S counts 9 leafs. (e) CODI-M counts 10 leafs. Experiments are performed 20 times on CODI-M, where 10 out of 20 trials results in count between 9 and 11. The subtle uncertainty comes from the delicate boundaries between the leaves in the original image. The average cpu time is 0.07 second. Figure (e) shows the best results among 20 CODI-M experiments.

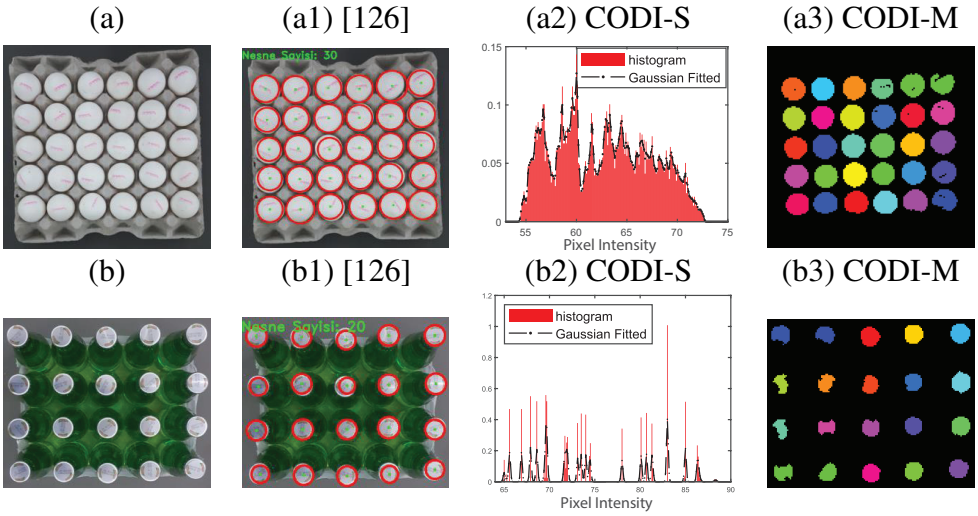
channel (the second dimension) from the red channel (the first dimenstion) followed by a thresholding $\chi_{t>80}$, $\chi_{t>110}$, $\chi_{t>130}$, $\chi_{t>110}$ for (a)-(d) respectively. Since there are many overlapping objects, a rough estimate of manual counts are provided in form of intervals. We apply the proposed methods to count the number of apples in (a),(c), cherries in (b) and tomatoes in (d). For (c) and (d), we compare the proposed methods with some learning methods [172] and [173] respectively. Figure 3.14 shows that the proposed methods are able to find a correct estimation for the number of fruits.

Objects in the production line: The production line images are considered in Figure 3.15: (a) a cart of eggs and (b) a case of soda bottles. We compare CODI-M and CODI-S with the method in [126]. In [126], the authors considered the segmentation, Gaussian filter, Otsu Thresholding [174], Sobel Edge Detection [175, 176], and Hough Circle Transform [177, 178]. For g, we used a threshold $\chi_{t>205}$ for (a), $\chi_{t>120}$ for (b) and an erosion step on (b) with structuring element parameters to be (disk,1,4) to further distinguish the boundary. Due to the use of Hough Circle Transform, the work [126] is geometry dependent. Figure 3.15 shows CODI is comparable while being geometry-free.

(a) (b) (c) (d)

Given image	Manual Count / Group Truth	CODI-S	CODI-M	[172]	[173]
(a)	[205, 235]	202 (7.57s)	[217, 226] (4.85s)		
(b)	[27,33]	31 (0.25s)	[27,33] (0.07s)		
(c) [172]	10	9 (1.85s)	[8,10] (0.19s)	9	
(d) [173]	19	19. (2.69s)	[16,22] (0.23s)		16

Figure 3.14: [Counting fruits] (a) An apple tree (b) A bunch of cherries. (c) an apple tree image from Five-Tropical-Fruits dataset [172]. (d) a tomato image from [173] Both CODI-S and CODI-M find a number within the accepted range. Experiments are performed 20 times on CODI-M. For (a), the results varies in [208, 226], where 14 out of 20 trials generate results in [217, 226]. For (b) the result varies between [25,40] where 14 out of 20 experiments results in [27,33]. This result is consistent with the large quantity of apples in (a) and the unclear boundaries between cherries in (b). The average cpu time is 4.85 and 0.07 for each image respectively. For (c), the result from CODI-M varies between [8,10] where 7 out of 20 experiments results in 9 or 10. For (d), the result from CODI-M caries between [16,22] where 9 out of 20 experiments result in [18,20].



Given image	CODI-S	CODI-M	Others
(a)	29 (0.38s)	30 (0.28s)	30 [126]
(b)	20 (1.08s)	19 (0.10s)	20 [126]

Figure 3.15: [Counting objects in the production lines] (a) A cart of eggs. (b) A case of soda bottles. The second column shows results by CODI-S, the third column by CODI-M, and the forth column by [126]. Experiments are performed 20 times on CODI-M. For (a), the results varies in [29, 31], where 18 out of 20 trials generate 30 as counting result. For (b) the result varies in [18, 23] where 18 out of 20 experiments generate results between [18, 20]. CODI gives comparable results to [126] without exploiting any geometrical information.

(a) (b) (c)



Given image	Manual count	CODI-S	CODI-M
(a)	292 ± 10	286 (6.48s)	[285,302] (6.29s)
(b)	$94^{\dagger}$	94 (3.22s)	[93,95] (3.26s)
(c)	17 ± 1	17 (0.33s)	[14,17] (1.10s)

Figure 3.16: (a) Concert crowd image (b) GPS image from DOTA dataset [179, 180]. (c) Penguin image from [181]. An estimated number of people and vehicles are obtained by manual counting. Experiments are performed 20 times on CODI-M. For (a), the results varies in [283, 315], where 13 out of 20 trials generate result in [285,302]. For (b) the result varies in [93,96] where 14 out of 20 experiments generate results between [93,95]. The subtle unstable of the result for (a) is due to the large quantity of people in the original image. † The ground truth of 94 is provided in the dataset. For (c) the result varies within the range [14,17], and more than 50% of experiments gives 16 or 17 counts.

Crowd , Vehicle, and Wild animal: Figure 3.16 displays (a) an image of concert crowd, (b) a GPS image from DOTA dataset [179, 180], and (c) a wild animal penguin image from dataset in [181]. An estimated number of people, vehicles, and penguins are obtained by manual counts given in Figure 3.16. For g, we used $\chi_{t<155}$ in (a), $\chi_{t>220}$ followed by a dilation step with structuring element parameters to be (disk,1,4), and $\chi_{t>220}$ in (c). We observe that the proposed CODI-S and CODI-M methods give good estimation of the counts.

In the following, we present a few aspects of CODI. First, to ensure the quality of diffused index, we present ideas to properly choose the seed location and size. Then, we present the effect of the downsampling of original image, and finally comment on the choice of parameter in computation.

Since CODI counts the diffused index, it is helpful to have the indexes to be as separated as possible. We propose the following simple rules on the distance between seeds and size

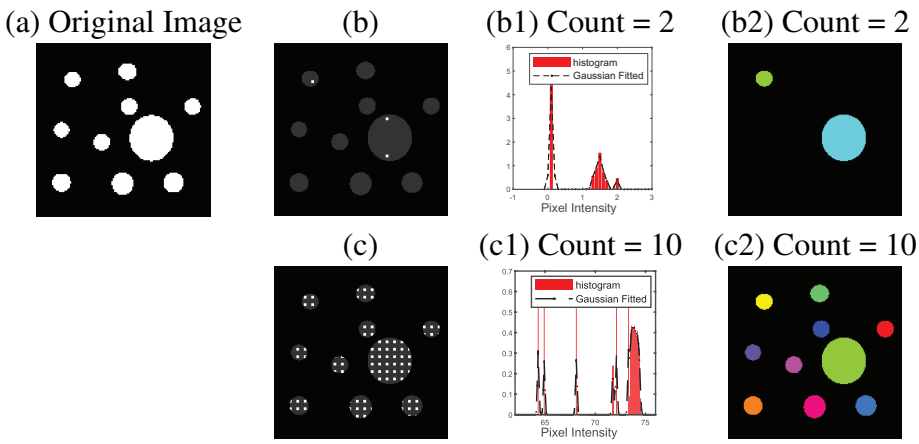


Figure 3.17: [Seed sparsity/distance] (a) The given image. (b) and (c) are two different seed images. If there are objects without any seeds inside, CODI misses counting these objects as expected as in (b1) and (b2). With multiple seeds within all objects, both methods count correctly as in (c1) and (c2). This illustrates the importance of having the distance between seeds to be smaller than the minimum distance between objects.

of seeds, for better performance of CODI:

1. The distance between (the boundary of) seeds should be smaller than the minimum distance between the boundary of objects, that every object has at least one seed inside.

2. The size of seed itself should be small compared to the minimum size of objects, that no two objects are covered by only one unique seed. In addition, we found that the convergence is faster with smaller seed size.

Figure 3.17 shows the effect of counting results based on different sparsity of seeds. (a) is a synthetic image of size 126×127, and experiments are preformed based on two seed images (b) and (c), with two different distance between seed boundaries $d = 38$ and $d = 6$ respectively. The size of seeds are both 2×2. (b1)-(b2) and (c1)-(c2) provide the counting results form CODI-S and CODI-M respectively. If there are objects without any seeds inside, CODI misses counting these objects as expected, shown in (b1) and (b2). As a comparison, both proposed methods count 10 objects in (c1) and (c2), if there are multiple seeds within all objects to be counted. This illustrates the importance of Rule 1 that it

is important to have the distance between seeds to be smaller than the minimum distance between objects.

As for the size of the seeds, if multiple objects have only one large seed shared, it will be identified as one object in CODI. Rule 2 suggest each seeds to be small compared to the minimum size of the objects to ensure separation between different objects. We further experiment with the size of seed in Figure 3.18. It shows that even if the size of the seed is smaller than the size of the objects to be counted, it is better to have smaller seeds for faster diffusion. We experiment on a binary image of size 281×87 with 6 hexagons using seeds of size 20×20 and 2×2 respectively. The distance between the boundaries of big seeds and small seeds are both 10 pixels, which is smaller than the minimum distance between any two hexagons to be counted. The first and second rows show CODI-S and CODI-M using big seeds, while the third and fourth rows show CODI-S and CODI-M using small seeds respectively. With smaller seeds, less than 40 iteration for both CODI-S and CODI-M give correct counting of 6, while for bigger seeds (top two rows) takes 300 to 400 iteration to find the correct counting. To demonstrate the relation between seed size and convergence, we set the objective function in (Section 3.2.2) at nth iteration to be E_n, and consider

$$R_n = \left| \frac{E_n - E_{n-1}}{E_{n-1}} \right| \tag{3.7}$$

for convergence measure. If R_n is small, it means the diffusion is converging. For each experiments, the clustering results are shown in 3 stages: first column: $R_n = 0.09$, second column: $R_n = 0.05$, and third column: $R_n = 0.01$. In Figure 3.18 the third row, after 32 iterations, $R_n = 0.09$ in CODI-S, 6 objects are found. After another 9 iterations, R_n decreased to 0.05 and 6 objects are found by CODI-S again. This shows that using relatively small seeds results in good counting results with $R_n = 0.05 - 0.09$. For bigger seeds $R_n = 0.01$ is needed, since changing given seed values to become a diffused index for each object takes longer.

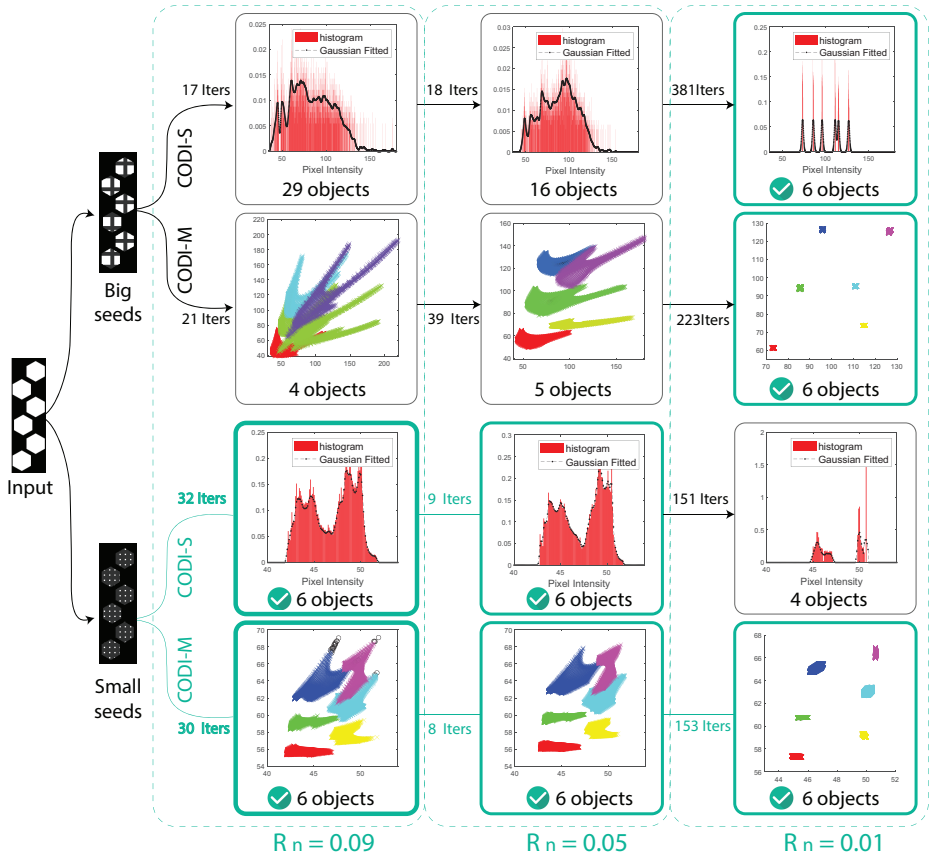


Figure 3.18: [Seed size v.s. Convergence] From one given image, two different sizes of seeds are used while keeping the distance between the seeds to be the same (smaller than the minimum distance between the objects). For smaller seeds in third and forth row, CODI gives good counting results with $R_n = 0.05 - 0.09$. For bigger seeds $R_n = 0.01$ is needed, since changing given seed values to become a diffused index for each object takes.

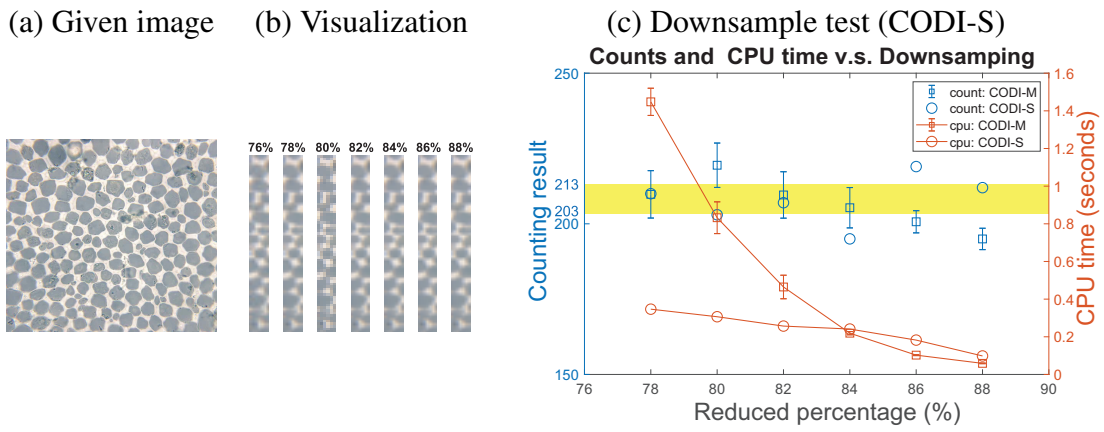


Figure 3.19: [Downsample and cpu time] (a) The given image of 1000×1097 with manual counting in the range of $[203, 213]$ which is shown as the highlighted region in (c). (b) a visualization of seven different downsampled image, ranging from 76% to 88%. (c) The blue circles represents CODI-S, the red circles the cpu time. The blue error bars denotes the mean and standard deviation of 50 experiments of CODI-M, and the red error bars those of cpu times. Notice while the counting results are near the correct range, cpu time clearly reduces with downsampling.

Given an image of high resolution, reducing the size of image while keeping the boundary information can significantly reduce the cpu time. Figure 3.19 shows reduction of size vs the counting result. (a) is the original image of size 1000×1097. With manual counting, there are about $[203, 213]$ number of cells, depending on how very small objects are counted. This image is reduced to 7 different levels of quality as shown in (b), level of reduction ranging from 76% to 88% reduced from the original image. For example, after 88% reduction, the given has been reduced to size 140×154. For each of the seven reduced image in (b), we perform CODI-S for once and CODI-M for 50 times. In (c), blue dots are CODI-S, blue bars are CODI-M, and the yellow color bar is a range of correct counting. Red bar graphs show the CPU time in seconds for CODI-S and CODI-M showing the clear reduction on cpu time. The x-axis shows the downsampling rate. Notice while the counting results are near the correct range, cpu time clearly reduces with downsampling.

As for the stability of parameters for CODI-S and CODI-M, we consider the parameter space in terms of r and σ for CODI-S, and in terms of MinPts and ϵ for CODI-M. We test

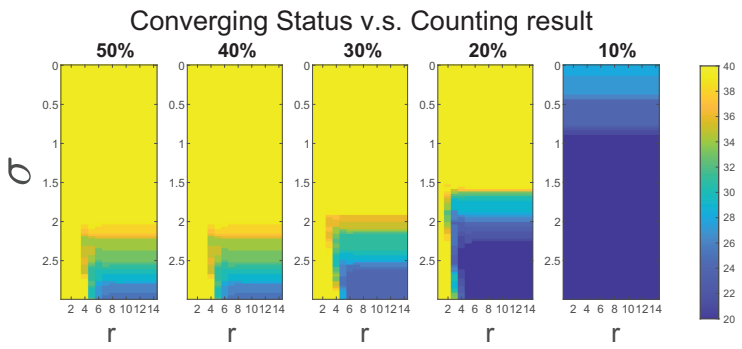


Figure 3.20: [CODI-S parameter space] Visualization of countingresult in the parameter space $(r, \sigma) \in [2, 15] \times [0.01, 3] \cup [1.6, 3]$ based on different diffusion stage. (a)-(e) shows when $R_n = 50\%, 40\%, 30\%, 20\%, 10\%$. The ground truth of counting result is 30, where the more yellow the color is more accurate the result. This result is consistent with Figure 3.4 where smaller number of iteration is favorable for CODI-S.

with Figure 3.15(a) image. In Figure 3.20 and Figure 3.21, the most yellow region denotes the parameter set that produce 100% correct counting results. We present the parameter graph as the diffusion algorithm convergence. We consider R_n in (Equation (3.7)) for convergence measure. If R_n is small, it means the diffusion is converging. In Figure 3.20, we show five experiments (a)-(e) where $R_n \in \{50\%, 40\%, 30\%, 20\%, 10\%\}$. The ground truth of counting result is 30. When $R_n = 10\%$, there are larger green region in the parameter space that produces high accuracy. These graphs also present the relation between the smoothing of histogram, the number of iteration and the counting results. In general, there are large regions with yellow which represent good counting results. This result is consistent with Figure 3.4 where smaller number of iteration is favorable for CODI-S. In Figure 3.21, the same experiments are conducted for CODI-M where we have R_n set to be $15\%, 10\%, 5\%$ in (a)-(c). The red marks denotes two examples of the optimal parameters we recommend for the experiment in similar cases. When $R_n \leq 10\%$, the counting result won't be affected by small perturbation of the parameters. As in the case of open boundary, CODI-M with longer iteration give stable results.

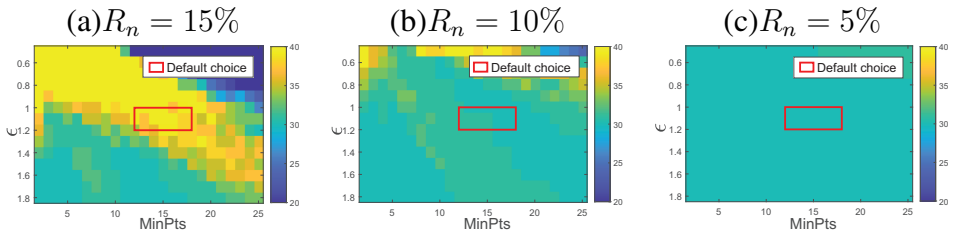


Figure 3.21: [CODI-M parameter space] Visualization of counting result in the parameter space $(\text{MinPts}, \epsilon) \in [2, 25] \times [0.5, 1.8]$ based on different diffusion stage. The ground truth in this example is 30, i.e., the green area represents good result. (a)-(c) shows when R_n to be $15\%, 10\%$, and 5%. The red marks denotes the parameters we recommend for similar cases. With enough iterations, the counting result of CODI-M is not affected by a small perturbation of the parameters.

CHAPTER 4

WEAKIDENT - IDENTIFYING DIFFERENTIAL EQUATIONS PART I (PHYSICAL DOMAIN)

This chapter reproduces our previously published paper [3]. The author of this thesis personally contributed to Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Visualization, and Writing – original draft.

In this chapter, we propose a Weak formulation for Identification of Differential Equations with Narrow-fit and Trimming (WeakIDENT). To recover (Equation 4.1) where f is a linear combination of various differential terms, we construct a linear system: the feature matrix consisting of linear and nonlinear terms called features, multiplied by a coefficient vector, is set equal to the time derivative. We use the term coefficient support to refer to a collection of nonzero components in the coefficient vector, such that the linear system is composed of the collection of features that contribute to the dynamics represented by the data. For the weak formulation, we follow the derivation proposed in [36]. For our sparse coefficient recovery, we perform an iterative greedy support identification scheme as in [30] to find the support which gives the collection of linear and nonlinear differential terms. For each sparsity level, we use the Subspace Pursuit (SP) algorithm [182] to first find the initial guess of the coefficient support. We propose new narrow-fit and trimming steps which improve the support selection as well as coefficient value recovery. Among different sparsity results, we choose the one with the minimum Cross-Validation (CV) error as the final result. For Cross-Validation, we randomly separate the given data in half, use one set to find the coefficients, then use this coefficient vector with the other set of data to compute the error. We provide an error analysis in Theorem 4.2.1 to show that the error in the linear system under the weak form is significantly smaller than that under the differential form.

Our contributions can be summarized as follows:

1. Proposing WeakIDENT to robustly identify differential equations in (Equation 4.1) from highly corrupted noisy data. The weak form proposed in [36] allows us to move the derivative to the test functions and facilitates robustness against noise. We propose two new and novel mechanisms, narrow-fit and trimming, to improve the coefficient support value and the coefficient support recovery, respectively. These mechanisms utilize a column-wise error normalization to improve the robustness of the coefficient recovery. Narrow-fit focuses on highly dynamic regions to reduce the size of the feature matrix, and trimming the features with small contributions to the result further contributes to the improvement.

2. We provide comprehensive numerical experiments for ordinary differential systems (ODEs) and partial differential equations (PDEs), and compare with existing methods such as [29, 30, 35, 36, 37].

4.1 Literature Review

In this chapter, we focus on the inverse problem of identifying a differential equation corresponding to given data corrupted by noise. Given a time-dependent discrete data set, we aim to discover the underlying equation of the form

$$\partial_t u = f(u, \partial_{\boldsymbol{x}} u, \ldots, \partial_{\boldsymbol{x}}^k u, \ldots, u^2, \partial_{\boldsymbol{x}} u^2, ..., \partial_{\boldsymbol{x}}^k u^2, \ldots, u^3, \partial_{\boldsymbol{x}} u^3, ..., \partial_{\boldsymbol{x}}^k u^3, \ldots) \tag{4.1}$$

where each differential term in the right hand side of (Equation 4.1) is called a feature in this prescribed dictionary. In particular, f is called the governing equation of (Equation 4.1). We assume that f in (Equation 4.1) is a linear combination of the features, so that this inverse problem becomes the identification of a sparse coefficient vector where both the support and the values of this coefficient vector are unknown. Since the features include linear and nonlinear terms, this f in (Equation 4.1) includes nonlinear differential equations. This model identification problem is very challenging when the given data are corrupted

by noise.

Parameter identification in differential equations and dynamical systems has been studied by scientists in various fields. Earlier works include [19, 20, 183, 16, 21, 17], where the differential equation (Equation 4.1) is considered in [20, 21], symbolic regression is used in [16, 17], and an optimization approach is taken in [19, 20, 21]. In recent years, sparse regression is incorporated into the model identification problem to promote sparsity in the coefficient recovery [22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37]. Representative works include Sparse Identification of Nonlinear Dynamics (SINDy)[22, 23, 24, 25], Identifying Differential Equations with Numerical Time evolution (IDENT)[29, 30], Weak SINDy [36, 37], RGG [38] and many others[31, 184, 32, 185]. The PDE and dynamics identification problem is also addressed by deep learning approaches [186, 187, 188, 189, 190, 191, 192].

The majority of existing works apply sparse regression on a linear system formed from (Equation 4.1) with differential features [22, 23, 24, 25, 29, 30, 31, 33]. From the given data, differential features are approximated via numerical differentiation. When the given data contain noise, a denoising step is applied before numerical differentiation. Least-squares moving average is applied in [29], successively denoised differentiation is proposed in [30] and regularization is used in [193]. In terms of sparse regression, L_1 or regularized L_1 minimization has been widely used [22, 29, 31, 184]; Sequentially thresholded least-squares is used in [24, 26, 27, 28]; Greedy algorithms are used in [30]. More generally, the coefficients are allowed to be spatially dependent in [194, 29], and the Group Lasso is used to promote group sparsity where each group represents a feature, which is also used for varying coefficient case in [29]. While these methods using differential features give good results, numerical differentiation can be unstable for high-order features, and the coefficient recovery may not be robust when the given data is corrupted by noise.

Recent progress using a weak/integral formulation [35, 38, 36, 37] shows improvements in the robustness of the sparse coefficient identification. A weak form for (Equation 4.1)

with a set of test functions gives rise to a linear system with integral features instead of differential features. Noise is tackled through the weak form, since a proper test function gives a denoising effect. The test functions are chosen to be localized smooth functions vanishing on the boundaries, thus resembling kernel functions commonly used in kernel denoising methods. It is shown in [36, 37] that using the weak form with the standard sequentially threshold least-squares algorithm gives rise to superior numerical performance. Differential equations with high-order derivatives, including the Korteweg–De Vries (KdV) equation, the Kuramoto–Sivashinsky (KS) equation, and 2D reaction-diffusion equations can be recovered even with a significant amount of noise. A related work [195] focuses on the identification of advection-diffusion equations, and shows that a Galerkin-type algorithm using the weak form outperforms the collocation-type algorithm using a differential form.

4.2 Modeling dynamics and Weak Formulation

In this section, we illustrate how to formulate a linear system in a weak form and state the identification problem for differential equations. We also discuss the choice of test functions and provide an error analysis of the weak formulation.

4.2.1 Formulating dynamics using differential equations

We present the identification problem with one spatial variable for simplicity. It can be easily extended to multi-variables, and numerical results are provided for the multi-variable case. We consider a spatial-temporal domain $\Omega = [X_1, X_2] \times [0, T]$ with $X_1 < X_2$ and $T > 0$. We assume a set of discrete time-dependent noisy data is given:

$$\mathcal{D} = \{\hat{U}_i^n | i = 1, 2, ..., \mathbb{N}_x; n = 1, ..., \mathbb{N}_t\} \in \mathbb{R}^{\mathbb{N}_x \times \mathbb{N}_t}, \tag{4.2}$$

where $\mathbb{N}_x$, and $\mathbb{N}_t \in \mathbb{N}$ are the size of discretization in spatial and temporal dimension respectively. The data point $\hat{U}_i^n$ is an approximation to the true solution of a differential equation

$$\hat{U}_i^n \approx u(x_i, t^n) \text{ for } (x_i, t^n) \in \Omega,$$

at the spatial location $x_i = i\Delta x \in [X_1, X_2]$ and $t^n = n\Delta t \in [0, T]$. Here $\Delta x = (X_2 - X_1)/(\mathbb{N}_x - 1)$ and $\Delta t = T/(\mathbb{N}_t - 1)$. In the noisy case, we express the noisy data $\hat{U}_i^n$ in terms of the clean data $U_i^n = u(x_i, t^n)$ as:

$$\hat{U}_i^n = U_i^n + \epsilon_i^n, \tag{4.3}$$

where $\epsilon_{i,}^n$ represents the noise at (x_i, t^n). The objective is to identify a differential equation in the form of (Equation 4.1) from the given data (Equation 4.2).

We assume that the governing equation f in (Equation 4.1) is a linear combination of linear and nonlinear terms including the derivatives of u. This covers a vast range of ODEs and PDEs in applications, e.g., the Lorenz equation, the Lotka-Volterra equation, transport equations, Burgers' equation, the heat equation, the KS equation, the KdV equation, and reaction-diffusion equations. In this chapter, to utilize the weak form, we consider the function f to be a linear combination of different derivatives of powers of u:

$$\frac{\partial u}{\partial t}(x,t) = \sum_{l=1}^{L} c_l F_l \quad \text{with} \quad F_l = \frac{\partial^{\alpha_l}}{\partial x^{\alpha_l}} f_l, \text{ where } f_l = f_l(u) = u^{\beta_l}. \tag{4.4}$$

The l^{th} feature $F_l(u)$ represents the α_l^{th} spatial derivative of the monomial $f_l = f_l(u) = u^{\beta_l}$ for some nonnegative integer β_l. Let the highest order of derivative be $\bar{\alpha}$ such that $\alpha_l \in \{0, \ldots, \bar{\alpha}\}$, and the highest order of monomial be $\bar{\beta}$ such that $\beta_l \in \{0, \ldots, \bar{\beta}\}$. We use L to denote the total number of features in the dictionary, which depends on $\bar{\alpha}$ and $\bar{\beta}$, since it includes all combinations. The formulation of (Equation 4.4) has the advantage in accurate feature approximation particularly for the weak form, since integration by parts moves the

derivatives to the test function. When the spatial domain is multi-dimensional, we consider f_l as monomials in the multivariable case, and we allow F_l to be partial derivatives of f_l across different spatial dimensions.

In (Equation 4.4), the coefficient can be considered as a sparse vector

$$\boldsymbol{c} = (c_1, ..., c_L)^T \in \mathbb{R}^L \tag{4.5}$$

which parametrizes the differential equation. The objective of this chapter is to recover the differential equation from the given noisy data set $\mathcal{D}$ (Equation 4.2), by finding a sparse coefficient vector $\boldsymbol{c}$ (Equation 4.5) of the linear system (Equation 4.1).

4.2.2 The Weak Formulation

The weak formulation of (Equation 4.4) is

$$\int_{\Omega_{h(x_i,t^n)}} \phi_{h(x_i,t^n)}(x,t)\frac{\partial u(x,t)}{\partial t}dxdt = \sum_{l=1}^{L} c_l \int_{\Omega_{h(x_i,t^n)}} \phi_{h(x_i,t^n)}(x,t)F_l dxdt, \tag{4.6}$$

where the test function $\phi_h(x,t)$ is locally defined on a region $\Omega_{h(x_i,t^n)}$, which is centered at (x_i, t^n) and indexed by h. Specifically, each test function $\phi_h(x,t)$ is a translation of a fixed function $\phi(x,t)$ such that $\phi_{h(x_i,t^n)}(x,t) = \phi(x - x_i, t - t^n)$. Integration by parts of (Equation 4.6) gives rise to

$$-\int_{\Omega_{h(x_i,t^n)}} u(x,t)\frac{\partial \phi_h(x,t)}{\partial t}dxdt = \sum_{l=1}^{L} c_l \int_{\Omega_{h(x_i,t^n)}} (-1)^{\alpha_l} u^{\beta_l}\frac{\partial^{\alpha_l}\phi_h}{\partial x^{\alpha_l}}dxdt, \tag{4.7}$$

as long as ϕ_h and its derivatives up to order $\bar{\alpha}$ vanish on the boundary of $\Omega_{h(x_i,t^n)}$. The l^{th} term

$$\int_{\Omega_{h(x_i,t^n)}} (-1)^{\alpha_l} u^{\beta_l}\frac{\partial^{\alpha_l}\phi_h(x,t)}{\partial x^{\alpha_l}}dxdt$$

is the l^{th} integral feature with the test function ϕ_h. Since the test function is smooth, the numerical integration can be carried out with higher order accuracy. With numerical integration, we obtain the following discrete linear system for WeakIdent:

$$\boldsymbol{W}\boldsymbol{c} = \boldsymbol{b} \tag{4.8}$$

where

$$\boldsymbol{W} = (w_{h(x_i,t^n),l}) \in \mathbb{R}^{H\times L}, \quad \boldsymbol{c} = (c_l) \in \mathbb{R}^L, \quad \text{and} \quad \boldsymbol{b} = (b_{h(x_i,t^n)}) \in \mathbb{R}^H,$$

for

$$w_{h(x_i,t^n),l} = \sum_{(x_j,t^k)\in\Omega_{h(x_i,t^n)}} (-1)^{\alpha_l}\hat{U}_j^k \frac{\partial^{\alpha_l}}{\partial x^{\alpha_l}}\phi_h(x_j,t^k)\Delta x\Delta t$$

and

$$b_{h_(x_i,t^n)} = -\sum_{(x_j,t^k)\in\Omega_{h(x_i,t^n)}} \hat{U}_j^k \frac{\partial\phi_h(x_j,t^k)}{\partial t}\Delta x\Delta t. \tag{4.9}$$

Here the numerical integration is computed with the data points $(x_j,t^k) \in \Omega_{h(x_i,t^n)}$, and $w_{h(x_i,t^n),l}$ represents an approximation of the integral of the feature F_l in the integral region $\Omega_{h(x_i,t^n)}$ centered at (x_i,t^n). The numerical integration is computed from $\mathbb{N}_x\mathbb{N}_t$ grid points.

For the test function, we follow the derivation and use $\phi(x,t)$ as in [36]:

$$\phi(x,t) = \left(1-\left(\frac{x}{m_x\Delta x}\right)^2\right)^{p_x}\left(1-\left(\frac{t}{m^t\Delta t}\right)^2\right)^{p_t}, \quad (x,t)\in\Omega_{h(x_i,t^n)} \tag{4.10}$$

for $i = 1,..,\mathbb{N}_x, n = 1,...,\mathbb{N}_t$ where p_x and p_t give the smoothness of ϕ in terms of x and t. The test function satisfies $\int_{\Omega_{h(x_i,t^n)}}\phi(x,t)dxdt = 1$ and $\phi(x,t) = 0$ for $(x,t)\in\partial\Omega_{h(x_i,t^n)}$, with $\phi(x,t)$ localized around (x_i,t^n) and is supported on $\Omega_{h(x_i,t^n)} = [x_i - m_x\Delta x, x_i + m_x\Delta x]\times[t^n - m_t\Delta t, t^n + m_t\Delta t]$ for some positive integers m_x and m_t. The weak features $w_{h(x_i,t^n)}$ in (Equation 4.9) can be written into a convolution form $U * \frac{\partial^{\alpha_l}}{\partial x^{\alpha_l}}\phi$ and calculated

through Fast Fourier Transform in terms of $\mathcal{F}^{-1}\left(\mathcal{F}(U) \circ \mathcal{F}\left(\frac{\partial^{\alpha_l}}{\partial x^{\alpha_l}}\phi\right)\right)$, where $\circ$ denotes point-wise multiplication, and p_x, p_t, m_x and m_t are carefully chosen to give a denoising effect depending on the frequency of the given data as in [36]. For the completeness, more details are presented in Section B.2.1.

The the weak form (Equation 4.8) has $\mathbb{N}_x\mathbb{N}_t$ rows. For computational efficiency, we subsample $\boldsymbol{W}$ to

$$H = N_x N_t \leq \mathbb{N}_x\mathbb{N}_t, \tag{4.11}$$

rows by uniformly subsampling N_x and N_t points in space and time respectively. Then, we consider highly dynamic regions to further reduce the size of $\boldsymbol{W}$ and $\boldsymbol{b}$ for an improved coefficient recovery (details in Section 4.3.2). In comparison, random subsampling is used in [38] for sparse regression, and regions with large gradients in time are considered in [37].

4.2.3 Error Analysis of the Weak formulation

We next analyze the approximation error of the weak formulation in (Equation 4.7). Suppose the given noisy data $\mathcal{D}$ (Equation 4.2) has mean-zero i.i.d Gaussian noise, $\mathbb{E}[\epsilon_i^n] = 0$, and $\mathrm{Var}(\epsilon_i^n) = \sigma^2$. Let c_l be the l^{th} true coefficient in the true support Supp*. The associated integral formulation using the test function (Equation 4.10) with the true coefficients from the true support becomes

$$\int_{\Omega_h} u(x,t)\frac{\partial \phi_h(x,t)}{\partial t}dxdt + \sum_{l\in \text{Supp}^*}(-1)^{\alpha_l}c_l\int_{\Omega_h} f_l(x,t)\frac{\partial^{\alpha_l}\phi_h(x,t)}{\partial x^{\alpha_l}}dxdt = 0. \tag{4.12}$$

We next analyze the error for the discretized system in (Equation 4.8) using the noisy data $\{\hat{U}_i^n\}$, approximating the true equation (Equation 4.12). The h^{th} row of the linear system (Equation 4.8) is obtained from the weak form with the test function ϕ_h. The error

for the discretized system in (Equation 4.8) is defined as

$$\boldsymbol{e} = \boldsymbol{W}\boldsymbol{c} - \boldsymbol{b} \tag{4.13}$$

where the row-wise error is

$$\begin{aligned} e_h = &\sum_{l\in \text{Supp}^*} \sum_{(x_j,t^k)\in\Omega_{h(x_i,t^n)}} (-1)^{\alpha_l} c_l \hat{U}_j^k \frac{\partial^{\alpha_l}}{\partial x^{\alpha_l}} \phi_h(x_j,t^k)\Delta x\Delta t \\ &+ \sum_{(x_j,t^k)\in\Omega_{h(x_i,t^n)}} \hat{U}_j^k \frac{\partial \phi_h}{\partial t}(x_j,t^k)\Delta x\Delta t. \end{aligned}$$

We decompose the error as

$$\boldsymbol{e} = \boldsymbol{e}^{\text{int}} + \boldsymbol{e}^{\text{noise}} \tag{4.14}$$

where

$$\begin{aligned} e_h^{\text{noise}} = e_h - &\left(\sum_{l\in \text{Supp}^*} \sum_{(x_j,t^k)\in\Omega_{h(x_i,t^n)}} (-1)^{\alpha_l} c_l U_j^k \frac{\partial^{\alpha_l}}{\partial x^{\alpha_l}} \phi_h(x_j,t^k)\Delta x\Delta t + \sum_{(x_j,t^k)\in\Omega_{h(x_i,t^n)}} U_j^k \frac{\partial \phi_h}{\partial t}(x_j,t^k)\Delta x\Delta t \right) \\ e_h^{\text{int}} = &\left(\sum_{l\in \text{Supp}^*} \sum_{(x_j,t^k)\in\Omega_{h(x_i,t^n)}} (-1)^{\alpha_l} c_l U_j^k \frac{\partial^{\alpha_l}}{\partial x^{\alpha_l}} \phi_h(x_j,t^k)\Delta x\Delta t + \sum_{(x_j,t^k)\in\Omega_{h(x_i,t^n)}} U_j^k \frac{\partial \phi_h}{\partial t}(x_j,t^k)\Delta x\Delta t \right) \\ &- \left(\sum_{l\in \text{Supp}^*} c_l \int_{\Omega_{h(x_i,t^n)}} (-1)^{\alpha_l} u^{\beta_l} \frac{\partial^{\alpha_l}\phi_h}{\partial x^{\alpha_l}} dxdt + \int_{\Omega_{h(x_i,t^n)}} u(x,t) \frac{\partial \phi_h(x,t)}{\partial t} dxdt. \right). \end{aligned}$$

In this decomposition, $\boldsymbol{e}^{\text{int}}$ represents the numerical integration error of the noise-free data U. It has been shown in [36] that $\boldsymbol{e}^{\text{int}} = \mathcal{O}((\Delta x\Delta t)^{q+1})$, where q is the order of the numerical integration as in [36], if the the decay of test function ϕ near the boundary of the test region satisfies $\max\{\phi(1-1/m_x,0), \phi(0,1-1/m_t)\} \leq (\frac{2\max\{m_x,m_t\}-1}{\max\{m_x,m_t\}^2})^{q+1}$.

The following Theorem 4.2.1 provides an estimate of the error $\boldsymbol{e}^{\text{noise}}$ arising from noise.

Theorem 4.2.1. *Consider a dynamical system*

$$u_t = \sum_{l \in Supp^*} c_l \frac{\partial^{\alpha_l}}{\partial x^{\alpha_l}} u^{\beta_l}$$

of one spatial variable where Supp* denotes the true support of the underlying differential equation. Assume the noise ϵ_i^n are i.i.d. and satisfies $\mathbb{E}[\epsilon_i^n] = 0$, $\mathrm{Var}(\epsilon_i^n) = \sigma^2$, and $|\epsilon_i^n| \leq \epsilon$ for all i and n. Each test region $\Omega_{h(x_i,t^n)}$ for $i = 1, ..., \mathbb{N}_x, n = 1, ..., \mathbb{N}_t$ has area $|\Omega_h| = m_x m_t \mathbb{N}_x \mathbb{N}_t$. Then,

1. *In* (Equation 4.14)*, the error from noise $\boldsymbol{e}^{\text{noise}}$ for the discretized system satisfies*

$$\|\boldsymbol{e}^{\text{noise}}\|_\infty \leq \bar{S}^* |\Omega_h| \epsilon + \mathcal{O}\left(\epsilon^2\right) \tag{4.15}$$

   with a constant

$$\bar{S}^* = \max_h \sup_{(x_j,t^k) \in \Omega_h} \left| \sum_{l \in Supp^*} (-1)^{\alpha_l} c_l \beta_l (U_j^k)^{\beta_l - 1} \frac{\partial^{\alpha_l} \phi}{\partial x^{\alpha_l}}(x_j, t^k) - \frac{\partial \phi}{\partial t}(x_j, t^k) \right|. \tag{4.16}$$

2. *The leading error in e_h^{noise} (that is linear in noise) for the test function ϕ_h has mean 0 and variance $\sigma^2 S_h^*$ where*

$$S_h^* = \Delta x \Delta t \sum_{(x_j,t^k) \in \Omega_{h(x_i,t^n)}} \left(\sum_{l \in Supp^*} (-1)^{\alpha_l} c_l \beta_l (U_j^k)^{\beta_l - 1} \frac{\partial^{\alpha_l} \phi_h}{\partial x^{\alpha_l}}(x_j, t^k) + \frac{\partial \phi_h}{\partial t}(x_j, t^k) \right)^2. \tag{4.17}$$

Theorem 4.2.1 is proved in Appendix Section B.1. In summary, we prove that the error $\boldsymbol{e}$ in (Equation 4.13) for the discretized linear system under the weak formulation satisfies the following upper bound

$$\|\boldsymbol{e}\|_\infty \leq \mathcal{O}((\Delta x \Delta t)^{q+1}) + \bar{S}^* |\Omega_h| \epsilon + \mathcal{O}(\epsilon^2) \tag{4.18}$$

where q is the order of the numerical integration as in [36]. By comparison, the error for the discretized system under the differential form [29] is on the order of

$$\mathcal{O}\left(\Delta t + \Delta x^{p+1-r} + \frac{\epsilon}{\Delta t} + \frac{\epsilon}{\Delta x^r}\right), \tag{4.19}$$

where r is the highest order of derivatives for the features in the true support, and the numerical differentiation is carried by interpolating the data by a pth order polynomial. By comparing (Equation 4.18) and (Equation 4.19), we observe that the error for the discretized linear system in the weak form is significantly smaller than the error in the differential form.

4.3 WeakIdent Algorithm

In this section, we present the details of the proposed Weak formulation for Identifying Differential Equation using Narrow-fit and Trimming (WeakIdent) model. There are mainly four steps to the algorithm: After the system is set-up as in (Equation 4.8),

[Step 1] For each sparsity level k, we use Subspace Pursuit (SP)[182] to find an initial choice of support $\mathcal{A}_0^k$ from the dictionary of L features. SP finds the choice with the minimum residual from a column-wise normalized (Equation 4.21) linear system as in [30].

[Step 2] Narrow-fit. To recover the coefficient value using the support $\mathcal{A}_j^k$, we (i) identify highly dynamic regions of certain features of interest; (ii) normalize the reduced feature matrix according to the leading error term, then (iii) determine a coefficient value vector $\boldsymbol{c}(k, j)$ from this reduced narrow system (We set $j = 0$ on the first iteration).

[Step 3] Trimming. With the updated coefficient values $\boldsymbol{c}(k, j)$ in [Step 2], we identify a single feature with the least contribution to f. If the contribution score is less than a

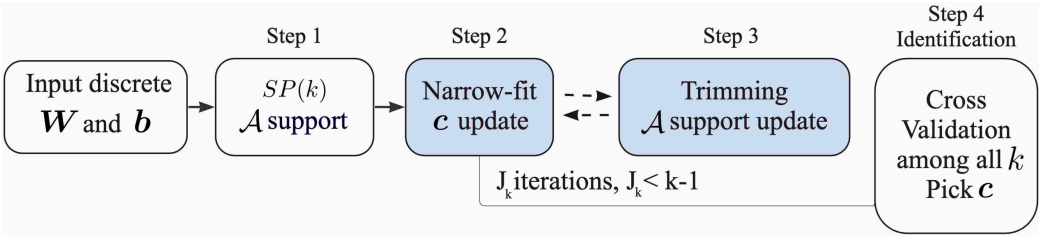


Figure 4.1: WeakIdent flowchart: Input weak formulation $\boldsymbol{W}$ and $\boldsymbol{b}$ in (Equation 4.8) subsampled as (Equation 4.11). [Step 1] SP for a given sparsity k gives the first candidate of coefficient support $\mathcal{A}_0^k$. [Step 2] Narrow-fit and [Step 3] Trimming improves the coefficient values $\boldsymbol{c}(k, j)$ and support $\mathcal{A}_j^k$. Steps 2 and 3 are iterated at most $k-1$ times. Finally, in [Step 4] the result $\boldsymbol{c}(k^*, J_{k^*})$ with the minimum Cross Validation among all different sparsity level k give the identification of the differential equation.

preset trimming parameter $\mathcal{T}$, we trim the corresponding coefficient. This trimming yields a new updated support $\mathcal{A}_j^k$. We iterate [Step 2] and [Step 3], with increment j, until no change is made to $\mathcal{A}_j^k$ at $j = J_k$.

[Step 4] Cross Validation. With the final support $\mathcal{A}_{J_k}^k$ and coefficient value vector $\boldsymbol{c}(k, J_k)$ for each different sparsity level k, we select the one $\boldsymbol{c}(k^*, J_{k^*})$ with the minimum Cross-Validation error (item 4.30) as the final result.

A schematic of the algorithm is given in Figure 4.1. From the weak form input $\boldsymbol{W}$ and $\boldsymbol{b}$, for a fixed sparsity level k, SP is used to find the initial set of support $\mathcal{A}_0^k$. Then [Step 2] Narrow-fit and [Step 3] Trimming are iterated until the support does not change, where the number of iterations is at most $k-1$. Here we use $\boldsymbol{c}(k, j)$ to indicate the coefficient vector for the sparsity level k and j iteration. The cross validation is used to select the optimal solution $\boldsymbol{c}(K, J_K)$ among all $k \leq L$.

We present the details in the following subsections. In [Step 2], we normalize each column of the feature matrix according to its leading error term, to balance the effect of noise perturbations across the features. The details for this error normalization of the feature matrix are given in Section 4.3.1. We detail the implementation of Narrow-fit using the highly dynamic regions in Section 4.3.2. In [Step 3], we trim the support removing features with contributions below a threshold, as described in detail in Section 4.3.3. The algorithm is

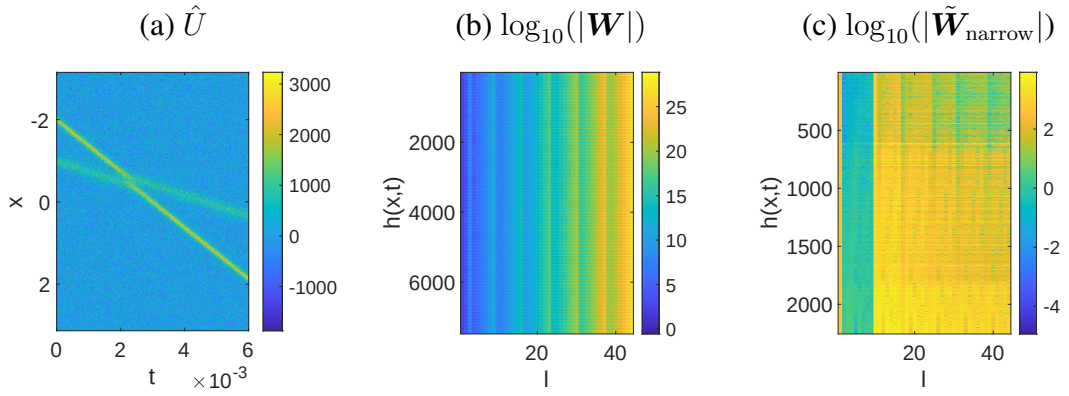


Figure 4.2: Error normalization: (a) The given noisy data $\hat{U}$ with $\sigma_{\text{NSR}} = 0.5$ in $x-t$ plane. (b) The entry-wise magnitude of the matrix $\boldsymbol{W}$. (c) The matrix $\tilde{\boldsymbol{W}}_{\text{narrow}}$ in (Equation 4.23). We use $\log 10$ scale in (b) and (c). The difference in scale has been reduced approximately from 10^{29} in the unnormalized matrix (b) to 10^{6} after normalization in (c). Our error normalization results in more uniform entry values with less variance across different columns.

summarized Section 4.3.4.

4.3.1 Column-wise error normalized matrix

We use least squares for coefficient recovery. The accuracy of least squares is highly dependent on the conditioning of the feature matrix [196, 197]. In this chapter, we utilize two types of normalization for the columns of the feature matrix to improve the coefficient recovery. For the linear system (Equation 4.8), we introduce a diagonal matrix $\boldsymbol{D} = \text{diag}(d_1, ..., d_L)$ and solve

$$\boldsymbol{W}\boldsymbol{D}^{-1}\bar{\boldsymbol{c}} = \boldsymbol{b} \quad \text{and then} \quad \boldsymbol{c} = \boldsymbol{D}^{-1}\bar{\boldsymbol{c}} \tag{4.20}$$

instead.

The first type of normalization we consider is **column normalization**, which is applied to the feature matrix as an input to SP in [Step 1]. Denote $\boldsymbol{W} = [\boldsymbol{w}_1\ \boldsymbol{w}_2\ \dots\ \boldsymbol{w}_L]$. We let $\boldsymbol{D} = \text{diag}(\|\boldsymbol{w}_1\|, ..., \|\boldsymbol{w}_L\|)$ and each column of $\boldsymbol{W}$ is normalized by its own norm:

$$\boldsymbol{W}^{\dagger} = \left[\frac{\boldsymbol{w}_1}{\|\boldsymbol{w}_1\|}, \frac{\boldsymbol{w}_2}{\|\boldsymbol{w}_2\|}, \dots, \frac{\boldsymbol{w}_L}{\|\boldsymbol{w}_L\|}\right]. \tag{4.21}$$

We observe that the scale of the columns in the feature matrix usually varies substantially from column to column, which negatively affects the SP step. This column normalization helps to prevent a large difference in the scale among the columns. For example, in Figure 4.2 (b) shows that the magnitude of the entries in $\boldsymbol{W}$ vary from 0 to 10^{29}.

In [Step 2], we introduce our second normalization – **error normalization**, which is particularly effective for coefficient recovery. The columns in $\boldsymbol{W}$ are given by certain derivatives of a monomial of u. When we compute the feature matrix with noisy data, the noise has different effects on different features. For the feature $\frac{\partial^\alpha}{\partial x^\alpha}\left(u^\beta\right)$, the noisy data with noise ϵ in (Equation 4.3) give rise to the following integral feature:

$$\int_{\Omega_h}(-1)^\alpha(u+\epsilon)^\beta\frac{\partial^\alpha}{\partial x^\alpha}\left(\phi_h(x,t)\right)dxdt=\sum_{k=0}^{\beta}(-1)^\alpha\binom{\beta}{k}\epsilon^{\beta-k}\int_{\Omega_h}u^k\frac{\partial^\alpha}{\partial x^\alpha}\phi_h(x,t)dxdt.$$

The leading coefficient in the error (that is linear in ϵ) in this integral feature is obtained for $k=\beta-1$:

$$s(h,l)=\beta\left|\int_{\Omega_h}u^{\beta-1}\frac{\partial^\alpha}{\partial x^\alpha}\left(\phi_h(x,t)\right)dxdt\right|,\quad h=1,2,...,H,\ \beta\geq 1. \tag{4.22}$$

When $\alpha=\beta=0$, we set $s(h,l)=1$. This leading coefficient $s(h,l)$ depends on the row index h and the column index l. For the l^{th} column, we define

$$\langle s(h,l)\rangle_h=\frac{1}{H}\sum_{h=1}^{H}s(h,l)$$

as an average of these leading coefficients over the rows.

By error normalization, we normalize $\boldsymbol{W}$ with the diagonal matrix $\boldsymbol{D}=\text{diag}(\langle s(h,1)\rangle_h,\ldots,\langle s(h,L)\rangle$ such that $\boldsymbol{W}$ is normalized to

$$\tilde{\boldsymbol{W}}=\left[\frac{\boldsymbol{w}_1}{\langle s(h,1)\rangle_h},\frac{\boldsymbol{w}_2}{\langle s(h,2)\rangle_h},\ldots,\frac{\boldsymbol{w}_L}{\langle s(h,L)\rangle_h}\right] \tag{4.23}$$

Figure 4.2 shows an example, with the given noisy data in (a) and the unnormalized feature matrix $\boldsymbol{W}$ in (b). Figure 4.2 (c) shows the normalized matrix $\tilde{\boldsymbol{W}}$ after the error normalization. We use $\log 10$ scale in Figure 4.2. The difference in scale has been reduced approximately from 10^{29} in the unnormalized matrix (b) to 10^{6} after normalization in (c). Our error normalization results in more uniform entry values with less variation across different columns.

In the following Subsection, we further discuss how error normalization is used to select the highly dynamic regions .

4.3.2 Highly dynamic regions

One of the benefits of using the weak form is to consider the influence of different regions on the integral computation. We take advantage of this and choose a subset of test functions indexed by $\{h|h = 2, \ldots, H\}$ to improve the coefficient recovery. We propose the following Narrow-fit procedure: (i) define the features of interest, (ii) determine the highly dynamic regions of the chosen features, and then (iii) use the subsampled matrix based on the highly dynamic regions for the coefficient recovery. This Narrow-fit procedure focuses on the regions with higher dynamical behaviors for the features of interest, so that these regions play a larger role in the coefficient recovery.

Features of interest: We focus on a small group of features which give the variation information for the differential equation, thus highlighting which rows to choose for the coefficient recovery. In this paper, we choose the features of interest to be the terms corresponding to u and first derivatives consistently for all experiments. We simply utilize the high variance region of the function value u and the first derivative, e.g., a term such as uu_x which gives a combined information, since they would likely represent a broad range of dynamical behavior observed in the data. We explored including other terms as features of interest, but they did not provide consistent improvements.

Details are as follows: In 1D, we choose the features with $(\alpha, \beta) = (1, 2)$ for the case

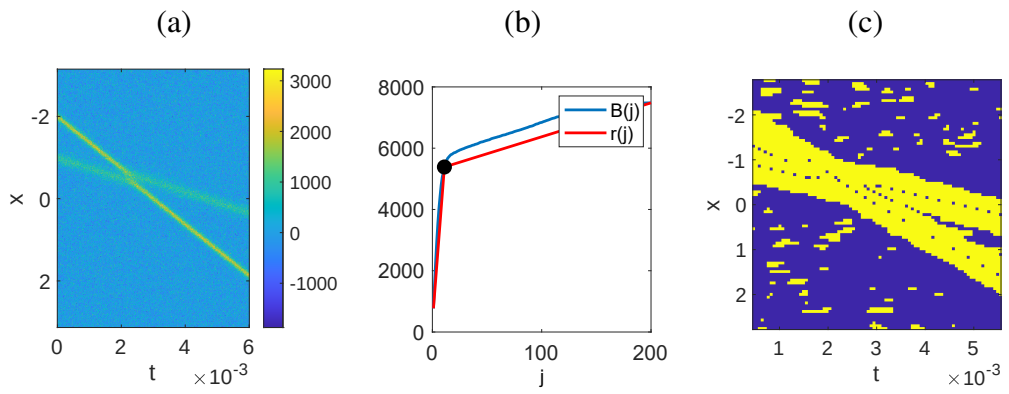


Figure 4.3: Highly dynamic regions for an experiment using the KdV equation (Equation 4.33) with $\sigma_{\text{NSR}} = 0.5$. (a) The given noisy data $\hat{U}$ with $\sigma_{\text{NSR}} = 0.5$ in $x - t$ plane. (b) The separation point Γ(black) for $\mathbb{H}$ (Equation 4.24) is found, from the accumulated function $B(j)$ (blue) and the fitted piecewise linear function $r(j)$ with one junction at Γ (red). (c) The location of highly dynamic regions in the $x - t$ plane.

of one variable in 1D which corresponds to $\frac{\partial}{\partial x}u^2$, this term is uu_x. For a system with two variables u, v in 1D, $(\alpha, \beta_u, \beta_v) = (1, 2, 0), (1, 0, 2)$, they are $\frac{\partial}{\partial x}u^2$ and $\frac{\partial}{\partial x}v^2$. In 2D, we choose the features with $(\alpha_x, \alpha_y, \beta) = \{(1, 0, 2), (0, 1, 2), (1, 1, 3)\}$ for a scalar equation in 2D, i.e., the features of interest are $\frac{\partial}{\partial x}u^2$, $\frac{\partial}{\partial y}u^2$ and $\frac{\partial^2}{\partial x \partial y}u^3$. For the case of 2 variables (u and v) in 2D, $(\alpha_x, \alpha_y, \beta_u, \beta_v) = \{(1, 0, 2, 0), (0, 1, 2, 0), (1, 0, 0, 2), (0, 1, 0, 2), (1, 0, 2, 1), (0, 1, 1, 2)\}$, that is there are six features of interest: $\frac{\partial}{\partial x}u^2$, $\frac{\partial}{\partial x}v^2$, $\frac{\partial}{\partial y}u^2$, $\frac{\partial}{\partial y}v^2$, $\frac{\partial}{\partial x}u^2v$, and $\frac{\partial}{\partial y}uv^2$.

For each feature of interest, we utilize the leading coefficient error (Equation 4.22) to select highly dynamic regions. For multiple features of interest with indices $l = l_1, l_2, ..., l_{\mathcal{L}}$, we take the average over l, and let

$$\bar{s}(h) = \frac{1}{\mathcal{L}} \sum_{i=1}^{\mathcal{L}} |s(h, l_i)|,$$

with $s = \bar{s}$ for $\mathcal{L} = 1$.

Highly dynamic regions: We consider the set $\mathcal{S} = \{\bar{s}(h) | h = 1, \ldots, H\}$, which is the collection of averaged leading coefficient errors over the features of interest. We divide the set $\mathcal{S}$ into mildly and highly dynamic regions, automatically identifying the transition point Γ between these two types of dynamics as follows.

After partitioning the histogram of $\mathcal{S}$ into $N_{\mathcal{S}}$ bins $(b_1, b_2, ..., b_{N_{\mathcal{S}}})$, we consider the cumulative sum of the bins $B(j) = \sum_{i=1}^{j} b_i$. We used $N_{\mathcal{S}} = 200$ for PDEs and $N_{\mathcal{S}} = 100$ for ODEs in this paper. We fit the function $B(j)$ with a piecewise linear function $r(j)$ with one junction point, using the cost function $\sum_j (B(j) - r(j))^2 / B(j)^2$. The junction point Γ separates the highly dynamic and mildly dynamic regions. Any h with $\bar{s}(h) \geq \Gamma$ gives the highly dynamic region Ω_h which we include for the coefficient recovery. Let the collection of the row indices of highly dynamic regions be an ordered set:

$$\mathbb{H} = \{h_i \mid \bar{s}(h_i) \geq \Gamma,\ h_i < h_j \text{ for } i < j\}. \tag{4.24}$$

Figure 4.3 illustrates how the transition point Γ is computed in (b) from the given data in (a). Figure 4.3 (c) shows the locations in $x-t$ plane of the highly dynamics regions with the index set $\mathbb{H}$.

Narrow-fit: We consider a submatrix using only the ordered rows from the highly dynamic region $\mathbb{H}$, indicated by a subscript $\mathbb{H}$, for both $\boldsymbol{W}$ and $\boldsymbol{b}$:

$$\boldsymbol{W}_{\text{narrow}} := \boldsymbol{W}_{\mathbb{H}} \quad \text{and} \quad \boldsymbol{b}_{\text{narrow}} := \boldsymbol{b}_{\mathbb{H}}.$$

We also error normalize this matrix, using the rows in $\mathbb{H}$:

$$\tilde{\boldsymbol{W}}_{\text{narrow}} = \left[\frac{\boldsymbol{w}_{1\mathbb{H}}}{\langle s(h,1) \rangle_{\mathbb{H}}}, \frac{\boldsymbol{w}_{2\mathbb{H}}}{\langle s(h,2) \rangle_{\mathbb{H}}}, \ldots, \frac{\boldsymbol{w}_{L\mathbb{H}}}{\langle s(h,L) \rangle_{\mathbb{H}}} \right], \tag{4.25}$$

where $\boldsymbol{w}_{i\mathbb{H}}$ represents the i^{th} column with the rows indexed by $\mathbb{H}$, and $\langle s(h,l) \rangle_{\mathbb{H}}$ takes the average of $s(h,l)$ for $h \in \mathbb{H}$. This matrix is represented in Figure 4.2 (c). Let $\bar{b} = \langle \boldsymbol{b}_{narrow} \rangle$ be the average of the entries of $\boldsymbol{b}_{\text{narrow}}$. After narrow-fitting, We solve:

$$\tilde{\boldsymbol{W}}_{\text{narrow}} \tilde{\boldsymbol{c}} = \tilde{\boldsymbol{b}}_{\text{narrow}} \quad \text{where} \quad \tilde{\boldsymbol{b}}_{\text{narrow}} = \boldsymbol{b}_{\text{narrow}} / \bar{b}. \tag{4.26}$$

We then compute the coefficient $\boldsymbol{c}$ by rescaling $\tilde{\boldsymbol{c}}$ back:

$$\boldsymbol{c} = \bar{b}\ \tilde{\boldsymbol{c}}\ \text{diag}\left\{\frac{1}{\langle s(h,1)\rangle_{\mathbb{H}}}, \frac{1}{\langle s(h,2)\rangle_{\mathbb{H}}}, \ldots, \frac{1}{\langle s(h,L)\rangle_{\mathbb{H}}}\right\}. \tag{4.27}$$

4.3.3 Trimming the support

After the coefficient values in $\boldsymbol{c}$ are recovered, some features give very small contributions to u_t. We further trim the support by eliminating these features corresponding to small contributions.

From the solution $\tilde{\boldsymbol{c}}$ of the linear equation (Equation 4.26), we define a **contribution score** a_i of each feature as

$$a_i = \frac{n_i}{\max_{i \le L} n_i} \quad \text{where} \quad n_i = ||\tilde{\boldsymbol{w}}_i||_2 |\tilde{c}_i|, \quad i = 1, 2, \ldots, L. \tag{4.28}$$

Here $\tilde{\boldsymbol{w}}_i$ denotes the i^{th}column of $\tilde{\boldsymbol{W}}_{\text{narrow}}$. We consider the L_2 norm of this column multiplied by the coefficient value of the i^{th} component of $\tilde{c}$. Since a_i is normalized by the maximum value of n_i, a_i gives the score of the contribution of the i^{th} feature relative to the contribution of the feature with the largest contribution.

We trim the coefficient, thus the feature, when the contribution score of that feature is below $\mathcal{T}$, i.e. $a_i < \mathcal{T}$. Typically, we set $\mathcal{T} = 0.05$ to trim the features with contributions less than 5% of u_t. Each time [Step 3] is called to trim the support set $\mathcal{A}_j^k$ to the new support set $\mathcal{A}_{j+1}^k$, and [Step 2] narrow-fit is called to find the updated coefficient value $c(k, j+1)$.

(Figure 4.4) shows the effect of trimming. For each sparsity level k in x-axis, the bar shows the cross validation value (item 4.30) of the recovered coefficient $c(k^*, J_{k^*})$. For a large sparsity level, thanks to the trimming step, the correct support and coefficient values are found.

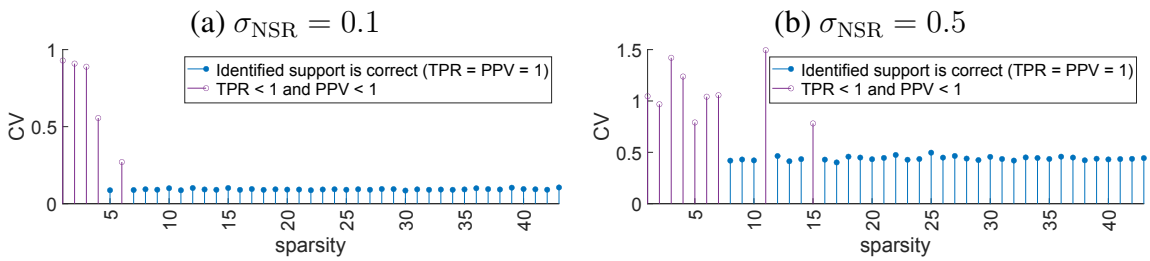


Figure 4.4: Trimming is demonstrated in an experiment using the KS equation (Equation 4.34). For each sparsity level k in x-axis, the bar shows the cross validation (item 4.30) of the recovered coefficient $c(k^*, J_{k^*})$. Notice for most sparsity levels 5 and above the correct support is found. After SP finds k supports, the trimming step reduces the support until only the correct ones are left. Here σ_{NSR} is the noise-to-signal ratio (Equation 4.44), TPR is true positive rate (Equation 4.47) and PPV is positive prediction value (Equation 4.48).

4.3.4 Algorithms

Our WeakIdent algorithm is summarized in Algorithm 6. From the linear system in (Equation 4.8)

$$\boldsymbol{W}\boldsymbol{c} = \boldsymbol{b},$$

we input $\boldsymbol{b}$ and $\boldsymbol{W}$ computed through (Equation 4.9), with subsampling in (Equation 4.11). For each sparsity level $k = 1, 2, \ldots, K$,

[Step 1] First, Subspace Pursuit (SP)[182] is applied to find $\mathcal{A}_o^k = \text{supp}\{SP(\boldsymbol{W}\dagger, \tilde{\boldsymbol{b}}, s)\}$ using the column normalized matrix $\boldsymbol{W}\dagger$ in (Equation 4.21) and $\tilde{\boldsymbol{b}} = \boldsymbol{b}/||\boldsymbol{b}||$.

[Step 2] Narrow-fit. To recover the coefficient values using the support $\mathcal{A}_j^k$, we find the row index set $\mathbb{H}$ of highly dynamic regions in (Equation 4.24), and solve

$$\tilde{\boldsymbol{W}}_{\text{narrow}}\tilde{\boldsymbol{c}} = \tilde{\boldsymbol{b}}_{\text{narrow}}$$

in (Equation 4.26) and get $\boldsymbol{c}(k, j)$ in (Equation 4.27).

[Step 3] Trimming. Update to $\mathcal{A}_{j+1}^k$, if there is any column with the contribution score in (Equation 4.28) below $\mathcal{T}$, i.e. $a_i < \mathcal{T}$. If trimmed, move to [Step 2] to get a new

updated $\boldsymbol{c}(k, j+1)$. If no column is trimmed, move to [Step 4] and set $J_k = j$.

[Step 4] Cross Validation. With the support $\boldsymbol{c}(k, J_k)$ computed for each sparsity level $k = 1, \ldots, K$, we select the final support by finding the k^* which gives the minimum cross-validation error. For a sparsity level k, we randomly sample regions from the $N_x N_t$ regions and equally partition these regions into two sets indexed by $\mathbb{A}$ and $\mathbb{B}$ respectively. We consider the linear system in (Equation 4.26):

$$\tilde{\boldsymbol{W}} = \left[\frac{\boldsymbol{w}_1}{\langle s(h,1) \rangle_{\mathbb{H}}}, \frac{\boldsymbol{w}_2}{\langle s(h,2) \rangle_{\mathbb{H}}}, \ldots, \frac{\boldsymbol{w}_L}{\langle s(h,L) \rangle_{\mathbb{H}}} \right], \quad \text{and} \quad \tilde{\boldsymbol{b}} = \boldsymbol{b}/\bar{b} \tag{4.29}$$

utilizing the highly dynamic region error normalization for the large full matrix. Here $\mathbb{H}$ indicates ordered row index from the set $\mathbb{H}$, and $\langle s(h,l) \rangle_{\mathbb{H}}$ taking the average of $s(h,l)$ for $h \in \mathbb{H}$. We solve least square problems $\tilde{\boldsymbol{W}}_{\mathbb{A}} \tilde{\boldsymbol{c}}_{\mathbb{A}} = \tilde{\boldsymbol{b}}_{\mathbb{A}}$ and $\tilde{\boldsymbol{W}}_{\mathbb{B}} \tilde{\boldsymbol{c}}_{\mathbb{B}} = \tilde{\boldsymbol{b}}_{\mathbb{B}}$, where $\tilde{\boldsymbol{W}}_{\mathbb{A}}$ and $\mathbb{B}$ contain the rows of $\tilde{\boldsymbol{W}}$ indexed by $\mathbb{A}$ and $\mathbb{B}$ respectively. Then, we compute the **cross validation** (CV) error

$$\mathrm{CV}(k) = \lambda ||\tilde{\boldsymbol{W}}_{\mathbb{A}} \tilde{\boldsymbol{c}}_{\mathbb{B}} - \tilde{\boldsymbol{b}}_{\mathbb{A}}||_2 + (1-\lambda) ||\tilde{\boldsymbol{W}}_{\mathbb{B}} \tilde{\boldsymbol{c}}_{\mathbb{A}} - \tilde{\boldsymbol{b}}_{\mathbb{B}}||_2, \tag{4.30}$$

where we set $\lambda = 1/100$. In practice, for each k, we generate 30 different random partitions of $\mathbb{H}$ to $\mathbb{A}$ and $\mathbb{B}$, then select the minimum:

$$\boldsymbol{c}(k^*, J_{k^*}) = \arg\min_k \{ \mathrm{CV}(k) | k = 1, 2, ..., L \}. \tag{4.31}$$

Here $K \leq L$, since L is the total number of features in the dictionary. In practice, a small K is needed. (Figure 4.4) illustrates that for (small) values of K around $K = 10$ and below, the correct coefficients are found, thanks to the trimming step.

Algorithm 6 WeakIdent Algorithm

Input: $\boldsymbol{W} \in \mathbb{R}^{H\times L}, \boldsymbol{b} \in \mathbb{R}^{H}$, from (Equation 4.8) uniformly subsampled as (Equation 4.11); Parameter $\mathcal{T} = 0.05$
for $k = 1, 2, ..., K$ **do**

[Step 1] $\mathcal{A}_0^k = \text{supp}\{\text{SP}(\boldsymbol{W}\dagger, \tilde{\boldsymbol{b}}, s)\}$ use SP [182] and set $j = 0$;
[Step 2] Find $c(k, j)$ by narrow-fit (Equation 4.26)
while there exists $a_i < \mathcal{T}$ as in (Equation 4.28) **do**
[Step 3] Trim as in Section 4.3.3 and set $j = j + 1$
[Step 2] Find $c(k, j + 1)$ by Narrow-fit (Equation 4.26)
end while
end for
Among $k = 1, \ldots, K$, find $\boldsymbol{c}(k^*, J_{k^*})$ by Cross Validation in (item 4.31).
Output: $\boldsymbol{c} = \boldsymbol{c}(k^*, J_{k^*}) \in \mathbb{R}^L$ such that $\boldsymbol{W}\boldsymbol{c} \approx \boldsymbol{b}$.

4.4 WeakIDENT results and comparions

In this section, we provide detailed experimental results. We summarize a list of PDEs and ODE systems in Table 4.1 and Table 4.2. For the systems of ODEs, we consider features with polynomial order between 3 and 5 , with $L \leq 21$ for all the cases. For the systems of PDEs, we consider features with both polynomial order and derivative order between 4 and 6, which gives a dictionary of size $L \leq 65$ for the 1 spatial dimension and $L \leq 190$ for 2 spatial dimensions. Simulation and feature details are presented in Table 4.1 and Table 4.2 for each experiment.

For PDEs, N_x and N_t are chosen such that $N_x N_t \in (1,000, 3,000)$ to reduce the computational cost. In particular, we set

$$N_x = \lceil \frac{\mathbb{N}_x - 2m_x - 1}{\lfloor \mathbb{N}_x/\mathbf{N} \rfloor} + 1 \rceil \quad \text{and} \quad N_t = \lceil \frac{\mathbb{N}_t - 2m_t - 1}{\lfloor \mathbb{N}_t/\mathbf{N} \rfloor} + 1 \rceil, \tag{4.38}$$

with $\mathbf{N} = 50$ as a default choice. Here $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ denotes the ceiling and floor operator. In Table 4.1, (Equation 4.38) is used for the transport question (Equation 4.32), the KS equation (Equation 4.34) and the nonlinear Schrodinger equation (Equation 4.35). For certain cases such as the KdV equation (Equation 4.33) where $|\mathbb{H}|$ is very small, we increase N_x

and N_t, e.g., using $\mathbf{N} = 70$, such that $|\mathbb{H}| > 800$. For the spatially 2 dimensional cases, we use $\mathbf{N} = (25, 25)$ for the anisotropic porous medium equation (PM) (Equation 4.36), and $\mathbf{N} = (19, 16)$ for the 2D reaction-diffusion equation (Equation 4.37) to reduce the time of computation. For the ODEs listed in Table 4.2, we choose $N_t \approx 1000$ by default with $\mathbf{N} = 1000$. Since we use different subsampling, we present additional comparisons in Section 4.5.3 to demonstrate that the effect of subsampling on the result is minimal.

The experiments are performed on both clean data and noisy data with various Noise-to-Signal Ratio, σ_{NSR} defined as follows:

$$\sigma_{\text{NSR}} = \frac{\epsilon_i^n}{\frac{1}{\mathbb{N}_t\mathbb{N}_x}\sum_{i,n} |U_i^n - (\max_{i,n} U_i^n + \min_{i,n} U_i)/2|^2} \tag{4.44}$$

for $i = 1, ..., \mathbb{N}_x$, $n = 1, ..., \mathbb{N}_t$. Note that our definition of NSR reflects the local variation of the given data. This is different from the absolute variation (absolute root mean squared of U_i^n) σ_{NR} used in [36], and this σ_{NSR} value tends to be smaller than the σ_{NR} value. We also mention the σ_{NR} value in the following experiments when it is relevant. We use Gaussian noise, such that $\epsilon_i^n \sim \mathcal{N}(0, \sigma_{\text{NSR}})$ for ϵ_i^n, and $\hat{U}_i^n$ in (Equation 4.3). For the case of multiple variables, we compute (Equation 4.44) for each variable.

Error measures: To quantify the quality of the recovery, we utilize different error measurements listed in Table 4.3. The relative coefficient errors E_2 in (Equation 4.45) and E_∞ in (Equation 4.46) measure the accuracy of the recovered coefficients $\boldsymbol{c}$ against the true coefficients $\boldsymbol{c}^*$ in terms of the l_2 and the infinity norm, respectively. We introduce two new measures to quantify the accuracy of the support recovery. The True Positive Rate (TPR) [1] (Equation 4.47) measures the fraction of features that are found out of all features in the true equation, and is defined as the ratio of the cardinality of the correctly identified support over the cardinality of the true support. The TPR is 1 if all the true features are found. The Positive Predictive Value (PPV) (Equation 4.48) indicates the presence of false

[1] The definition of TPR in (Equation 4.47) is different from that used in [36]

positives: it is the ratio of the cardinality of the correctly identified support over the total cardinality of the identified support. The PPV is 1 if the recovered support is also in the true support. The residual error E_{res} in (Equation 4.49), which is also used in [30], measures the relative difference between the learned differential equation and the given data. To show the effectiveness of WeakIdent in the recovery of the dynamics, we define the dynamical error E_{dyn} in (Equation 4.50) to measure the difference between the true dynamics and the expected dynamics simulated from the recovered equation. In (Equation 4.50), we use $U^n_{\mathbf{i},\text{forward}}$ and $U^n_{\mathbf{i},\text{clean}}$ to denote the simulated data and the true data without noise. We simulate ODEs using RK45 with the relative error tolerance to be 10^{-10}. This is measured for ODEs only, due to restricted stability conditions for PDEs. If the identified equation blows up before the final time T is reached, we compare $U^n_{\mathbf{i},\text{forward}}$ and $U^n_{\mathbf{i},\text{clean}}$ just before the blow-up.

4.4.1 WeakIdent on PDEs

We present the WeakIdent and comparisons in this subsection for PDEs, and in Section 4.4.2 for ODEs. We compare with existing methods, such as the IDENT in [29], the Robust Ident, with Subspace pursuit Cross validation (SC) and Subspace pursuit Time evolution (ST) in [30], SINDy [22], and methods using the weak form such as RGG [38], Weak SINDy for first order dynamical systems (WODE) [37], and Weak SINDy for PDEs (WPDE) [36].

For fair comparisons, when available, we used the same underlying equations provided by SINDy[22], WODE[37], WPDE [36] or RGG [38] provided in their respective Githubs[2]. WeakIdent and WPDE use the same dictionary of features as well as the same parameters for the weak form (a system with the same number of variables and dimensions in the spatial domain) to each other. RGG [38] uses a subset of features (e.g. 8-14 features), which is different from other methods which use the full feature matrix ($L = 21$ to 190 features). For each experiment in the comparison, we specify which features are used for

[2]SINDy and WODE at https://github.com/dm973/WSINDy_ODE, WPDE at https://github.com/dm973/WSINDy_PDE, and RGG at https://github.com/pakreinbold/PDE_Discovery_Weak_Formulation

RGG. In many of the PDE experiments in this section, we show comparisons only between our proposed WeakIdent and WPDE [36], since these two methods give the best results compared to others, based on the error measures in Table 4.3.

Transport equation

The first set of results in Figure 4.5 shows results for the transport equation (Equation 4.32) with clean and noisy data. (a), (b) and (c) compare the recovery results with clean data, and (d), (e) and (f) compare the results with highly corrupted data where $\sigma_{\text{NSR}} = 100\%$. For the case of clean data, RGG [38], WPDE [36] and the proposed WeakIdent find the correct support u_x, u_{xx}, while the latter two methods have higher accuracy. In the noisy case of $\sigma_{\text{NSR}} = 100\%$, only WeakIdent is able to identify the correct support with the E_2 value as low as 0.008.

In Figure 4.6, we provide statistical comparisons between our proposed WeakIdent and WPDE [36] applied to the transport equation (Equation 4.32) for different levels of σ_{NSR}. We show box-plots for the distribution of the identification errors E_2, E_∞, TPR and PPV over 50 experiments for each level of $\sigma_{\text{NSR}} \in \{0.01, 0.1, 0.2, , ..., 0.9\}$. The WeakIdent results are robust even for large noise levels: Panels (a3) and (a4) show that in the majority($> 75\%$) of the cases, a correct support is found by WeakIdent with low E_2 error in Panel (a1).

Anisotropic Porous Medium (PM) equation

In Figure 4.7, we compare the recovery results for the 2D anisotropic porous medium equation (PM) (Equation 4.36), which includes a feature with the cross-dimensional derivative u_{xy}. Figure 4.7 (a) shows $\hat{U}(\boldsymbol{x}, 0)$ and (b) shows $\hat{U}(\boldsymbol{x}, T)$, where the given noisy data has noise-to-signal ratio $\sigma_{\text{NSR}} = 0.08$. This noise level is equivalent to $\sigma_{\text{NR}} = 0.4139$ as defined in WPDE [36]. We show different recovered equations with the identification error E_2 in (c). WeakIdent is able to identify the correct support with the coefficient error $E_2 = 0.0056$, demonstrating WeakIdent's capability to identify features across multiple dimensions on 2D spatial domain.

Reaction-diffusion equation In Figure 4.8, we compare the recovery results for the 2D reaction-diffusion equation (Equation 4.37). These systems can generate a variety of patterns such as dots, strips, waves and hexagons. The Laplacian (diffusion) features $\Delta u, \Delta v$ in this equation may be difficult to identify in general, particularly in the case where the diffusion coefficients are small compared to those of other features, and accumulated noise can be emphasized. We use the spiral pattern data set from [36]. Figure 4.8 (a) shows $\hat{U}(\boldsymbol{x}, 0)$ and (b) shows $\hat{U}(\boldsymbol{x}, T)$, where the given noisy data has $\sigma_{\mathrm{NSR}} = 0.08$ (equivalent to $\sigma_{\mathrm{NR}} = 0.08$ defined in [36]). We show different recovered equations with the E_2 identification error in Figure 4.8 (c). WeakIdent finds the correct terms with a small coefficient error.

In Figure 4.9, we present the statistical results of WeakIdent over 50 experiments for the 2D reaction-diffusion equation (Equation 4.37).

PDEs and sytems of PDEs with higher order features

In Figure 4.10, we show the average errors of WeakIdent and WPDE over 50 experiments on the PDEs and systems of PDEs in Table 4.1 with different noise levels. Each column gives the E_2 error , TPR and PPV respectively. In each row, we present the results from the transport equation (Equation 4.32), KdV equation (Equation 4.33), the KS (Equation 4.34), the nonlinear Schrodinger (Equation 4.35), the anisotropic PM equation (Equation 4.36), and the 2D reaction-diffusion equation (Equation 4.37). In the first column, we present the ratio $\tilde{\sigma} = \sigma_{\mathrm{NSR}}/\sigma_{\mathrm{NR}}$ where σ_{NR} denotes the noise ratio in WPDE[36]. (The upper bounds of the noise ratio σ_{NR} [36] are 1.07, 0.78, 0.9, 0.81, 0.78, 0.1 for each equation.) Here the KdV (Equation 4.33) and KS equations (Equation 4.34) include higher order derivative features u_{xxx} and u_{xxxx}. These features are in general difficult to recover, especially from highly corrupted noisy data. Each plot gives comparisons between WeakIdent (Red) and WPDE (blue), with σ_{NSR} on the x-axis. The y-axis is the E_2 error, TPR, or PPV averaged over 50 experiments for a given σ_{NSR}. According to the E_2 error shown in the first column, WeakIdent has smaller E_2 errors than other methods, showing that WeakI-

dent is more accurate in the coefficient recovery. According to the TPR and PPV in the second and third column, WeakIdent is more accurate in support recovery since the TPR and PPV values of WeakIdent are closer to 1.

4.4.2 WeakIdent on ODEs

Since ODE systems do not include spatial derivatives, they have lower computational cost in feature computation. We consider polynomial terms with the highest order being 5. Table 4.2 presents details of the parameters used for simulation. In Figure 4.11, we show the identified dynamics and various identification errors obtained from WeakIdent on the 5 ODE systems listed in Table 4.2. The noise-to-signal ratio is $\sigma_{\mathrm{NSR}} = 0.2$ for the linear system (Equation 4.39), the Van der Pol nonlinear system (Equation 4.40) and $\sigma_{\mathrm{NSR}} = 0.1$ for the rest of the systems. Figure 4.11 (a)-(e) show the phase portraits of the given noisy data for the different ODEs (red) superimposed on the simulated true data (black). Figure 4.11 (f)-(j) show the WeakIdent results (green) compared to the true solution (black). WeakIdent is able to find the correct support in the majority of the cases with $E_2 \leq 0.088$.

Figure 4.12 compares the recovery results for the Lotka-Volterra (LV) system (Equation 4.42) across different methods, showing results for the given data sets with various noise levels. The methods we compare include WODE[37], SINDy[22], Robust IDENT SC[30] and ST[30]. Each column is associated with an error type and each row gives results from one method. WeakIdent is able to capture the correct support with a low coefficient error in the last rows. WODE, SINDy, SC and ST has larger coefficient errors with incorrect support in many cases. A similar statistical comparison between these methods on the Lorenz system (Equation 4.43) is shown in Figure B.4 in the subsubsection B.2.0.2. We refer to Table B.1 and Table B.2 for the recovery results of the Lotka-Volterra system (Equation 4.42) and the Lorenz system (Equation 4.43) from two noisy data sets with $\sigma_{SNR} = 0.1$. We also provide a comprehensive comparison on all ODE systems listed in Table 4.2 in subsubsection B.2.0.2 (See Figure B.3 for the details).

4.5 Additional experiments on the Effectiveness of WeakIdent

In this section, we show some additional experiments of using WeakIdent on equations with differential initial conditions, using different trimming parameters $\mathcal{T}$, and simulated datasets with various subsampling parameters to further show the effectiveness of WeakIdent.

4.5.1 Influence of the initial condition in WeakIdent

Figure 4.13 shows comparisons of WeakIdent and WPDE for the KS equation (Equation 4.34) on noisy data with $\sigma_{\text{NSR}} = 0.6$, using 5 different initial conditions: (1) $u(x, 0) = \cos(x/16).*$ $(1 + \sin(x/16))$, (2) $u = \cos(x/4). * (1 + \sin(x/5))$, (3) $u = \cos(x/10). * (1 + \cos(x/5))$, (4) $u = \sin(x/4). * (1 + \cos(x/5))$, (5) $u = \sin(x.^2/4)$. The top row illustrates the given clean data from the different initial conditions yielding different pattern evolution. In each box plot, the x-axis gives the indices of the initial condition (1)-(5). WeakIdent recovery is robust across these different patterns in recovering this system with higher order features.

4.5.2 The choice of the trimming parameter

In Figure 4.14, we present the coefficient E_2 error (y-sxis) against different values of the trimming parameter $\mathcal{T}$ (x-axis) for different noise-to-signal ratios (different color curves) for (a) the KdV equation (Equation 4.33) and (b) the KS equation (Equation 4.34). In general, we use $\mathcal{T} = 0.05$ as a default for all equations in Table 4.1 and Table 4.2, except for the KS equation (Equation 4.34) and the PM equation (Equation 4.36) for which we use $\mathcal{T} = 0.2$. Our experiments use the same distribution of seeds for the noise with different variances. Different color curves represent the different values of noise-to-signal ratio $\sigma_{\text{NSR}} \in \{0, 0.1, ..., 1\}$. For example, when there is no noise, $\sigma_{\text{NSR}} = 0$ (the lowest blue curve), it gives the lowest recovery error (compared to other colored curves) over the widest range of allowable $\mathcal{T}$. There is a wide range of $\mathcal{T}$ that yields the same recovery. We

use $\mathcal{T} = 0.2$ for the KS and PM equations, by choosing a value of $\mathcal{T}$ from a large plateau. This makes the algorithm more robust. In general, since the colored curves are decreasing functions in terms of $\mathcal{T}$, if the given data is highly corrupted by noise, using a larger $\mathcal{T}$ can help with the identification.

4.5.3 Effects of subsampling

In Figure 4.15, we show the effects of changing the final time T (the top row), and of changing Δx and Δt for $\mathbb{N}_x\mathbb{N}_t$ (the second row), and of changing the uniform subsampling in (Equation (4.11)), i.e., Δt^* and Δx^* for the generation of the feature matrix (the third row). We compare for the KS equation (Equation 4.34), the 2D linear ODE system (Equation 4.39), the Van der Pol equation (Equation 4.40), and the Duffing equation (Equation 4.41) to illustrate the effects. The noise level is $\sigma_{\text{NSR}} = 0.1$ for each example. We present the average of the E_2 error , the TPR and PPV values from 20 independent experiments for one varying variable among the variables $\{T, \Delta t, \Delta x, \Delta t^*, \Delta x^*\}$ while fixing the rest. The first row shows that the recovery by WeakIdent is robust as long as T is above a sufficiently large value (e.g. 100 or 10), which indicates that there is a time T such that the solution of the differential equations contains enough dynamics up to time T. The second row shows that WeakIdent gives a smaller error with smaller Δx and Δt. The bottom row shows that the size of uniform subsampling in space and time of the feature matrix does not affect the recovery.

In Table 4.4, we show an example of the size reduction from $\mathbf{W}$ to $\mathbf{W}_{\text{narrow}}$for the PDEs and ODEs considered in this chapter. We use $\sigma_{\text{NSR}} = 0.1$ for the RD equation (Equation 4.37) and $\sigma_{\text{NSR}} = 0.2$ for the rest of the equations. The given data is of size $\mathbb{N}_x\mathbb{N}_t$ and it is subsampled to $H = N_xN_t$ number of rows for $\boldsymbol{W}$. The narrow-fit further reduces the feature matrix to $\tilde{\mathbf{W}}_{\text{narrow}}$ for computational accuracy.

4.5.4 Speed of WeakIdent

We perform experiments using Matlab on the Apple M1 processor with 8-core CPU and 16GB of RAM. The computational cost of WeakIdent is typically about 1-5 seconds for an ODE system or a PDE with one dependent variable in a 1D spatial domain. For example, the cpu times to recover the Lotka-Volterra system (Equation 4.42) and the KdV equation (Equation 4.33) are 1.11 and 0.63 seconds, respectively. For the cases in 2D spatial domains, such as the anisotropic PM equation (Equation 4.36) with one variable, and the 2D reaction-diffusion equation (Equation 4.37) with two variables, the recovery can take about 3 and 35 seconds, respectively. The speed is comparable with WPDE[36], which takes 16 and 75 seconds for these two examples. We note that the main difference in computation comes from using modified sequential thresholding least-squares (MSTLS) and Subspace Pursuit as in this chapter. For the methods using MSTLS, thresholding lease square is performed for a large number (e.g., 50) of different λ (a parameter in MSTLS) to seek for a good threshold, so that the solutions are computed many times, while SP doesn't require to do this. For the computational cost scaling as the number of feature increases, with the trimming step, as in the case of Figure 4.4, typical examples converged to the correct support for a smaller sparsity then L. One may be able to stop SP after a reasonable sparsity k is reached to reduce unnecessary computation.

In Appendix B, we present additional results and more comparisons. The additional results for PDEs are in subsubsection B.2.0.1 and additional results for ODEs are in subsubsection B.2.0.2. Details about how to construct test functions are given in Section B.2.1.

Equation		Parameters
Transport equation $\frac{\partial u}{\partial t} = -\frac{\partial u}{\partial x} + 0.05\frac{\partial^2 u}{\partial x^2}$	(4.32)	$L = 43$, $\bar{\alpha} = 6$, $\bar{\beta} = 6$, $[X_1, X_2] = [0, 1]$, $\Delta x = 0.039$, $T = 0.3$, $\Delta t = 0.001$ $u(x, 0) = \sin(4\pi/(1-T)x)^3 \cos(\pi/(1-T)x)$ for $x < 1 - T$, and 0 otherwise
Korteweg-de Vires (KdV) $\frac{\partial u}{\partial t} = -0.5u\frac{\partial u}{\partial x} - \frac{\partial^3 u}{\partial x^3}$	(4.33)	$L = 43$, $\bar{\alpha} = 6$, $\bar{\beta} = 6$,$[X_1, X_2] = [-\pi, \pi]$, $\Delta x = 0.0157$, $T = 0.006$, $\Delta t = 10^{-5}$ $u(x, 0) = 3.0 \times 25^2 * \text{sech}(0.5 \times (25 \times (x + 2.0)))^2$ $+3.0 \times 16^2 * \text{sech}(0.5 \times (16 * (x + 1.0)))^2$
Kuramoto-Sivashinsky (KS) $\frac{\partial u}{\partial t} = -u - \frac{\partial^2 u}{\partial x^2} - \frac{\partial^4 u}{\partial x^4}$	(4.34)	$L = 43$, $\bar{\alpha} = 6$, $\bar{\beta} = 6$, $[X_1, X_2] = [0, 100.53]$, $\Delta x = 0.3927$, $T = 150$, $\Delta t = 0.5$ $u(x, 0) = \cos(x/16)(1 + \sin(x/16))$.
Nonlinear Schrodinger (NLS) (1D) $\begin{cases} \frac{\partial u}{\partial t} &= 0.5\frac{\partial^2 v}{\partial x^2} + u^2 v + v^3 \\ \frac{\partial v}{\partial t} &= -0.5\frac{\partial^2 u}{\partial x^2} - uv^2 - u^3 \end{cases}$	(4.35)	$L = 190$, $\bar{\alpha} = 6$, $\bar{\beta} = 6$ $[X_1, X_2] = [-5, 5]$, $\Delta x = 0.0391$ $T = 3.1416$, $\Delta t = 0.0126$
Anisotropic Porous Medium (PM) (2D) $\frac{\partial u}{\partial t} = +0.3\frac{\partial^2 u^2}{\partial^2 y} - 0.8\frac{\partial^2 u^2}{\partial x \partial y} + \frac{\partial^2 u^2}{\partial^2 x}$	(4.36)	$L = 65$, $\bar{\alpha} = 4$, $\bar{\beta} = 4$ $[X_1, X_2] = [-5, 5]$, $\Delta x = 0.0503$ $T = 5$, $\Delta t = 0.0503$
Reaction-Diffusion (2d) $\begin{cases} \frac{\partial u}{\partial t} &= 0.1\frac{\partial^2 u}{\partial^2 x} + 0.1\frac{\partial^2 u}{\partial^2 t} + u + v^3 - uv^2 + u^2 v - u^3 \\ \frac{\partial v}{\partial t} &= 0.1\frac{\partial^2 v}{\partial^2 y} + 0.1\frac{\partial v^2}{\partial^2 x} + v - v^3 - uv^2 - u^2 v - u^3 \end{cases}$	(4.37)	$L = 155$, $\bar{\alpha} = 4$, $\bar{\beta} = 5$, $[X_1, X_2] = [-10, 10]$ $\Delta x = 0.0781$, $T = 9.9219$, $\Delta t = 0.0781$ $u(x, y, 0) = \tanh(\sqrt{x^2 + y^2}\cos\left(\theta(x + iy) - \pi\sqrt{x^2 + y^2}\right)$, $v(x, y, 0) = \tanh(\sqrt{x^2 + y^2}\sin\left(\theta(x + iy) - \pi\sqrt{x^2 + y^2}\right)$

Table 4.1: A list of PDEs considered in this paper. Here L is the total number of features, $\bar{\alpha}$ is the highest order of partial derivative, $\bar{\beta}$ is the highest degree used in f_l in (Equation 4.4), $[X_1, X_2]$ is the range of the spatial domain, T is the final time for simulation. Δx and Δt are the spatial and temporal increment of the given data. The set up of (Equation 4.33), (Equation 4.34), (Equation 4.35), (Equation 4.36), and (Equation 4.37) are identical to [36].

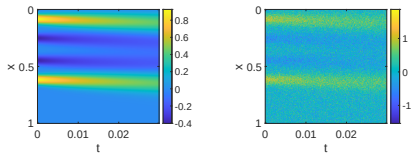
Name	Equation		parameters
2D Linear System	$\frac{d}{dt}\begin{bmatrix}x\\y\end{bmatrix} = \begin{bmatrix}-0.15 & 2.5\\ -2.5 & -0.15\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}$	(4.39)	$(x_0, y_0) = (2, 50)$, $\Delta t = 0.01, T = 10$ $L = 21, \bar{\beta} = 5$
2D Nonlinear (Van der Pol)	$\frac{d}{dt}\begin{bmatrix}x\\y\end{bmatrix} = \begin{bmatrix}0 & 1 & 0\\ 4 & -1 & -4\end{bmatrix}\begin{bmatrix}x\\y\\x^2y\end{bmatrix}$	(4.40)	$(x_0, y_0) = (0, 1)$ $\Delta t = 0.001, T = 15$ $L = 21, \bar{\beta} = 5$
2D Nonlinear (Duffing)	$\frac{d}{dt}\begin{bmatrix}x\\y\end{bmatrix} = \begin{bmatrix}0 & 1 & 0\\ -0.2 & -0.05 & -1\end{bmatrix}\begin{bmatrix}x\\y\\x^3\end{bmatrix}$	(4.41)	$(x_0, y_0) = (0, 2)$ $\Delta t = 0.01, T = 10$ $L = 21, \bar{\beta} = 5$
2D Nonlinear (Lotka-Volterra)	$\frac{d}{dt}\begin{bmatrix}x\\y\end{bmatrix} = \begin{bmatrix}0.67 & 0 & -1.33\\ 0 & -1 & 1\end{bmatrix}\begin{bmatrix}x\\y\\xy\end{bmatrix}$	(4.42)	$(x_0, y_0) = (10, 10)$ $\Delta t = 0.05, T = 50$ $L = 21, \bar{\beta} = 5$
3D Nonlinear (Lorenz)	$\frac{d}{dt}\begin{bmatrix}x\\y\\z\end{bmatrix} = \begin{bmatrix}-10.2 & 10.2 & 0 & 0 & 0\\ 29 & -1 & 0 & 0 & -1\\ 0 & 0 & -2 & 1 & 0\end{bmatrix}\begin{bmatrix}x\\y\\z\\xy\\xz\end{bmatrix}$	(4.43)	$(x_0, y_0, z_0) = (-8, 7, 10)$ $\Delta t = 0.001, T = 15$ $L = 20, \bar{\beta} = 3$

Table 4.2: A list of ODEs considered in this paper. This table includes the initial condition, the temporal increment Δt, the total simulation time T, the total number of features L and the highest degree of polynomials $\bar{\beta}$ in (Equation 4.4) for each equation. The Solution is simulated with RK45 with tolerance 10^{-10}.

Relative coefficient Error l_2	$$E_2 = \|\|\boldsymbol{c}^* - \boldsymbol{c}\|\|_2 / \|\|\boldsymbol{c}^*\|\|_2 \quad (4.45)$$
Relative coefficient Error l_∞	$$E_\infty = \max_l \{\|\boldsymbol{c}^*(l) - \boldsymbol{c}(l)\| / \|\boldsymbol{c}^*(l)\| : \boldsymbol{c}^*(l) \neq 0\} \quad (4.46)$$
True Positive Rate	$$\text{TPR} = \|\{l : \boldsymbol{c}^*(l) \neq 0,\ \boldsymbol{c}(l) \neq 0\}\| / \|\{l : \boldsymbol{c}^*(l) \neq 0\}\| \quad (4.47)$$
Positive Predictive Value	$$\text{PPV} = \|\{l : \boldsymbol{c}^*(l) \neq 0,\ \boldsymbol{c}(l) \neq 0\}\| / \|\{l : \boldsymbol{c}(l) \neq 0\}\| \quad (4.48)$$
Residual Error	$$E_{\text{res}} = \|\|\boldsymbol{W}\boldsymbol{c} - \boldsymbol{b}\|\|_2 / \|\|\boldsymbol{b}\|\|_2 \quad (4.49)$$
Dynamic Error	$$E_{\text{dyn}} = \sum_{1 \leq i \leq \mathbb{N}_x, 1 \leq n \leq \mathbb{N}_t} (\|U^n_{\text{i,forward}} - U^n_{\mathbf{i},clean}\|^2) / (\mathbb{N}_x \mathbb{N}_t) \quad (4.50)$$

Table 4.3: Error measurements used for comparisons.

(a) U (d) $\hat{U}$



(b) $\sigma_{\text{NSR}} = 0$

	WeakIdent	WPDE [36]	RGG[38]	IDENT[29]	SC[30]	ST[30]
E_2	**0.001**	**0.001**	**0.001**	-	2.26	2.24
E_∞	**0.001**	**0.001**	**0.001**	-	-	3.08
E_{res}	**0.001**	**0.001**	**0.001**	0.98	0.03	0.03
TPR	**1.0**	**1.0**	**1.00**	-	0.00	0.50
PPV	**1.0**	**1.0**	**1.00**	0.00	0.00	0.20

(c) $\sigma_{\text{NSR}} = 0$

True equation	$u_t = -1.00000u_x + 0.05000u_{xx}$
WeakIdent	$\mathbf{u_t = -1.00145u_x + 0.04999u_{xx}}$
WPDE [36]	$\mathbf{u_t = -1.00144u_x + 0.05000u_{xx}}$
RGG [38]	$\mathbf{u_t = -1.00119u_x + 0.04999u_{xx}}$
IDENT[29]	$u_t = -0.0006 + 0.0036u + 0.0244u^2 - 0.9992u_x + 0.0004(u_x)^2 + ...$
SC[30]	$u_t = +1.74039u^2 - 1.03236u_x + 0.05168u_{xx} + 0.00298uu_{xx}$
ST[30]	$u_t = +1.73061u^2 - 1.01121u_x - 0.10390uu_x + 0.05167u_{xx} + 0.00298uu_{xx}$

(e) $\sigma_{\text{NSR}} = 1$

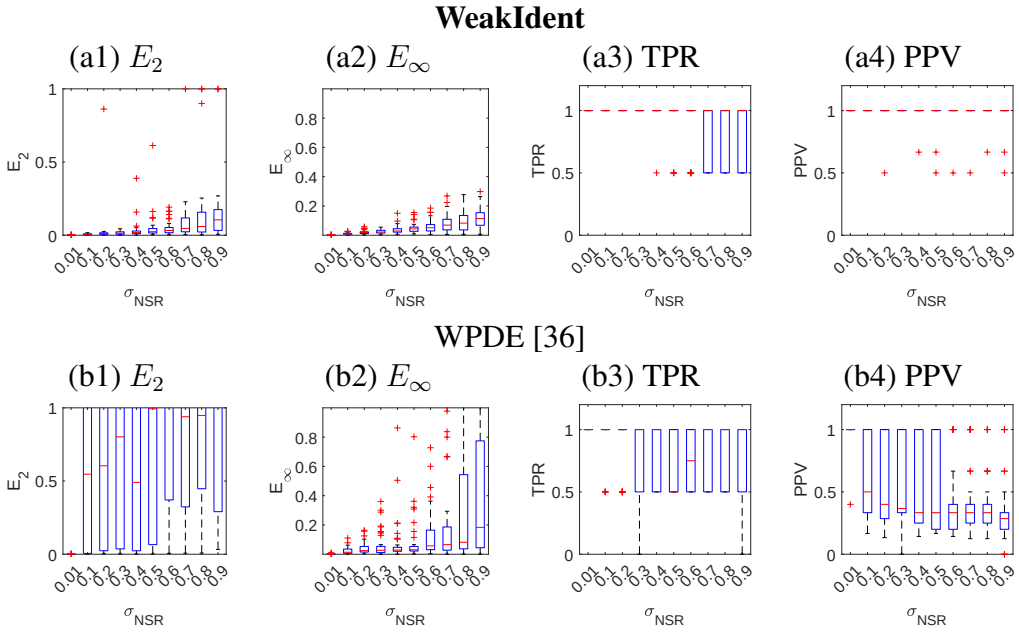
	WeakIdent	WPDE [36]	RGG [38]	IDENT [29]	SC [30]	ST[30]
E_2	**0.008**	0.184	135.33	-	17.43	20.32
E_∞	**0.008**	1.129	0.13	-	-	18.23
E_{res}	**0.811**	0.830	0.95	0.82	0.91	0.89
TPR	**1.0**	**1.0**	0.50	0.00	0.00	0.50
PPV	**1.0**	0.5	0.25	0.00	0.00	0.20

(f) $\sigma_{\text{NSR}} = 1$

True equation	$u_t = -1.00000u_x + 0.05000u_{xx}$
WeakIdent	$\mathbf{u_t = -1.00792u_x + 0.05029u_{xx}}$
WPDE [36]	$u_t = -1.02983u_x + 0.10647u_{xx} - 0.15741(u^3)_{xx} + 0.07197(u^6)_{xx}$
RGG [38]	$u_t = -0.00003u_{xxxx} - 0.87531u_x + 44.70146u^2 - 127.91622u^3$
IDENT[29]	$u_t = -0.2710 + 6.1120u - 3.4346u^2 - 0.0000u_x + 0.0000(u_x)^2 + ...$
SC[30]	$u_t = -3.35704u - 17.09628u^2 - 0.26659u_x$
ST[30]	$u_t = +0.60609 - 4.44906u - 19.79311u^2 - 0.17997u_x - 0.86156uu_x$

For RGG [38], we use 8 default features $\{uu_x, u_{xx}, u_{xxxx}, u, u_x, u_{xxx}, u^2, u^3\}$ and the parameters $p_x = 4, p_t = 3, N_d = 100, D = (40, 20)$ are used. For IDENT [29], we use $\lambda = 200$ for the sparse regression algorithm, and the dictionary is set to be $\{1, u, u^2, u_x, u_x^2, uu_x, u_{xx}, u_{xx}^2, uu_{xx}, u_xu_{xx}\}$. SC and ST [30] use the same dictionary as IDENT. For SC, we use $\alpha = 100$ and for ST, we use $s = 20$ and $n = 5$. These parameters are from the original papers.

Figure 4.5: Transport equation with diffusion (Equation 4.32): clean data case in (a), (b) and (c), and noisy data with $\sigma_{\text{NSR}} = 100\%$ in (d), (e) and (f). WeakIdent is compared with WPDE [36], RGG [38], IDENT[29], SC[30], and ST[30]. The error measures are in Table (b) and (e) and the recovered equations are in (c) and (f).



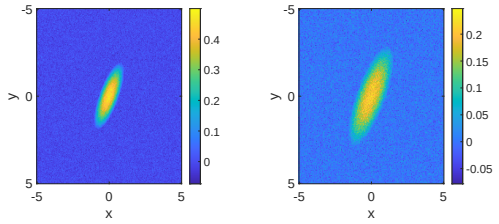
In each box-plot, the red line is the median, the lower bound is the 25% quantile, the upper bound is the 75% quantile, and $+$ signs represent outliers of each identification error. We use the same criteria for the box-plots in the rest of the figures.

Figure 4.6: Transport equation (Equation 4.32), statistical comparison between WeakIdent (the top row) and WPDE [36] (the second row). The errors E_2, E_∞, TPR and PPV are shown from 50 experiments for each $\sigma_{\text{NSR}} \in \{0.01, 0.1, 0.2, , ..., 0.9\}$ using box-plots. The E_2 and E_∞ errors by WeakIent are lower than the errors of WPDE, with less variations. The TPR and PPV by WeakIdent are closer to 1 with less variations as well.

Equation	$\mathbb{N}_x$	$\mathbb{N}_t$	N_x	N_t	$\boldsymbol{W}$ size$(H \times L)$	$\tilde{\boldsymbol{W}}_{\text{narrow}}$ size
(Equation 4.32)	257	300	36	39	1404 $\times$43	824 $\times$43
(Equation 4.33)	400	601	71	65	4615 $\times$43	1367 $\times$43
(Equation 4.34)	256	301	46	43	1935$\times$43	916$\times$43
(Equation 4.35)	256	251	39	42	1225$\times$190	159$\times$190
(Equation 4.36)	200$\times$200	128	14$\times$14	16	3136$\times$65	1349$\times$65
(Equation 4.37)	256$\times$256	201	13$\times$13	14	2366$\times$155	2271$\times$155
(Equation 4.39)	-	1001	-	851	877$\times$21	127$\times$21
(Equation 4.40)	-	15001	-	958	958$\times$21	295$\times$21
(Equation 4.41)	-	1001	-	915	915$\times$21	57$\times$21
(Equation 4.42)	-	1001	-	947	947$\times$10	338$\times$10
(Equation 4.43)	-	15001	-	983	983$\times$20	930$\times$20

Table 4.4: Typical examples of the feature matrix size and the reduction in narrow-fit. The given data is of size $\mathbb{N}_x\mathbb{N}_t$ and it is subsampled to $H = N_xN_t$ rows for $\boldsymbol{W}$. We use $\sigma_{\text{NSR}} = 0.1$ for the RD equation (Equation 4.37) and $\sigma_{\text{NSR}} = 0.2$ for the rest of the equations. For systems of equations, the size of the feature matrix for each dependent variable is identical.

(a) $\hat{U}(\boldsymbol{x}, 0)$ (b) $\hat{U}(\boldsymbol{x}, T)$



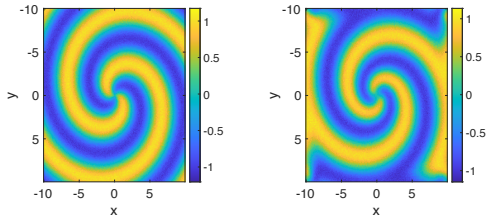
(c) $\sigma_{\text{NSR}} = 0.08$

True equation	$u_t = +0.30000(u^2)_{yy} - 0.80000(u^2)_{xy} + 1.00000(u^2)_{xx}$	
WeakIdent	$\mathbf{u_t = +0.29912(u^2)_{yy} - 0.79416(u^2)_{xy} + 0.99568(u^2)_{xx}}$	$E_2 = \mathbf{0.0056}$
WPDE	$\mathbf{u_t = +0.29928(u^2)_{yy} - 0.79362(u^2)_{xy} + 0.99512(u^2)_{xx}}$	$E_2 = \mathbf{0.0061}$
RGG[38]	$u_t = +0.00028 + 0.23457u - 6.28949u^2 + 21.31151u^3 - 0.36341(u^2)_x + 0.48573(u^2)_y - 0.12914(u^2)_{xx} + 1.34146(u^2)_{yy} - 1.18102(u^2)_{xy} + 0.03796u_x - 0.03006u_y + 0.06174u_{xx} - 0.04775u_{yy} + 0.06238u_{xy}$	$E_2 = 8.9376$

For RGG [38], we use a dictionary of 14 features $\{1, u, u^2, u^3, (u^2)_x, (u^2)_y, (u^2)_{xx}, (u^2)_{yy}, (u^2)_{xy}, u_x, u_y, u_{xx}, u_{yy}, u_{xy}\}$ adding the true features, and the parameters $p_x = 2, p_t = 1, N_d = 100$, and $D = (20, 10)$.

Figure 4.7: Anisotropic Porous Medium (PM) equation (Equation 4.36) on a 2-D spatial domain with cross derivative feature. We set $\sigma_{\text{NSR}} = 0.08$, which is equivalent to $\sigma_{\text{NR}} = 0.4139$ in WPDE [36]. (a) Given noisy data $\hat{U}(\boldsymbol{x}, 0)$ and (b) $\hat{U}(\boldsymbol{x}, T)$. (c) Identified equations with the E_2 error.

(a) $\hat{U}(\boldsymbol{x}, 0)$ (b) $\hat{U}(\boldsymbol{x}, T)$



(c) $\sigma_{\text{NSR}} = 0.08$

True equation	$u_t = +v^3 + u + 0.1u_{yy} + 0.1u_{xx} - uv^2 + u^2v - u^3$ $v_t = v + 0.1v_{yy} + 0.1v_{xx} - v^3 - uv^2 - u^2v - u^3$	
WeakIdent	$u_t = +0.99213v^3 + 0.98572u + 0.09660u_{yy} + 0.09695u_{xx} - 0.93229uv^2 + 0.97678u^2v - 0.99018u^3$ $v_t = +0.97792v + 0.09662v_{yy} + 0.09636v_{xx} - 0.97161v^3 - 0.96468uv^2 - 0.95572u^2v - 0.99605u^3$	E_2 =**0.0316**
WPDE	$u_t = +1.34525v^3$ $v_t = -1.34499u^3$	E_2 =0.9081
RGG[38]	$u_t = +0.10204\nabla u + 1.02296u - 1.01966u^3 + 1.01341v^3 + 1.03003u^2v - 1.01767uv^2$ $v_t = +0.09244\nabla v - 0.07400u - 0.93640u^3 + 0.95099v - 0.95370v^3 - 0.95450u^2v - 0.93750uv^2$	E_2 =0.0793

For RGG [38], the provided default features for reaction-diffusion type equation in [38] is used: for u, the dictionary is $\{\nabla u, u, u^2, u^3, v, v^2, v^3, uv, u^2v, uv^2\}$ and for v, the dictionary is $\{\nabla v, u, u^2, u^3, v, v^2, v^3, uv, u^2v, uv^2\}$, and parameters $p_x = 2, p_t = 1, N_d = 100, D = (20, 10)$.

Figure 4.8: Reaction-diffusion equation (Equation 4.37) on a 2D spatial domain with $\sigma_{\text{NSR}} = 0.08$ (equivalent to $\sigma_{\text{NR}} = 0.08$ defined in [36]). (a) Given noisy data $\hat{U}(\boldsymbol{x}, 0)$ and (b) $\hat{U}(\boldsymbol{x}, T)$. (c) The identified equations and the E_2 errors. WeakIdent finds the correct terms with a small coefficient error.

(a1) E_2 (a2) E_∞ (a3) TPR (a4) PPV

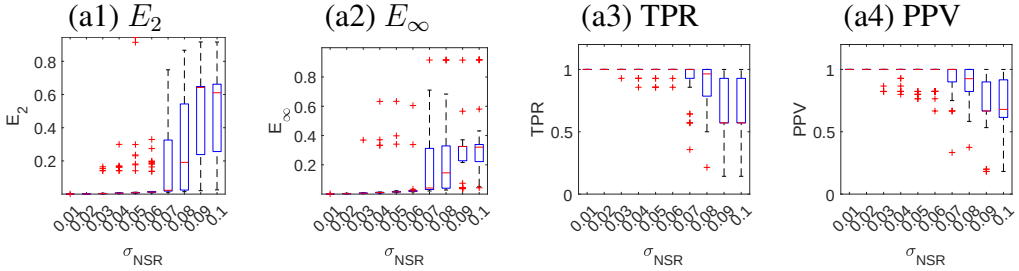


Figure 4.9: The Identification results from WeakIdent for the reaction diffusion equation (Equation 4.37): The E_2, E_∞ errors , TPR and PPV are shown from 50 experiments for each $\sigma_{\text{NSR}} \in \{0.01, 0.02, , ..., 0.1\}$ using box-plots.

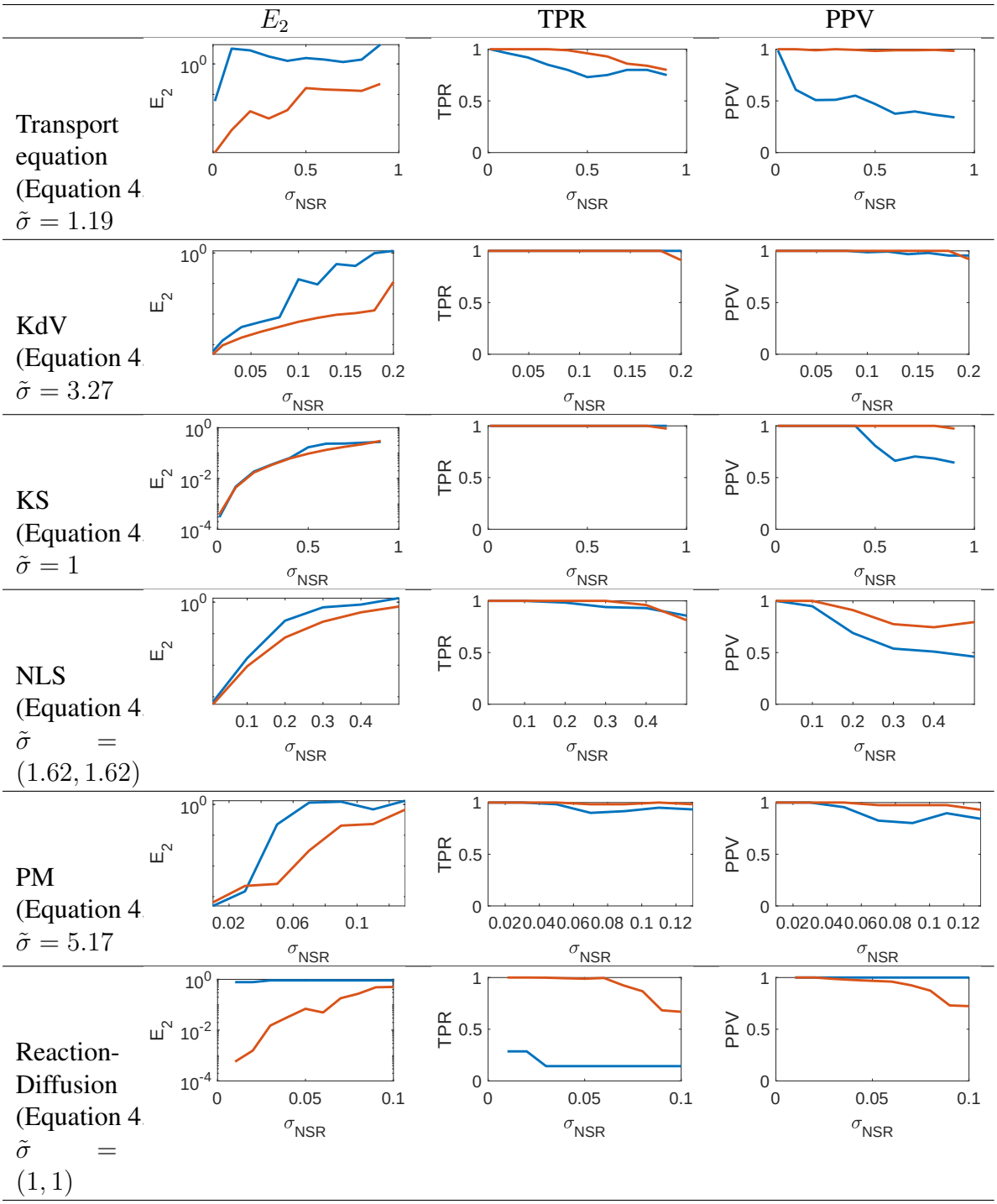
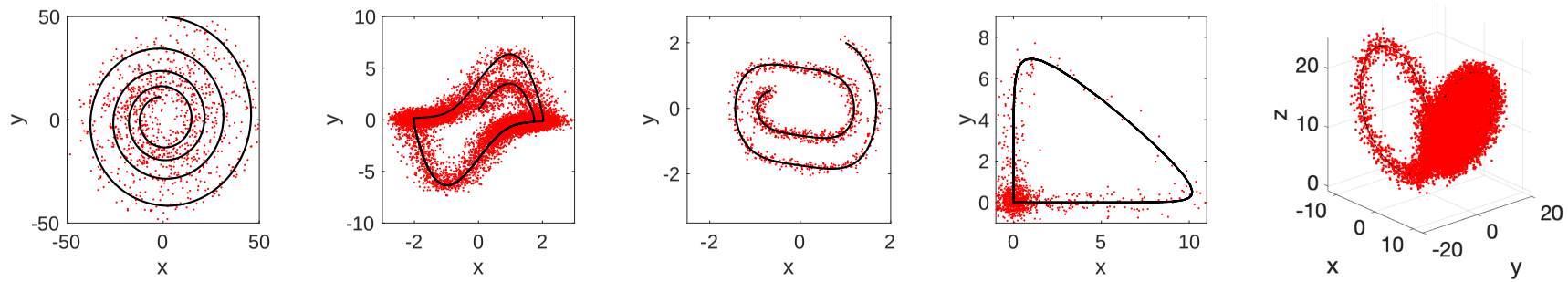
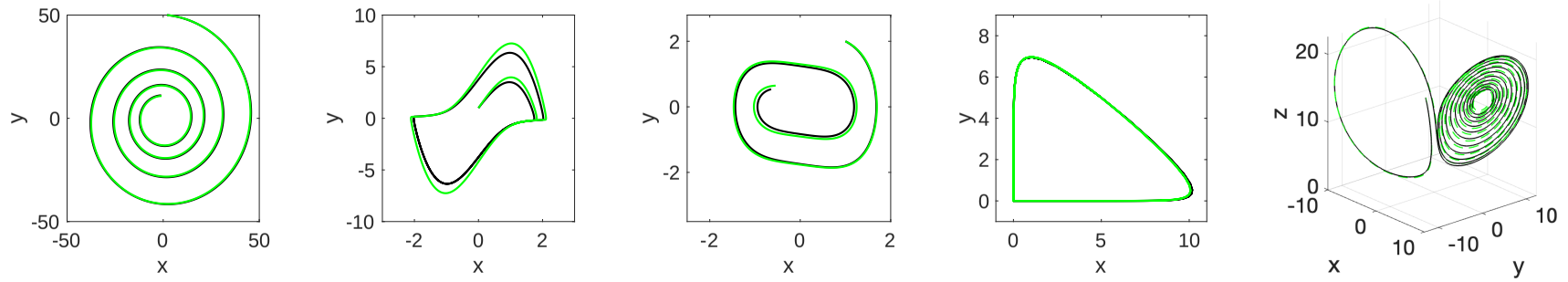


Figure 4.10: The identified PDEs in Table 4.1 for different noise levels. We compare WeakIdent (Red) and WPDE (Blue). The x-axis is σ_{NSR}, while the y-axis is the average E_2 error, TPR and PPV over 50 experiments. The relative noise ratio $\tilde{\sigma} = \sigma_{\text{NSR}}/\sigma_{\text{NR}}$ compares our noise level σ_{NSR} vs. σ_{NR} in [36]. We present results for the transport equation (Equation 4.32), the KdV equation (Equation 4.33), the KS equation (Equation 4.34), the NLS equation (Equation 4.35), the PM equation (Equation 4.36), and the reaction-diffusion (2D) equation (Equation 4.37). The noise-to-signal ratio σ_{NSR} ranges in $\{0, 0.1, 0.2, ..., 0.9\}$, $\{0.01, 0.02, 0.04, ..., 0.24\}$, $\{0, 0.1, 0.2, ..., 0.9\}$, $\{0.01, 0.1, 0.2, ..., 0.5\}$, $\{0.01, 0.03, 0.05, ..., 0.15\}$, and $\{0.01, 0.02, ..., 0.1\}$ for each equation respectively.

(a) Equ. (Equation 4.39) (b) Equ. (Equation 4.41) (c) Equ. (Equation 4.40) (d) Equ. (Equation 4.42) (e) Equ. (Equation 4.43)



(f) Equ. (Equation 4.39) (g) Equ. (Equation 4.41) (h) Equ. (Equation 4.40) (i) Equ. (Equation 4.42) (j) Equ. (Equation 4.43)



	σ_{NSR}	E_2	E_∞	E_{res}	E_{dyn}	TPR	PPV
(f) 2D Linear System (Equation 4.39)	0.2	0.006	0.063	0.061	0.484	1	1
(g) Van der Pol (Equation 4.41)	0.2	0.055	0.080	0.042	0.120	1	1
(h) Duffing (Equation 4.40)	0.2	0.088	0.204	0.048	0.156	0.75	1
(i) Lotka-Volterra (Equation 4.42)	0.1	0.013	0.023	0.120	0.365	1	1
(j) Lorenz (Equation 4.43)	0.2	0.011	0.041	0.032	2.380	1	1

Figure 4.11: WeakIdent results for ODE systems in Table 4.2. (a)-(e): Given noisy data compared to the true dynamics. (f)-(j): Recovered systems via WeakIdent using true initial conditions. WeakIdent recovers the dynamics close to the true dynamics with a small identification error.

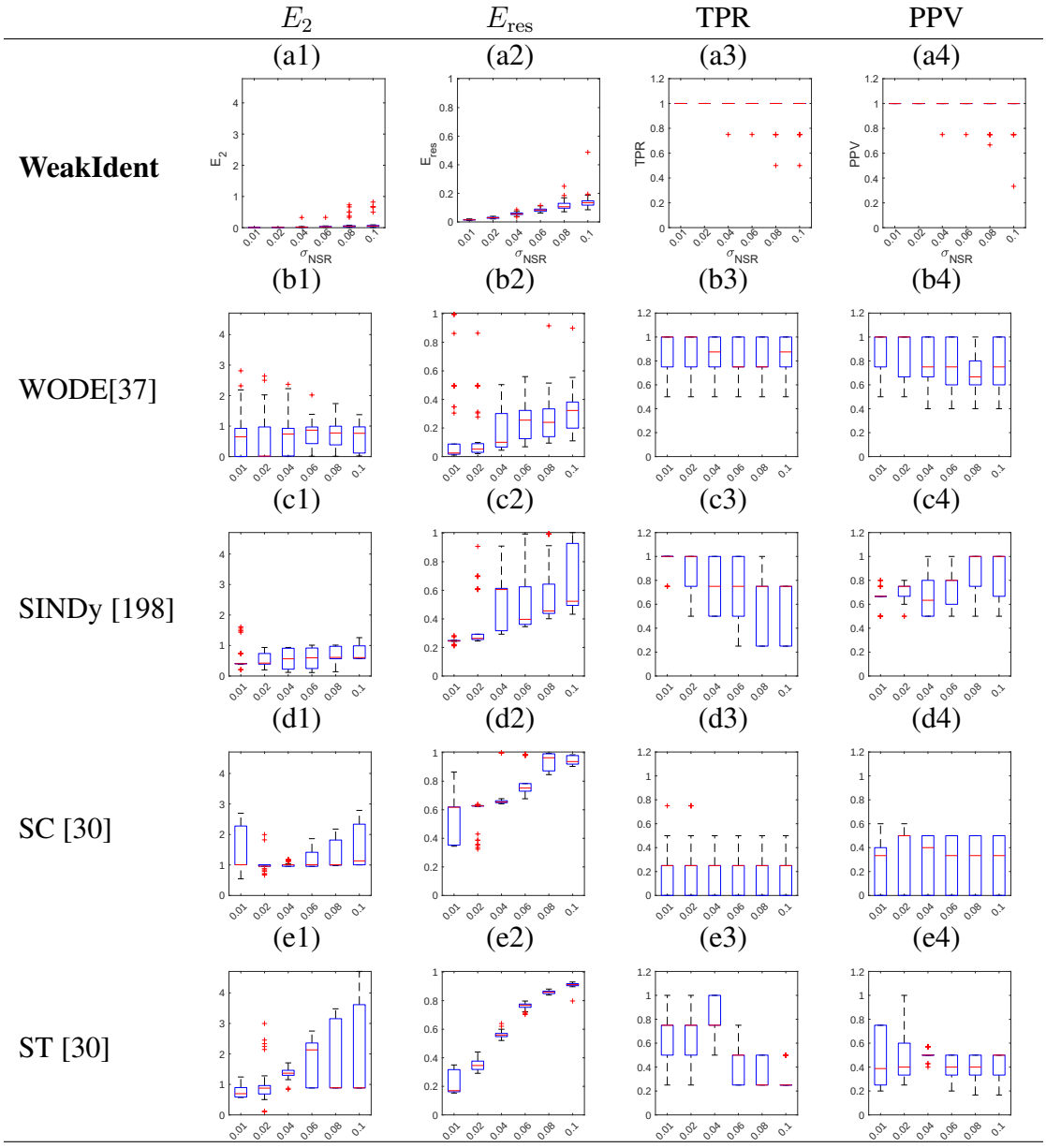


Figure 4.12: The Lotka-Volterra equation (Equation 4.42). Statistical comparisons between (a1)-(a4) WeakIdent, (b1)-(b4) WODE [37], (c1)-(c4) SINDy[198], (d1)-(d4) SC[30] and (e1)-(e4) ST[30]. The E_2, E_{res} errors, TPR and PPV are shown from 50 experiments for each $\sigma_{\text{NSR}} \in \{0.01, 0.02, , ..., 0.1\}$ using box-plots. Notice that for WeakIdent, the E_2 error is lower with less variations, and the TPR and PPV are closer to 1 as compared with that obtained from other methods.

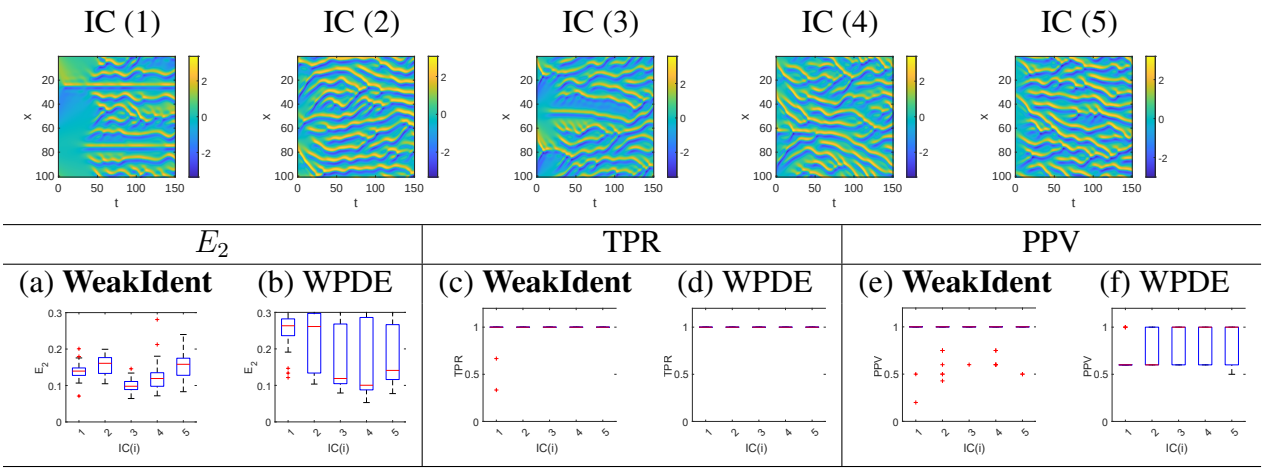


Figure 4.13: The KS equation (Equation 4.34) using five different initial conditions (1)-(5) with the noisy level of $\sigma_{\text{NSR}} = 0.6$. In (a)-(f) the x-axis is the index of initial conditions (1)-(5). For each initial condition, the box plot represents the statistical results over 50 experiments. WeakIdent gives a smaller E_2 error, and PPV is closer to 1 with less variations.

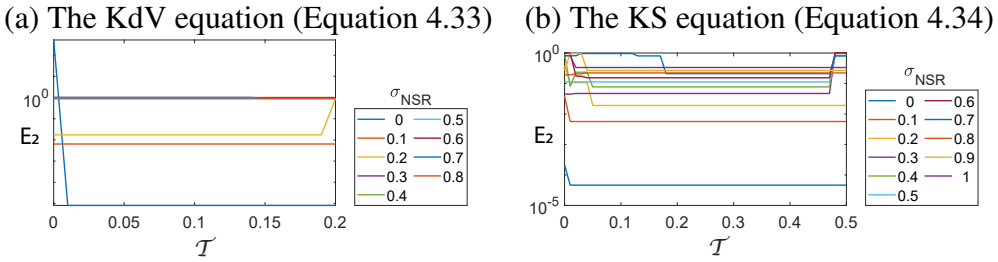


Figure 4.14: The coefficient E_2 error (y-axis) versus the trimming parameter $\mathcal{T}$ (x-axis) for the identification of (a) the KdV equation (Equation 4.33) and (b) the KS equation (Equation 4.34). Different color curves represent results for various noise-to-signal ratios $\sigma_{\text{NSR}} \in \{0, 0.1, ..., 1\}$. Notice a wide range of $\mathcal{T}$ gives the same recovery.

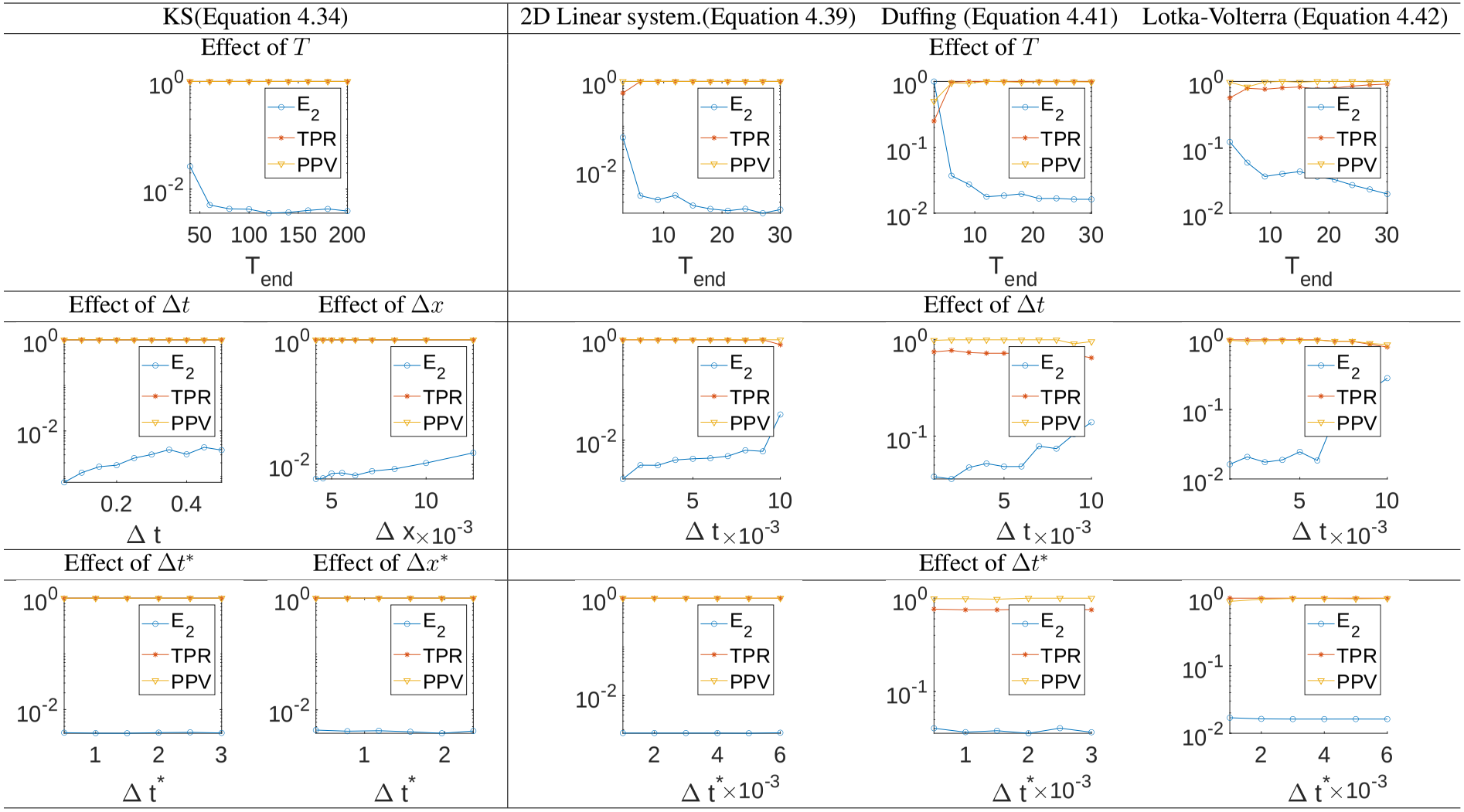


Figure 4.15: Effects of the final time T (the top row), $\Delta t, \Delta x$ for $\mathbb{N}_x\mathbb{N}_t$ of the given data (the second row), and subsampling $\Delta t^*, \Delta x^*$ in (Equation 4.11) in the third row. Each graph shows the average of the E_2 error, the TPR and PPV values over 20 experiments for one varying variable among the variables in $\{T, \Delta x, \Delta t, \Delta x^*, \Delta t^*\}$ while the rest is fixed. The noise level is $\sigma_{\text{NSR}} = 0.1$. The left column gives the PDE results for the KS equation while both $\Delta t, \Delta x$ are shown. The right columns show Δt only for ODEs, including the 2D Linear system (Equation 4.39), the Duffing equation (Equation 4.41) and the Lotka-Volterra equation(Equation 4.42). There is a transition point in T such that the given data up to T contain enough dynamics. The recovery is in general better with smaller Δt and Δx, and the rate of uniform subsampling has a minimal effect on the results.

CHAPTER 5

FOURIERIDENT - IDENTIFYING DIFFERENTIAL EQUATIONS PART II (FREQUENCY DOMAIN)

This chapter studies this inverse problem of identifying a differential equation from a single trajectory of noisy observations. In particular, we consider an identification in the frequency domain

$$\mathcal{F}\left\{\frac{\partial u(x,t)}{\partial t}\right\} = \mathcal{F}\{G(u)\} = \mathcal{F}\{\sum_{l=1}^{L} c_l g_l(x,t)\}, \tag{5.1}$$

where $\mathcal{F}$ represents *Fourier transform*. The $G(u)$ is the governing equation which is assumed to be a linear combination of linear and nonlinear features of u, e.g. $g_l(x,t)$s are monomials u^β and spatial derivative of monomials $\frac{\partial^\alpha}{\partial x^\alpha}(u^\beta)$, where α, β are nonnegative integers. We consider one variable partial differential equation. Our objective is to find the coefficient vector $\boldsymbol{c} = (c_l) \in \mathbb{R}^L$; the support and value of $\boldsymbol{c}$ for identifying the governing differential equation. In this chapter, we present the challenges in working with data in the frequency domain and propose a robust framework, FourierIdent, for differential equation identification. We propose to use Fourier features for IDENTifying differential equations (FourierIdent). After taking the Fourier transform of (Equation 5.1), we perform model selection and parameter estimation in the frequency domain, which is different from existing works on model identification in the physical domain. This is motivated by the weak formulation of features used in [37, 3], where the integral forms are calculated through convolution and evaluated in the frequency domain using FFT. In [37, 3], the rest of computation is done on the physical domain using inverse FFT.

The contributions of this chapter are summarized as follows:

1. We propose a stable denoising methods on frequency domain to effectively han-

dle large frequency magnitudes and to reduce the sensitivity against noise. We use Fourier analysis and fit the correct decay to the coefficients, and define the core region of features for identification. Using FFT gives the efficiency to the algorithm.

2. We develop a comprehensive framework, FourierIdent, to identify the coefficients of the differential equation from a single observation. Within this framework, we use Subspace Pursuit and group trimming to find the coefficient support for each sparsity level. We further consider core regions of features to refine the coefficient value identification and identify the optimal fit for the given data.

3. FourierIdent shows benefits in identifying equations under high level of noise, also when the realization of equation, the given data, has a complex pattern, e.g., many different frequency modes. We present various numerical experimental results for comparison.

This chapter is organized as follows: In section 5.1, we provide a short literature review of identifying differential equations in the frequency domain. In section 5.2, we provide the problem setup and the formulation of Fourier features, and the error analysis of using a Fourier feature is discussed in subsection 5.2.1. In section 5.3, we propose the denoising method for Fourier features and define the core region of feature in the frequency domain. In section 5.4, we present the methodology of FourerIdent, using Subspace Pursuit, group trimming and the new enery using the core region of features for coefficient identification. In section 5.5 and section 5.6, we provide details of implementation and numerical results of FourierIdent. We conclude the paper with remarks in section 5.7.

5.1 Literature review

Parameter estimation for differential equations in the frequency domain has been considered in science and engineering applications. In [199], a sample maximum likelihood estimator in the frequency domain is used to identify some spatially dependent parameters

in a parabolic equation, from the given PDE form. In [200], a frequency domain identification technique is proposed to estimate Linear Parameter-Varying differential equations with weighted nonlinear least squares. [201] examines how physics-informed neural networks successively capture different frequencies of the solution that the low-frequency components is captured at the beginning of training then the high-frequency components as the training process proceeds. The authors in [202] consider cutting the given data in the frequency domain after numerical differentiation of NN-denoised data in the model identification problem. The authors in [203] provide a mathematical theory on the possibility of learning an PDE from a single solution trajectory. While frequencies have already been considered important in some particular identification methods in the literature, no general framework is available to identify an unknown partial differential equation using frequency responses.

5.2 Problem set-up and Fourier features

In this paper, we present FourierIdent on one-dimensional PDEs, but our proposed method can be extended to high-dimensional PDEs. Suppose the physical domain is $\Omega = [0, X] \times [0, T]$ with $X > 0, T > 0$, and data are sampled on a uniform grid with step size Δx, and Δt. We denote $x_i = i\Delta x, t^n = n\Delta t$, and the PDE solution at (x_i, t^n) as $U_i^n = u(x_i, t^n)$. Our observation of the solution at (x_i, t^n) is contaminated by noise such that

$$U_{noise,i}^n = U_i^n + \epsilon_i^n \text{ for } (x_i, t^n) \in \Omega, \tag{5.2}$$

where ϵ_i^n represents zero-mean noise at each point (x_i, t^n). We denote the given data as

$$\mathcal{D} = \{U_{noise,i}^n | i = 0, 1, 2, ..., \mathbb{N}_x - 1, n = 0, 1, ..., \mathbb{N}_t - 1\} \in \mathbb{R}^{\mathbb{N}_x \times \mathbb{N}_t}, \tag{5.3}$$

where $\mathbb{N}_x$ and $\mathbb{N}_t \in \mathbb{N}$ are the discretization sizes in the spatial and temporal dimensions. We assume that the underlying equation has the form of (Equation 5.1):

$$\frac{\partial u}{\partial t}(x,t) = \sum_{l=1}^{L} c_l g_l = \sum_{l=1}^{L} c_l \frac{\partial^{\alpha_l} f_l(u)}{\partial x^{\alpha_l}}, \text{ with } f_l(u) = u^{\beta_l}, \tag{5.4}$$

where L is the total number of features in the dictionary which consists of feature terms as $\left\{\frac{\partial^{\alpha_l} f_l(u)}{\partial x^{\alpha_l}}\right\}_{l=1}^{L}$. The l-th feature $\frac{\partial^{\alpha_l} u^{\beta_l}}{\partial x^{\alpha_l}}$ is the α_l-order derivative of the monomial u^{β_l} where α_l, β_l are nonnegative integers. The objective of this paper is to find the support and values of this coefficient vector in (Equation 5.4)

$$\boldsymbol{c} = [c_1, c_2, \cdots, c_L]^\top$$

from the given data $\mathcal{D}$.

Since we consider PDEs in the frequency domain, we assume that u is a periodic function in x and t. For non-periodic functions, we extend the function to a periodic function, which is to be discussed in subsection 5.5.1. We define the Fourier transform of $u(x,t)$ as:

$$\mathcal{F}(u)[\xi_x, \xi_t] = \int_0^T \int_0^X u(x,t) e^{-(\frac{\xi_x x}{X} + \frac{\xi_t t}{T})2\pi\sqrt{-1}} dx dt.$$

The Fourier transform of (Equation 5.4) with respect to x and t becomes

$$\begin{aligned}
&\frac{2\pi}{T}\xi_t\sqrt{-1}\mathcal{F}(u)[\xi_x, \xi_t] \\
=&\sum_{l=1}^{L} c_l \left(\frac{2\pi}{X}\xi_x\sqrt{-1}\right)^{\alpha_l} \mathcal{F}(f_l)[\xi_x, \xi_t] \\
=&\left[(\tfrac{2\pi}{X}\xi_x\sqrt{-1})^{\alpha_1}\mathcal{F}(f_1(u)) \quad \cdots \quad (\tfrac{2\pi}{X}\xi_x\sqrt{-1})^{\alpha_l}\mathcal{F}(f_2(u)) \quad \cdots \quad (\tfrac{2\pi}{X}\xi_x\sqrt{-1})^{\alpha_L}\mathcal{F}(f_L(u))\right]\mathbf{c}.
\end{aligned} \tag{5.5}$$

The first term in (Equation 5.5) is the Fourier feature of u_t, and $(\frac{2\pi}{X}\xi_x\sqrt{-1})^{\alpha_l}\mathcal{F}(f_l(u))$ in

the right of (Equation 5.5) is that of the l-th feature in the library, i.e., $\frac{\partial^{\alpha_l} u^{\beta_l}}{\partial x^{\alpha_l}}$.

Our idea is to find the support and values of the coefficient vector $\mathbf{c}$ from (Equation 5.5) at the frequency modes (ξ_x, ξ_t) for $\xi_x = 0, 1, ..., \mathbb{N}_x - 1, \xi_t = 0, 1, 2..., \mathbb{N}_t - 1$. We denote $h = \mathcal{H}(\xi_x, \xi_t)$ as the index of the frequency mode (ξ_x, ξ_t), where $\mathcal{H}$ is a map from the frequency mode (ξ_x, ξ_t) to the unique index $h \in \{1, 2, \ldots, H\}$ with $H = \#\{(\xi_x, \xi_t) : \xi_x = 0, 1, ..., \mathbb{N}_x - 1, \xi_t = 0, 1, ..., \mathbb{N}_t - 1\} = \mathbb{N}_x \mathbb{N}_t$. The notation $\#$ denotes the cardinality of a set.

FourierIdent aims to solve **a discrete Fourier system** of (Equation 5.5)

$$\boldsymbol{F}\boldsymbol{c} = \boldsymbol{b}, \tag{5.6}$$

where

$$\boldsymbol{F} = (a_{h,l}) \in \mathbb{C}^{H \times L}, \quad \boldsymbol{c} = (c_l) \in \mathbb{R}^L, \quad \boldsymbol{b} = (b_h) \in \mathbb{C}^H,$$

for

$$a_{h,l} = \left(\frac{2\pi\xi_x}{X}\sqrt{-1}\right)^{\alpha_l} \sum_{p=0}^{\mathbb{N}_x - 1} \sum_{q=0}^{\mathbb{N}_t - 1} \left\{ e^{-\left(\frac{2\pi p}{\mathbb{N}_x}\xi_x + \frac{2\pi q}{\mathbb{N}_t}\xi_t\right)\sqrt{-1}} \left(U^q_{noise,p}\right)^{\beta_l} \right\} \Delta x \Delta t$$

and

$$b_{h,l} = \left(\frac{2\pi\xi_t}{T}\sqrt{-1}\right) \sum_{p=0}^{\mathbb{N}_x - 1} \sum_{q=0}^{\mathbb{N}_t - 1} \left\{ e^{-\left(\frac{2\pi p}{\mathbb{N}_x}\xi_x + \frac{2\pi q}{\mathbb{N}_t}\xi_t\right)\sqrt{-1}} U^q_{noise,p} \right\} \Delta x \Delta t.$$

Here $\Delta x = \frac{X}{\mathbb{N}_x}, \Delta t = \frac{T}{\mathbb{N}_t}$. The l-th column of $\boldsymbol{F}$ is the discrete Fourier feature associated with the l-th feature in the dictionary, and the vector $\boldsymbol{b}$ contains the discrete Fourier feature for u_t.

5.2.1 Error analysis for Fourier feature

We analyze the error of the linear system (Equation 5.6) with discrete Fourier features. Assume that $\{\epsilon_i^n\}$ in (Equation 5.3) are *i.i.d* and have zero-mean such that $\mathbb{E}(\epsilon_i^n) = 0$. Denote the true coefficient vector for the underlying PDE by $\boldsymbol{c}$, and the support of the true coefficient vector by $\mathrm{Supp}^* = \{l : \ c_l \neq 0\}$. The true coefficients satisfy the following equation :

$$\frac{2\pi}{T}\xi_t\sqrt{-1}\mathcal{F}(u)[\xi_x,\xi_t] = \sum_{l\in\mathrm{Supp}^*} c_l\left(\frac{2\pi}{X}\xi_x\sqrt{-1}\right)^{\alpha_l}\mathcal{F}(f_l)[\xi_x,\xi_t], \tag{5.7}$$

for $xi_x = 0, 1, ..., \mathbb{N}_x - 1, \xi_t = 0, 1, ..., \mathbb{N}_t - 1$. The L_∞-residual of (Equation 5.6) consists of two errors: one is the discretization error of Fourier features and the other is the error from noise:

$$e = \|\boldsymbol{F}\boldsymbol{c} - \boldsymbol{b}\|_\infty \leq e_{\text{Fourier}} + e_{\text{noise}}, \tag{5.8}$$

where

$$\begin{aligned} e_{\text{Fourier}} =& \max_{\xi_x,\xi_t}\left|\sum_{l\in\mathrm{Supp}^*} c_l\left(\frac{2\pi\xi_x}{X}\sqrt{-1}\right)^{\alpha_l}\sum_{p=0}^{\mathbb{N}_x-1}\sum_{q=0}^{\mathbb{N}_t-1}\left\{e^{-\left(\frac{2\pi p}{\mathbb{N}_x}\xi_x+\frac{2\pi q}{\mathbb{N}_t}\xi_t\right)\sqrt{-1}}\left(U_p^q\right)^{\beta_l}\right\}\right. \\ &\left. -\left(\frac{2\pi\xi_t}{T}\sqrt{-1}\right)\sum_{p=0}^{\mathbb{N}_x-1}\sum_{q=0}^{\mathbb{N}_t-1}\left\{e^{-\left(\frac{2\pi p}{\mathbb{N}_x}\xi_x+\frac{2\pi q}{\mathbb{N}_t}\xi_t\right)\sqrt{-1}}U_p^q\right\}\right|\frac{XT}{\mathbb{N}_x\mathbb{N}_t} \end{aligned} \tag{5.9}$$

$$\begin{aligned} e_{\text{noise}} =& \max_{\xi_x,\xi_t}|e_{\text{noise}}[\xi_x,\xi_t]| \\ =& \max_{\xi_x,\xi_t}\left|\sum_{l\in\mathrm{Supp}^*} c_l\left(\frac{2\pi\xi_x}{X}\sqrt{-1}\right)^{\alpha_l}\sum_{p=0}^{\mathbb{N}_x-1}\sum_{q=0}^{\mathbb{N}_t-1}\left\{e^{-\left(\frac{2\pi p}{\mathbb{N}_x}\xi_x+\frac{2\pi q}{\mathbb{N}_t}\xi_t\right)\sqrt{-1}}\left(\left(U_{noise,p}^q\right)^{\beta_l}-\left(U_p^q\right)^{\beta_l}\right)\right\}\right. \\ &\left. -\left(\frac{2\pi\xi_t}{T}\sqrt{-1}\right)\sum_{p=0}^{\mathbb{N}_x-1}\sum_{q=0}^{\mathbb{N}_t-1}\left\{e^{-\left(\frac{2\pi p}{\mathbb{N}_x}\xi_x+\frac{2\pi q}{\mathbb{N}_t}\xi_t\right)\sqrt{-1}}\epsilon_p^q\right\}\right|\frac{XT}{\mathbb{N}_x\mathbb{N}_t}. \end{aligned} \tag{5.10}$$

Here $e_{\text{noise}}[\xi_x,\xi_t]$ denotes the residual at the frequency mode (ξ_x,ξ_t).

In (Equation 5.8), we decompose the L_∞-residual e to e_{Fourier} and e_{noise}. The discretiza-

tion error of Fourier features, denoted by e_{Fourier}, decreases to 0 as the data sampling grid is refined:

$$\lim_{\Delta x,\Delta t\to 0} e_{\text{Fourier}} = \max_{\xi_x,\xi_t} \left| \sum_{l\in \text{Supp}^*} c_l \left(\frac{2\pi}{X}\xi_x\sqrt{-1}\right)^{\alpha_l} \mathcal{F}(u^{\beta_l})[\xi_x,\xi_t] - \frac{2\pi}{T}\xi_t\sqrt{-1}\mathcal{F}(u)[\xi_x,\xi_t] \right| = 0.$$

In the following, we prove an upper bound for the error e_{noise} resulted from noise.

Theorem 5.2.1. *Consider the differential equation in* (Equation 5.4) *whose Fourier form is* (Equation 5.7). *Assume that the given data $\mathcal{D}$ in* (Equation 5.3) *are contaminated by i.i.d. noise: $\{\epsilon_i^n\}$ are i.i.d bounded random variables with $\mathbb{E}[\epsilon_i^n] = 0$ and $|\epsilon_i^n| \le \epsilon$ for some $\epsilon > 0$. The area of Ω is denoted by $|\Omega| = XT$. Then the error e_{noise} satisfies*

$$e_{\text{noise}} \le \frac{XT}{\mathbb{N}_x\mathbb{N}_t} S\epsilon + \mathcal{O}(\epsilon^2), \tag{5.11}$$

where

$$S = \max_{\xi_x,\xi_t} \left| \sum_{p=0}^{\mathbb{N}_x-1}\sum_{q=0}^{\mathbb{N}_t-1} e^{-\left(\frac{2\pi p}{\mathbb{N}_x}\xi_x + \frac{2\pi q}{\mathbb{N}_t}\xi_t\right)\sqrt{-1}} \left(\sum_{l\in \text{Supp}^*} c_l \left(\frac{2\pi\xi_x}{X}\sqrt{-1}\right)^{\alpha_l} \beta_l (U_p^q)^{\beta_l - 1} - \frac{2\pi\xi_t}{T}\sqrt{-1} \right) \right|. \tag{5.12}$$

Proof. Following the definition of e_{noise} in (Equation 5.10), we have

$$\begin{aligned} & e_{\text{noise}} \\ &= \max_{\xi_x,\xi_t} \left| \sum_{l\in \text{Supp}^*} c_l \left(\frac{2\pi\xi_x}{X}\sqrt{-1}\right)^{\alpha_l} \sum_{p=0}^{\mathbb{N}_x-1}\sum_{q=0}^{\mathbb{N}_t-1} \left\{ e^{-\left(\frac{2\pi p}{\mathbb{N}_x}\xi_x + \frac{2\pi q}{\mathbb{N}_t}\xi_t\right)\sqrt{-1}} \left(\beta_l \left(U_p^q\right)^{\beta_l-1} \epsilon_p^q + \mathcal{O}((\epsilon_p^q)^2) \right) \right\} \right. \\ &\quad \left. - \left(\frac{2\pi\xi_t}{T}\sqrt{-1}\right) \sum_{p=0}^{\mathbb{N}_x-1}\sum_{q=0}^{\mathbb{N}_t-1} \left\{ e^{-\left(\frac{2\pi p}{\mathbb{N}_x}\xi_x + \frac{2\pi q}{\mathbb{N}_t}\xi_t\right)\sqrt{-1}} \epsilon_p^q \right\} \right| \frac{XT}{\mathbb{N}_x\mathbb{N}_t} \le \frac{XT}{\mathbb{N}_x\mathbb{N}_t} S\epsilon + \mathcal{O}(\epsilon^2), \end{aligned}$$

where S is defined in (Equation 5.12).

□

First, Theorem 5.2.1 shows that the error $\boldsymbol{e}_{\text{noise}}$ scales linearly with respect to the noise

level ϵ. In comparison, numerical differentiation of the features in the physical domain can give error as

$$\mathcal{O}\left(\Delta t + \Delta x^{p+1-r} + \frac{\epsilon}{\Delta t} + \frac{\epsilon}{(\Delta x)^r}\right)$$

where r is the highest order of derivatives for the active features in the true support, and numerical differentiation is carried by interpolating the data by a pth order polynomial [29, 204]. The noise is magnified by $1/\Delta t$ and $1/(\Delta x)^r$, which shows the challenges of dealing with noisy data. Theorem 5.2.1 shows the formulation of Fourier features (Equation 5.11) is more robust against noise in comparison with numerical formulation of differential features. In the weak formulation, such as WeakIdent[3], the error (resulted from noise) of the linear system also scales linearly with respect to noise,

$$e_{\text{noise}}^{\text{weak}} \leq \bar{S}^* |\Omega_h| \epsilon + \mathcal{O}\left(\epsilon^2\right), \tag{5.13}$$
$$\text{with } \bar{S}^* = \max_h \sup_{(x_j,t^k)\in\Omega_h} \left| \sum_{l\in \text{Supp}^*} (-1)^{\alpha_l} c_l \beta_l (U_j^k)^{\beta_l - 1} \frac{\partial^{\alpha_l}\phi}{\partial x^{\alpha_l}}(x_j,t^k) - \frac{\partial \phi}{\partial t}(x_j,t^k) \right|,$$

where h is an index of the test function, Ω_h is the support of the h-th test function, and $|\Omega_h|$ is the area of Ω_h [3, Theorem 1]. The error bounds in both (Equation 5.11) and (Equation 5.13) scale linearly with respect to the noise level ϵ, which demonstrates that the numerical formulations of weak features and Fourier features are both robust to noise.

Secondly, the major difference between (Equation 5.11) and (Equation 5.13) is that, the error in WeakIdent is local depending on the local support Ω_h of the h-th test function, while the error in FourierIdent is global.

5.3 Fourier feature denoising and core regions of features

One of the main difficulties of FourierIdent is that frequency responses may have large magnitude, which is different from the linear system generated in the physical domain. As in [29, 204, 3], even in physical domain denoising is very important for coefficient

identification. In this section, we propose a denoising process for FourierIdent using (i) Fourier feature denoising, and defining (ii) the meaningful data region, and (iii) the core regions of features in the frequency domain. We denoise Fourier features as in the physical domain, and further partition the frequency domain to the core regions of features and the rest to separate noise from data. We give details of denoising in this section before we present the main algorithm of FourierIdent in section 5.4.

5.3.1 Denoising Fourier features

We first denoise Fourier features by applying convolution with a Gaussian-shape kernel $\phi(x,t)$. By the convolution theorem, we conduct all computations about convolution in the frequency domain. This can be applied in a restricted domain. For example, let Λ be a smaller region in the frequency domain, and the matrix $\boldsymbol{F}_\Lambda$ and the vector $\boldsymbol{b}_\Lambda$ be restricted to the entries from the region Λ. We let $(\xi_x,\xi_t) \in \Lambda$, and h be the associated index such that $h = \mathcal{H}(\xi_x,\xi_t)$, which is also the row index of the frequency mode (ξ_x,ξ_t) in $\mathcal{S}(\boldsymbol{F})$. We define the smoothing operator $\mathcal{S}$ on the matrix $\boldsymbol{F}_\Lambda$ as follows:

$$
\begin{aligned}
&[\mathcal{S}(\boldsymbol{F})_\Lambda]_{h,l} \qquad (5.14)\\
&= \left(\frac{2\pi\xi_x\sqrt{-1}}{X}\right)^{\alpha_l} \sum_{p=0}^{\mathbb{N}_x-1}\sum_{q=0}^{\mathbb{N}_t-1} e^{-\left(\frac{2\pi p}{\mathbb{N}_x}\xi_x+\frac{2\pi q}{\mathbb{N}_t}\xi_t\right)\sqrt{-1}} \left(U^q_{noise,p}\right)^{\beta_l} \cdot \sum_{p=0}^{\mathbb{N}_x-1}\sum_{q=0}^{\mathbb{N}_t-1} e^{-\left(\frac{2\pi p}{\mathbb{N}_x}\xi_x+\frac{2\pi q}{\mathbb{N}_t}\xi_t\right)\sqrt{-1}}\Phi^q_p,
\end{aligned}
$$

where $(\xi_x,\xi_t) \in \Lambda$, and h and l are row and column index of the smoothed matrix $\mathcal{S}(\boldsymbol{F})_\Lambda$,

$$
\phi(x,t) = \left(1-\left(\frac{x}{m_x\Delta x}\right)^2\right)^{p_x}\left(1-\left(\frac{t}{m_t\Delta_t}\right)^2\right)^{p_t}
$$

for

$$
(x,t) \in (-m_x\Delta x, m_x\Delta x)\times(-m_t\Delta t, m_t\Delta t)
$$

and Φ^q_p $(p=0,...,\mathbb{N}_x-1, q=0,...,\mathbb{N}_t-1)$ gives a discrete evaluation of $\phi(x,t)$ on the grid point (x_p,t^q). The operation with $\phi(x,t)$ in (Equation 5.14) smoothes the data, because the

point-wise multiplication between Φ and U in the frequency domain is equivalent to the convolution between ϕ and u in the physical domain. This denoising process is motivated by the weak form of feature as in [37, 3], and we apply the same method to determine the nonnegative integers m_x, m_t, p_x, p_t related with the decay of the test function ϕ, from the given data. The idea is to match the shape of the smoothing kernel ϕ to the shape of a Gaussian function, such that the noise region will be mostly in the tail of the Gaussian. Another aspect is the smoothness of ϕ that m_x, m_t, p_x, p_t are chosen to guarantee all the features in the dictionary, including the highest derivative terms, are at least continuous.

5.3.2 The meaningful data region Λ in the frequency domain

We separate the frequency domain into the meaningful data region and the core regions of features for further noise separation. We first utilize a theoretical bound for the decay of Fourier coefficients to define a meaningful data region Λ.

Lemma 5.3.1. *Let $u \in L^2([0,1])$ be continuous and denote its Fourier transform by $\mathcal{F}(u)$ with $\mathcal{F}(u)[\xi] = \int_0^1 u(x)e^{-2\pi\sqrt{-1}\xi x}dx$. If $u^{(p-1)} = \frac{\partial^{p-1}u}{\partial x^{p-1}}$ is continuous and $u^{(p-1)} \in L^2([0,1])$. then the Fourier coefficients $\mathcal{F}(u^{(q)})$ of the q-th ($q \leq p$) order derivative $\frac{\partial^q u}{\partial x^q}$ satisfies*

$$\mathcal{F}(u^{(q)})[\xi] \propto \frac{1}{\xi^{p+1-q}}, \tag{5.15}$$

where $\propto$ denotes the proportional relation.

Proof. It is shown in [205] that the Fourier coefficients of u satisfies $\mathcal{F}(u)[\xi] \propto \frac{1}{\xi^{p+1}}$. Since $\mathcal{F}(u^{(q)})[\xi] = (\sqrt{-1}2\pi\xi)^q\mathcal{F}(u)[\xi]$, we obtain that $\mathcal{F}(u^{(q)})[\xi] \propto \frac{1}{\xi^{p+1-q}}$. □

Following Lemma 5.3.1, we assume that PDE solution satisfies $\mathcal{F}(u)[\xi_x, \xi_t] \propto \frac{1}{\xi_x^{c_1}}$, and $\mathcal{F}(u)[\xi_x, \xi_t] \propto \frac{1}{\xi_t^{c_2}}$ for some $c_1, c_2 > 0$. Motivated by this decay rate, we partition the frequency domain into two regions. One region follows a proper decay rate which gives the meaningful data region Λ for differential equation identification, and the other high frequency region indicates the noise from the given data.

For simplicity, we use $U \in \mathbb{C}^{N_x \times N_t}$ to denote the given data. We can assume the data are periodic in both time and space by the augmentation along the temporal domain in subsection 5.5.1. After augmentation, the spatial and temporal dimensions are updated to $N_x = \mathbb{N}_x$ and $N_t = 2\mathbb{N}_t - 2$.

We take the two-dimensional Fourier Transform of the noisy data and accumulate the responses in the ξ_x and ξ_t dimension (frequency index ξ_x and ξ_t) respectively, and exploit the same decay rate in both dimensions:

$$\sum_{\xi_t=0}^{N_t-1} \big|\mathcal{F}(U)[\xi_x, \xi_t]\big| \propto \frac{1}{\xi_x^{c_1}}, \quad \sum_{\xi_x=0}^{N_x-1} \big|\mathcal{F}(U)[\xi_x, \xi_t]\big| \propto \frac{1}{\xi_t^{c_2}},$$

This can be written in a linear form using the log function:

$$\log\left(\sum_{\xi_t=0}^{N_t-1} \big|\mathcal{F}(U)[\xi_x, \xi_t]\big|\right) \propto \quad -c_1 \log(\xi_x) + d_1. \tag{5.16}$$

We find **a transition frequency mode** a_x separating two regions, by fitting a linear relation for low frequency modes ($\xi_x \leq a_x$), and fitting a flat line for high frequency modes ($\xi_x > a_x$) in the log-log scale:

$$\begin{aligned} a_x^* = \underset{a_x \in \{0,\ldots,\lfloor N_x/2 \rfloor\}}{\arg\min} \Bigg\{ & \left(\min_{a,b \in \mathbb{R}} \sum_{\xi_x \in \{0,1,\ldots,a_x\}} \left(a \log(\xi_x) + b - \log[y(\xi_x)]\right)^2 \right) \\ & + \sum_{\xi_x = a_x+1}^{\lfloor N_x/2 \rfloor} \left(\log[\bar{y}(a_x+1)] - \log[y(\xi_x)]\right)^2 \Bigg\}. \end{aligned} \tag{5.17}$$

with

$$y(\xi_x) = \sum_{\xi_t=0}^{N_t-1} \big|\mathcal{F}(U)[\xi_x, \xi_t]\big|, \quad \bar{y}(a_x+1) = \frac{1}{\lfloor N_x/2 \rfloor - a_x} \sum_{\xi_x=a_x+1}^{\lfloor N_x/2 \rfloor} \sum_{\xi_t=0}^{N_t-1} \big|\mathcal{F}(U)[\xi_x, \xi_t]\big|.$$

The first term in (Equation 5.17) fits the decay rate (Equation 5.16) and the second term fits the flat noise region. This is based on the assumption that low frequency data are dom-

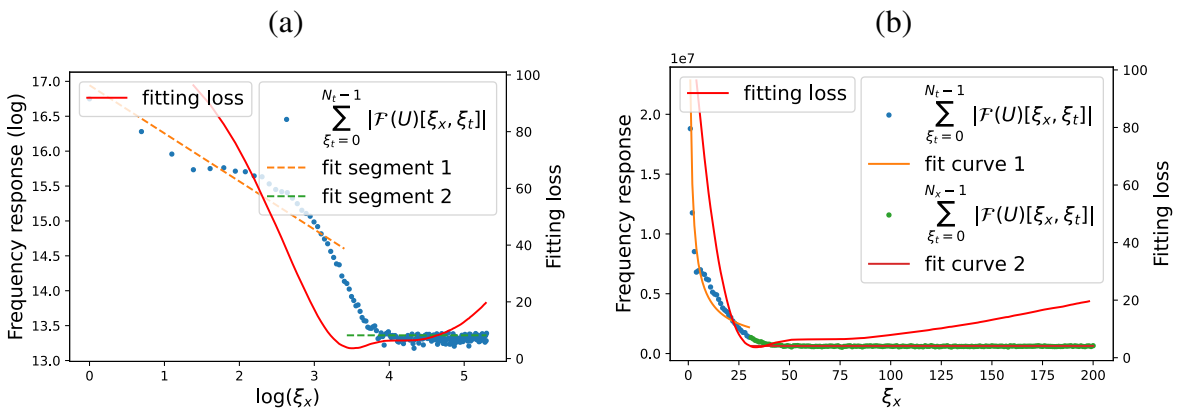


Figure 5.1: Decay fit in the frequency and physical domains. (a) Blue dots are the given data, orange and green dotted lines are the fitting lines to find the transition frequency mode a_x^* (Equation 5.17) in log-log scale. (b) the same plot in physical domain and original unit. Red curves in both graphs show the fitting loss.

inated by the actual dynamics of the differential equation, while the high frequency data are mostly dominated by noise. Thus, a transition frequency mode can be detected by the change of the relation between $\log\left(\sum_{n=0}^{N_t-1}\left|\mathcal{F}((U))[\xi_x,\xi_t]\right|\right)$ and $\log(\xi_x)$. The transition frequency mode in time, denoted by a_t^*, can be obtained similarly.

In Figure 5.1 (a) we show the accumulated frequency responses (blue dots), the linear fitting curve at the low-frequency modes (dash lines), and the total loss function $\mathcal{L}$ of (Equation 5.17) (red lines). The x-axis and left-y-axis provide the scales of points (x_i, y_i) in the linear fit. The right-y-axis provides the scale of the loss function $\mathcal{L}$ in (Equation 5.17). In (b), visualization of the fitted curves is shown in the original scale (instead of $\log$ scale) in the frequency domain. The transition mode is detected as $a_x^* = 27$ from (Equation 5.17).

With the transition frequency modes a_x^* and a_t^* in ξ_x and ξ_t directions respectively, we partition the frequency domain into two regions: the meaningful data region Λ and noise regions, where

$$\Lambda = \{(\xi_x, \xi_t) : \xi_x = 0, 1, ..., a_x^* - 1, \xi_t = 0, 1, ..., a_t^* - 1\}$$

and the complement $\Lambda^{\complement}$ is the noise region. This is motivated from the theoretical decay

rate of the Fourier coefficient of U.

When the given data are noisy, each Fourier feature $\mathcal{F}\{g_l\}$ is affected by noise. The region with a high response of $\mathcal{F}\{g_l\}$ usually contains more signal information than the region with a low response of $\mathcal{F}\{g_l\}$. We further define the **core region of feature** for g_l, denoted as $\mathcal{V}(g_l)$ such that:

$$\begin{aligned}
&\mathcal{V}(g_l) = (\mathcal{V}_{\mathcal{R}}(g_l), \mathcal{V}_{\mathcal{I}}(g_l)) \quad \text{with} \\
&\mathcal{V}_{\mathcal{R}}(g_l) = \left\{ h = \mathcal{H}(\xi_x, \xi_t) : (\xi_x, \xi_t) \in \Lambda,\ |\mathcal{R}\left(\left[\mathcal{S}(\boldsymbol{F})\right]_{h,l} \right)| \geq \beta_{g_l} \right\}, \\
&\mathcal{V}_{\mathcal{I}}(g_l) = \left\{ h = \mathcal{H}(\xi_x, \xi_t) : (\xi_x, \xi_t) \in \Lambda,\ |\mathcal{I}\left(\left[\mathcal{S}(\boldsymbol{F})\right]_{h,l} \right)| \geq \beta_{g_l} \right\}.
\end{aligned} \tag{5.18}$$

Here $\mathcal{V}(g_l)$ contains two ordered sets, in which the first set denotes the high response region for the real part of the Fourier feature for g_l, and the second set denotes the high response region for the imaginary part of the Fourier feature for g_l.

For the feature g_l, we pick the high responses among the frequencies in Λ to be in the core region, and consider low responses as noise and remove them from the system (Equation 5.6). We refer to subsection 5.5.3 for more details on how to choose the threshold β_{g_l}. For the feature u_t, we consider the smoothed frequency response restricted to Λ, $\mathcal{S}(\boldsymbol{b})_\Lambda$, and its magnitude at each frequency. We take the high response region as the core region determined by a threshold β_{u_t}. Specifically, the core region of u_t, $\mathcal{V}(u_t)$, is defined similarly as $\mathcal{V}(g_l)$ in (Equation 5.18), where β_{g_l} is replaced by β_{u_t} and $\mathcal{S}(\boldsymbol{F})_{h,l}$ is replaced by $\mathcal{S}(\boldsymbol{b})_h$.

Using the core region of feature u_t, we construct the linear system as

$$\overline{\mathcal{S}(\boldsymbol{F})_{\mathcal{V}(u_t)}}\boldsymbol{c} = \overline{\mathcal{S}(\boldsymbol{b})_{\mathcal{V}(u_t)}}, \tag{5.19}$$

where

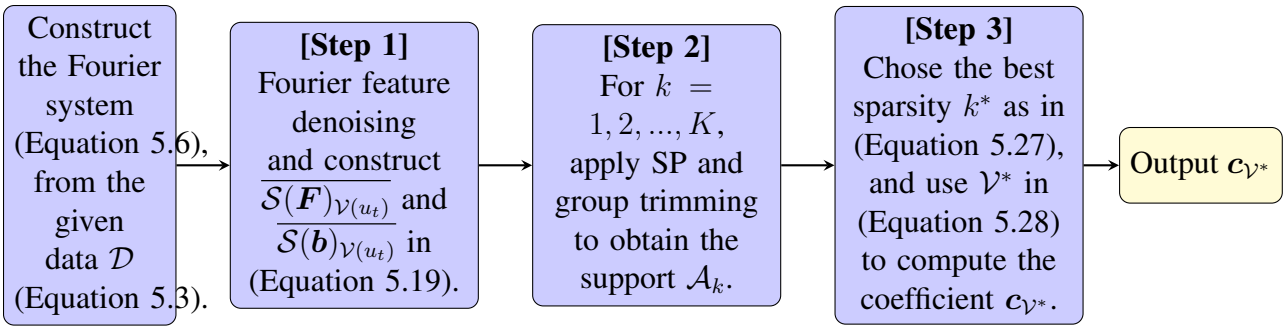


Figure 5.2: The flowchart of FourierIdent. A discrete Fourier system (Equation 5.6) is constructed from the given data. In [Step 1], Fourier features are denoised and a smaller system (Equation 5.19) is constructed. In [Step 2], we apply Subspace Pursuit to obtain initial supports for each sparsity level $k = 1, 2, ..., K$ and new group trimming is applied. In [Step 3], the best coefficient is computed from the energy (Equation 5.27) based on core region.

$$\overline{\mathcal{S}(\boldsymbol{F})_{\mathcal{V}(u_t)}} = \begin{bmatrix} \mathcal{R}\left(\mathcal{S}(\boldsymbol{F})_{\mathcal{V}_{\mathcal{R}}(u_t)}\right) \\ \mathcal{I}\left(\mathcal{S}(\boldsymbol{F})_{\mathcal{V}_{\mathcal{I}}(u_t)}\right) \end{bmatrix} \quad \text{and} \quad \overline{\mathcal{S}(\boldsymbol{b})_{\mathcal{V}(u_t)}} = \begin{bmatrix} \mathcal{R}\left(\mathcal{S}(\boldsymbol{b})_{\mathcal{V}_{\mathcal{R}}(u_t)}\right) \\ \mathcal{I}\left(\mathcal{S}(\boldsymbol{b})_{\mathcal{V}_{\mathcal{I}}(u_t)}\right) \end{bmatrix}.$$

Here we use $\mathcal{S}(\boldsymbol{F})_{\mathcal{V}_{\mathcal{R}}(u_t)}$ to denote the submatrix of $\mathcal{S}(\boldsymbol{F})$ with rows restricted to the ones indexed by $\mathcal{V}_{\mathcal{R}}(u_t)$.

The overline denotes the vertical stacking of real and imaginary responses. This leads to a reduced linear system that is defined only on the core region of the feature u_t. To represent the coefficients identified within the core region of u_t, we use

$$\boldsymbol{c}_{\mathcal{V}(u_t)} = \mathrm{LeastSquare}(\widetilde{\mathcal{S}(\mathrm{F})_{\mathcal{V}(\mathrm{u_t})}}, \widetilde{\mathcal{S}(\mathrm{b})_{\mathcal{V}(\mathrm{u_t})}}). \tag{5.20}$$

Here we use the error-normalized Fourier feature matrix $\widetilde{\mathcal{S}(\boldsymbol{F})_{\mathcal{V}(u_t)}}$ and the error-normalized dynamic variable $\widetilde{\mathcal{S}(\boldsymbol{b})_{\mathcal{V}(u_t)}}$ defined in (Equation 5.30) to solve the least square problem in (Equation 5.20) (details presented in subsection 5.5.2).

5.4 Fourier features for Identifying differential equations (FourierIdent)

From the given data, Fourier features are denoised and a smaller linear system with reduced size is constructed in section 5.3. We utilize Subspace Pursuit [182] for the coefficient support identification, and we propose (i) a new group trimming for stable support recovery, and (ii) a new energy based on the core regions of features for the coefficient identification. There are three steps to the proposed FourierIdent, and Figure 5.2 shows the flowchart to illustrate the process.

1. From the given input data, we construct a denoised linear system with $\overline{\mathcal{S}(\boldsymbol{F})_{\mathcal{V}(u_t)}}$ and $\overline{\mathcal{S}(\boldsymbol{b})_{\mathcal{V}(u_t)}}$ in (Equation 5.19). This is generated by the denoising operation $\mathcal{S}$ in (Equation 5.14), the discrete Fourier system (Equation 5.6) and using the core region of u_t, $\mathcal{V}(u_t)$, to reduce the size and denoise the system.

2. For each sparsity level $k = 1, 2, \ldots, K$, we apply Subspace Pursuit [182] on a column normalized system of $\overline{\mathcal{S}(\boldsymbol{F})_{\mathcal{V}(u_t)}}^{\dagger}$, and $\overline{\mathcal{S}(\boldsymbol{b})_{\mathcal{V}(u_t)}}^{\dagger}$ († denotes the column normalization) and get an initial support $\mathcal{A}_k^0$. We apply a new group trimming to remove features which minimally contribute to the dynamics of the system. This is applied iteratively, until the support converges to $\mathcal{A}_k$ for each k.

3. To choose the optimal coefficient, we propose a new energy based on the core regions of features. First, the optimal sparsity is determined by choosing k^* which gives the minimum residual on the union of the core regions of features identified in $\mathcal{A}_k$. Secondly, we identify the optimal coefficients, by finding the coefficients which gives the minimum residual among least-square fits on the core region of each feature in $\mathcal{A}_k$.

Algorithm 7 FourierIdent Algorithm

Input: Given data (Equation 5.3). ;
[Step 1] Construct the discrete linear system $\overline{\mathcal{S}(\boldsymbol{F})}_{\mathcal{V}(u_t)}, \overline{\mathcal{S}(\boldsymbol{b})}_{\mathcal{V}(u_t)}$ as in (Equation 5.19)
[Step 2]
for k = 1,2,...,K **do**
  Run SP($\overline{\mathcal{S}(\boldsymbol{F})_{\mathcal{V}(u_t)}}^{\dagger}, \overline{\mathcal{S}(\boldsymbol{b})_{\mathcal{V}(u_t)}}^{\dagger}, k$) using SP [182] and set $j = 0$
  Compute the coefficients $\boldsymbol{c}_{\mathcal{V}(u_t)}$ by the Least Square in (Equation 5.20)
  while $\mathcal{A}_k^j$ not convergent **do**
    Perform group trimming in (Equation 5.23) with $\mathcal{T} = 0.08$ and set $j = j + 1$
    Update the contribution score s_l in (Equation 5.22), using the updated coefficients $\boldsymbol{c}_{\mathcal{V}(u_t)}$ in (Equation 5.20)
  end while
end for
[Step 3 - 1] Determine the optimal sparsity k^* according to (Equation 5.27) and the support $\mathcal{A}_{k^*}$.
[Step 3 - 2] Compute $\boldsymbol{c}^*$ by minimizing (Equation 5.29).
Output: $\boldsymbol{c}^* \in \mathbb{R}^L$ such that $\boldsymbol{F}\boldsymbol{c}^* \approx \boldsymbol{b}$.

5.4.1 Subspace Pursuit (SP) and Group trimming

With the denoised Fourier features, we apply Subspace Pursuit (SP) [182] to obtain an initial support A_k^0 with each sparsity level k. Subspace Pursuit is a greedy algorithm where one can input a column normalized matrix Φ, a vector $\boldsymbol{b}$, and a sparsity k. The goal is to find a k-sparse solution of the linear system $\Phi\boldsymbol{x} = \boldsymbol{b}$ as

$$\min_{\boldsymbol{x}:\|\boldsymbol{x}\|_0=k} \|\Phi\boldsymbol{x} - \boldsymbol{b}\|_2^2. \tag{5.21}$$

However, this optimization problem is hard to solve. Instead, $\mathrm{SP}(\Phi, \boldsymbol{b}, k)$ outputs a k-sparse solution in a greedy way (See [182] for details).

For each sparsity level $k = 1, \ldots, K$, we compute $\mathrm{SP}(\overline{\mathcal{S}(\boldsymbol{F})_{\mathcal{V}(u_t)}}^{\dagger}, \overline{\mathcal{S}(\boldsymbol{b})_{\mathcal{V}(u_t)}}^{\dagger}, k)$, and denote the recovered support by A_k^0. To further remove the insignificant features in A_k^0, we trim the features which have minimum contributions to the dynamics, measured by the *contribution score*;

$$s_l = s(g_l) = |c_l| \cdot \|\overline{\mathcal{S}(\boldsymbol{F})_{\mathcal{V}(u_t),l}}\|_2 \ . \tag{5.22}$$

Here $\overline{\mathcal{S}(\boldsymbol{F})_{\mathcal{V}(u_t),l}}$ is the l-th column of $\overline{\mathcal{S}(\boldsymbol{F})}$ with rows restricted to the core region $\mathcal{V}(u_t)$. The value c_l is the l-th entry of the identified coefficient from the least-square fit of $\overline{\mathcal{S}(\boldsymbol{F})_{\mathcal{V}(u_t)}}$ and $\overline{\mathcal{S}(\boldsymbol{b})_{\mathcal{V}(u_t)}}$.

To improve the efficiency of the algorithm, we propose to **trim features as a group**. We remove a set of the least relevant features for a given sparsity level k using a threshold $\mathcal{T}$. In this paper, we fix this threshold to be $\mathcal{T} = 0.008$. The group-trimming is used as follows:

- **Re-order features by contribution scores.** We arrange the features in ascending order, denoted as $1', 2', ..., k'$, based on their respective contribution scores, represented as $s_{1'} \leq s_{2'} \leq ... \leq s_{k'}$.

- **Remove the low contributing terms as a group.** We identify the largest value $k'_{\max}$ satisfying condition

$$\frac{\sum_{j=1'}^{k'_{\max}} s_j}{\sum_{k\in\mathcal{A}_k^0} s_k} < \mathcal{T} \leq \frac{\sum_{j=1'}^{k'_{\max}+1} s_j}{\sum_{k\in\mathcal{A}_k^0} s_k}, \tag{5.23}$$

  and remove features with contribution scores lower than $s_{k'_{\max}}$, i.e., $s_l \leq s_{k'_{\max}}$ from the support set $\mathcal{A}_k^i$ when $k'_{\max} > 0$, where $\mathcal{A}_k^i$ denotes the support at the i^{th} iteration after the ith group trimming. This step reduces the size of the support and the support becomes $\mathcal{A}_k^{i+1}$. Here $\mathcal{T}$ helps to remove a subgroup of features that contributes less than a threshold (which is set to be 8% in this paper) overall compared to all features in the support $\mathcal{A}_k^i$. This $\mathcal{T}$ is a measure of the relative significance of a smaller subgroup of trimmed features compared to the whole set of features.

- **Trimming in each iteration.** This group trimming is applied iteratively within each sparsity level k, and the converged support set is assigned as $\mathcal{A}_k$. Note that (Equation 5.23) changes in each iteration and must be recomputed.

The group trimming is a grouped version of trimming in WeakIdent [3]. The group trimming improves efficiency by removing insignificant as a group and gives stable results for different types of equations with various levels of noise.

5.4.2 Identification of coefficients from the core regions of features

Once we have a collection of support sets $\mathcal{A}_k$'s for $k = 1, \ldots, K$, we choose the optimal sparsity k^* (and the associated support $\mathcal{A}_{k^*}$) which minimizes an energy based on the core regions of features. The energy consists of two terms:

$$\begin{aligned} \text{Energy } = & \text{Fitting residual on the union of core regions about } \mathcal{A}_k \\ & + \text{ Stability of the identified coefficient.} \end{aligned} \tag{5.24}$$

The first term in (Equation 5.24) is the fitting residual at the frequencies on the union of the core regions for the features in $\mathcal{A}_k$:

$$\text{Fitting residual on the union of core regions } := \frac{\|\mathcal{S}(\boldsymbol{F})_{\mathcal{V}(\mathcal{A}_k)}\boldsymbol{c}_{\mathcal{V}(\mathcal{A}_k)} - \mathcal{S}(\boldsymbol{b})_{\mathcal{V}(\mathcal{A}_k)}\|_2}{\|\mathcal{S}(\boldsymbol{b})_{\mathcal{V}(\mathcal{A}_k)}\|_2}. \tag{5.25}$$

Here $\boldsymbol{c}_{\mathcal{V}(\mathcal{A}_k)}$ denotes the coefficients recovered using an error-normalized feature matrix $\widetilde{\mathcal{S}(\boldsymbol{F})_{\mathcal{V}(u_t)}}$ and dynamic variable $\widetilde{\mathcal{S}(\boldsymbol{b})_{\mathcal{V}(u_t)}}$ defined in (Equation 5.30) in subsection 5.5.2.The union of core regions for the features in $\mathcal{A}_k$ is

$$\mathcal{V}(\mathcal{A}_k) = \bigcup_{l \in \mathcal{A}_k} \mathcal{V}(g_l).$$

While each feature gives different core regions (high response regions), by using the union $\mathcal{V}(\mathcal{A}_k)$, we consider fitting on the high response regions for all the features in $\mathcal{A}_k$

The second term in (Equation 5.24) measures the stability of the identified coefficients. For each feature g_l such that $l \in \mathcal{A}_k$, we compute the coefficients $\boldsymbol{c}_{\mathcal{V}(g_l)}$ as in (Equation 5.20) while the core region $\mathcal{V}(u_t)$ is replaced by $\mathcal{V}(g_l)$. We compare the normalized distance among all coefficient vectors computed on the core region of each feature in $\mathcal{A}_k$, and define

the follwing stability term:

$$\text{Stability of the identified coefficients} := \frac{1}{\left|\mathcal{A}_k\right|^2}\frac{1}{\left\|\boldsymbol{c}_{\mathcal{V}(u_t)}\right\|_2}\sum_{l,l'\in\mathcal{A}_k,l\neq l'}\left\|\boldsymbol{c}_{\mathcal{V}(g_l)}-\boldsymbol{c}_{\mathcal{V}(g_{l'})}\right\|_2, \tag{5.26}$$

If fitting on the core region of one feature gives a very different coefficient vector from fitting on the core region of another feature, this stability term in (Equation 5.26) will be larger than that when the coefficients computed on the core regions of different features are similar. This term measures the stability of coefficient computation on the core regions of different features in $\mathcal{A}_k$.

We find the optimal sparsity k^* and the associated support set $\mathcal{A}_{k^*}$ by minimizing the energy:

$$k^* = \arg\min_k \left\{ \frac{\|\mathcal{S}(\boldsymbol{F})_{\mathcal{V}(\mathcal{A}_k)}\boldsymbol{c}_{\mathcal{V}(\mathcal{A}_k)} - \mathcal{S}(\boldsymbol{b})_{\mathcal{V}(\mathcal{A}_k)}\|_2}{\|\mathcal{S}(\boldsymbol{b})_{\mathcal{V}(\mathcal{A}_k)}\|_2} + \frac{1}{\left|\mathcal{A}_k\right|^2}\frac{1}{\left\|\boldsymbol{c}_{\mathcal{V}(u_t)}\right\|_2}\sum_{l,l'\in\mathcal{A}_k,l\neq l'}\left\|\boldsymbol{c}_{\mathcal{V}(g_l)}-\boldsymbol{c}_{\mathcal{V}(g_{l'})}\right\|_2 \right\}. \tag{5.27}$$

The energy summing (Equation 5.25) and (Equation 5.26) measures how meaningful the recovered support $\mathcal{A}_k$ is in terms of the best fitting on the union of core regions and the stability of coefficient identification.

Finally, we compute the best coefficient vector $\boldsymbol{c}^*$ by solving a least square problem on a properly selected core region, that gives the smallest residual among the features in $\mathcal{A}_{k^*}$ and u_t. For each core region of each features in $\mathcal{A}_{k^*}$ and u_t, residual error is computed. Let $\mathcal{V}^*$ be the best core region of features among all features in $\mathcal{A}_{k^*}$ and u_t, which gives the minimum residual:

$$\mathcal{V}^* = \underset{V\in\left\{\mathcal{V}(u_t),\mathcal{V}(g_l):l\in\mathcal{A}_{k^*}\right\}}{\arg\min} \frac{\|\mathcal{S}(\boldsymbol{F})_V\boldsymbol{c}_V - \mathcal{S}(\boldsymbol{b})_V\|_2}{\|\mathcal{S}(\boldsymbol{b})_V\|_2}. \tag{5.28}$$

The core regions for each feature in $\mathcal{A}_{k^*}$ and u_t are used to find the minimum residual in (Equation 5.28) such that $V \in \left\{\mathcal{V}(u_t), \mathcal{V}(g_l) : l \in \mathcal{A}_{k^*}\right\}$. In our experiments, using the

union $\mathcal{V}(\mathcal{A}_{k^*})$ does not give good results. When fewer number of features are used for the core region, it gives better coefficient recovery. In (Equation 5.28), we experiment on the core regions for each feature, and pick one feature to define the core region.

With $\mathcal{V}^*$, *the identified coefficient vector* $\boldsymbol{c}^*$ is given by

$$\boldsymbol{c}^* = \mathrm{LeastSquare}(\widetilde{\mathcal{S}(\mathrm{F})_{\mathcal{V}^*}}, \widetilde{\mathcal{S}(\mathrm{b})_{\mathcal{V}^*}}). \tag{5.29}$$

using normalized feature matrix $\widetilde{\mathcal{S}(\boldsymbol{F})_{\mathcal{V}^*}}$ and dynamic variable $\widetilde{\mathcal{S}(\boldsymbol{b})_{\mathcal{V}^*}}$ as defined in (Equation 5.30).

5.4.3 Solving least square with column rescaling

We define the rescaling factor as

$$s(l, \mathcal{V}(u_t)) = \begin{cases} \beta_l \mathrm{Avg}_{h \in \mathcal{V}(u_t)} \overline{|\mathcal{S}(\boldsymbol{F})|}_{h,l=\mathcal{L}(\alpha_l, \beta_l - 1)} & \beta_l > 1 \\ \mathrm{Avg}_{h \in \mathcal{V}(u_t)} \overline{|\mathcal{S}(\boldsymbol{F})|}_{h,l=\mathcal{L}(\alpha_l, \beta_l)} & \beta_l = 1 \end{cases}$$

where $\overline{|\mathcal{S}(\boldsymbol{F})|}$ represents the magnitude of the smoothed feature matrix with a vertical stacking of the real and imaginative features, which is similar to that defined in (Equation 5.19). Here Avg_h denotes the average operation over the h index, and $\mathcal{L}(\alpha_l, \beta_l)$ denotes the column index of a feature with derivative order α_l and polynomial degree β_l. This $s(l, \mathcal{V}(u_t))$ is the column scale of l-th feature in $\overline{\mathcal{S}(\boldsymbol{F})}$. Similarly, the scale constant for $\overline{\mathcal{S}(\boldsymbol{b})_{\mathcal{V}(u_t)}}$ is

$$s(b, \mathcal{V}(u_t)) = \mathrm{Avg}_{h \in \mathcal{V}(u_t)} \overline{|\mathcal{S}(\boldsymbol{b})|}$$

Using these scale constants, an alternative least square problem on some customized core regions $\mathbb{A}$ for $\overline{\mathcal{S}(\boldsymbol{F})_{\mathbb{A}}}\boldsymbol{c} = \overline{\mathcal{S}(\boldsymbol{b})_{\mathbb{A}}}$, such as (Equation 5.19), can be converted to

$$\widetilde{\mathcal{S}(\boldsymbol{F})_{\mathbb{A}}}\widetilde{\boldsymbol{c}} = \widetilde{\mathcal{S}(\boldsymbol{b})_{\mathbb{A}}}, \tag{5.30}$$

where

$$\widetilde{\mathcal{S}(\boldsymbol{F})_{\mathbb{A}}} = \overline{\mathcal{S}(\boldsymbol{F})_{\mathbb{A}}} \cdot \operatorname{diag}\left(\frac{1}{s(1,\mathcal{V}(u_t))},...,\frac{1}{s(L,\mathcal{V}(u_t))}\right),\ \widetilde{\mathcal{S}(\boldsymbol{b})_{\mathbb{A}}} = \overline{\mathcal{S}(\boldsymbol{b})_{\mathbb{A}}} \cdot \frac{1}{s(b,\mathcal{V}(u_t))}.$$

Once we compute the coefficient vector $\widetilde{\boldsymbol{c}}$, it is rescaled to the coefficient $\boldsymbol{c}$ by

$$\boldsymbol{c} = \widetilde{\boldsymbol{c}} \cdot \operatorname{diag}\left(\frac{s(b,\mathcal{V}(u_t))}{s(1,\mathcal{V}(u_t))},...,\frac{s(b,\mathcal{V}(u_t))}{s(L,\mathcal{V}(u_t))}\right). \tag{5.31}$$

This normalization is applied to compute coefficients in determining sparsity level Equation 5.27 and recovering coefficients Equation 5.29.

5.5 Numerical Implementation Details

We present implementation details in this section. First, we propose a method to extend and augment the data when the given data are not periodic. Secondly, we review the details of error-normalization on how it is applied in the frequency domain. Thirdly, we discuss the details of how the threshold is computed for the core regions of features.

5.5.1 Domain extension for different boundary conditions

Applying the Fourier Transform on u requires u to be periodic along both t and x directions. When $u(x,t)$ does not satisfy this periodic condition, we extend it to a periodic function to compute Fourier coefficients.

We introduce two transformation operators $\mathcal{H}_t$ and $\mathcal{G}_t$ defined as follows:

$$\mathcal{G}_t(f_l(u(x,t))) = \begin{cases} f_l(u(x,t)) & 0 \le x \le X, 0 \le t \le T \\ -f_l(u(x,2T-t)) & 0 \le x \le X, T \le t \le 2T \end{cases}$$

and

$$\mathcal{H}_t(u(x,t)) = \begin{cases} u(x,t) & 0 \le x \le X, 0 \le t \le T \\ u(x,2T-t) & 0 \le x \le X, T \le t \le 2T \end{cases},$$

where $u(x,t)$ is a continues function for $x \in [0,X]$ and $t \in [0,T]$. The subscript t denotes that we are extending $u(x,t)$ along t. One can easily check that both extended functions $\mathcal{G}_t(u(x,t))$ and $\mathcal{H}_t(u(x,t))$ are periodic along t. Then the (Equation 5.4) is converted to an alternative form

$$\frac{\partial \mathcal{H}_t(u)}{\partial t}(x,t) = \sum_{l=1}^{L} c_l \frac{\partial^{\alpha_l} \mathcal{G}_t(f_l(u))}{\partial x^{\alpha_l}}, \tag{5.32}$$

where u_t is computed with $\mathcal{H}_t(U^n_{noise,i})$ instead of $U^n_{noise,i}$ and features on the right-hand side are computed using $\mathcal{G}_t(f_l(U^n_{noise,i}))$. For simplification of notation, we present the paper with $U^n_{noise,i}$. However, when the given data do not satisfy the period boundary condition, we consistently use the extended boundary (Equation 5.32) instead of (Equation 5.4).

5.5.2 Error-normalization on the core region of features

We review the method of error-normalization introduced in [3] and illustrate how this is implemented in the frequency domain. The motivation is to unify the effect of noise among different feature terms in the library. Using the noise model of the given data (Equation 5.2), each Fourier feature has the following expression

$$\left(\frac{2\pi\xi_x}{X}\sqrt{-1}\right)^{\alpha} \sum_{p=0}^{\mathbb{N}_x-1} \sum_{q=0}^{\mathbb{N}_t-1} \left\{ e^{-\left(\frac{2\pi p}{\mathbb{N}_x}\xi_x + \frac{2\pi q}{\mathbb{N}_t}\xi_t\right)\sqrt{-1}} \left(U^q_p + \epsilon^q_p\right)^{\beta} \right\},$$

where the leading coefficient of the error ϵ^q_p is

$$\left(\frac{2\pi\xi_x}{X}\sqrt{-1}\right)^{\alpha} \sum_{p=0}^{\mathbb{N}_x-1} \sum_{q=0}^{\mathbb{N}_t-1} \left\{ e^{-\left(\frac{2\pi p}{\mathbb{N}_x}\xi_x + \frac{2\pi q}{\mathbb{N}_t}\xi_t\right)\sqrt{-1}} \beta \left(U^q_p\right)^{\beta-1} \right\}. \tag{5.33}$$

5.5.3 Threshold computation for the core region of features

We present how the threshold β_{u_t} used in subsection 5.3.2 is chosen to determine the core region of feature for u_t. In subsection 5.3.2, we partition the meaningful data region Λ further into the core region (high response region of the feature), and the noise region (low response region of the feature). The partition is based on the threshold β_{u_t}.

1. First, we collect absolute values of the smoothed responses for u_t in Λ in a new set denoted by $\mathcal{B}$. We take the absolute values of both the real and imaginary parts for each index $h = \mathcal{H}(\xi_x, \xi_t)$. Here the frequency index $h = \mathcal{H}(\xi_x, \xi_t)$ is in the meaningful data region Λ such that $\xi_x = 1, \ldots, a_x^*$ and $\xi_t = 1, \ldots, a_t^*$.

2. We partition the range of frequency responses in $\mathcal{B}$ into a fixed number of bins $N_{\mathcal{B}} = 300$. The partition is on an equally spaced grid. We denote the index of each bin by $\theta = 1, ..., N_{\mathcal{B}}$. Let b_θ represent the number of Fourier responses located in the θth bin. In other words, each b_θ counts the number of elements of $\mathcal{B}$ with values in $[b_\theta^{\text{left}}, b_\theta^{\text{right}}]$, where $b_\theta^{\text{left}}, b_\theta^{\text{right}}$ denote the lower and upper bound of the responses in the θth bin.

3. We apply a two-piece linear fit on the cumulative sums of these bins, denoted by $B(k) = \sum_{\theta=1}^{k} b_\theta$ for $k = 1, 2, ..., N_{\mathcal{B}}$. The threshold β_{u_t} is chosen as $b_{\theta^*+1}^{\text{left}}$ where θ^* is determined by the minimizer of the sum of two linear fitting residuals:

$$\beta_{u_t} = b_{\theta^*+1}^{\text{left}}$$

where

$$\theta^* = \arg\min_{\theta} \left\{ \sum_{\theta=2}^{k} \left(\frac{B(k) - B(1)}{k-1}\theta + B(1) \right) + \sum_{\theta=k+1}^{N_{\mathcal{B}}-1} \left(\frac{B(N_{\mathcal{B}}) - B(k+1)}{N_{\mathcal{B}} - k - 1}\theta + B(k+1) \right) \right\}.$$

For each feature g_l, the core region of feature is computed with the threshold β_{g_l}. We

choose the threshold β_{gl} in a similar way to the choice of β_{ut}.

5.6 Numerical experiments

In this section, we present numerical experiments. The noise ϵ_i^n for $i = 0, ..., \mathbb{N}_x - 1$, $n = 0, ..., \mathbb{N}_t - 1$, follows a Gaussian distribution with mean zero and variance σ^2_{Noise}. For the noise-to-signal ratio (NSR), we use the following definition:

$$\sigma_{\text{NSR}} = \frac{\sigma_{\text{Noise}}}{\sqrt{\frac{1}{\mathbb{N}_t\mathbb{N}_x}\sum_{i,n}|U_i^n|^2}},$$

as in [3].

To measure the accuracy of identification, we use four identification errors in Table Table 5.1. They are the Relative coefficient error e_2, Relative residual error e_{res}, True Positive Rate (TPR), and Positive Predictive Value (PPV). Here e_2 measure the accuracy of the coefficient identification compared to the true coefficient; e_{res} shows the relative residual error of fitting the identified differential equation to the given data; TPR provides the percentage of how many true features are identified compared to the total count of the true features; PPV provides the percentage of the true features identified as a result among all the identified features.

Name	Definition		Name	Definition	
e_2	$\frac{\|\boldsymbol{c}-\boldsymbol{c}_{\text{true}}\|_2}{\|\boldsymbol{c}\|_2}$	(5.34)	TPR	$\frac{\|\{l : c_{\text{pred}}(l)\neq 0, c_{\text{true}}(l)\neq 0\}\|}{\|\{l : c_{\text{true}}(l)\neq 0\|}$	(5.35)
e_{res}	$\frac{\|\boldsymbol{F}\boldsymbol{c}-\boldsymbol{b}\|_2}{\|\boldsymbol{b}\|_2}$	(5.36)	PPV	$\frac{\|\{l : c_{\text{pred}}(l)\neq 0, c_{\text{true}}(l)\neq 0\}\|}{\|\{l : c_{\text{pred}}(l)\neq 0\|}$	(5.37)

Table 5.1: We use four errors to measure the accuracy of identifying a partial differential equation in this paper: Relative coefficient error (e_2), Relative residual error(e_{res}), True Positive Rate (TPR), and Positive Predictive Value (PPV).

We experiment on various differential equations which are listed in Table 5.2. The Heat (Equation 5.38), Transport equation (Equation 5.39), and Burgers' equation (Equation 5.40) are simulated using the spectrum method. The KdV (Equation 5.41) and KS (Equation 5.42) are simulated using the ETD RK4 method [206]. The true features in these equations contains $u_x, (u^2)_x, u_{xx}, u_{xxx}$, and u_{xxxx}. A diffusion term is added to the transport and burger equation to stabilize the pattern and increase the complexity of the model. These experiments show the robustness of FourierIdent in identifying linear, nonlinear, or higher-order derivative features.

Name	Definition		Simulation paramters
Heat Equation	$\frac{\partial u}{\partial t} = 0.1\frac{\partial^2 u}{\partial x^2}$	(5.38)	$T = [0, 0.0999]$, $[X_1, X_2] = [0, 10]$, $\Delta x = 0.0391$, $\Delta t = 0.003$.
Transport Equation with diffusion	$\frac{\partial u}{\partial t} = -\frac{\partial u}{\partial x} + 0.1\frac{\partial^2 u}{\partial x^2}$	(5.39)	$T = [0, 0.0999]$, $[X_1, X_2] = [0, 10]$, $\Delta x = 0.0391$, $\Delta t = 0.003$.
Burger's Equation with diffusion	$\frac{\partial u}{\partial t} = 0.25\frac{\partial}{\partial x}u^2 + 0.05\frac{\partial^2 u}{\partial x^2}$	(5.40)	$T = [0, 0.4995]$, $[X_1, X_2] = [-3.1416, 3.1293]$, $\Delta x = 0.0123$, $\Delta t = 0.001$.
Korteweg-de Vires (KdV) equation	$\frac{\partial u}{\partial t} = -0.5\frac{\partial}{\partial x}u^2 - \frac{\partial^3 u}{\partial x^3}$	(5.41)	$T = [0, 0.0200]$, $[X_1, X_2] = [-3.1416, 3.1416]$, $\Delta x = 0.0157$, $\Delta t = 4 \times 10^{-5}$.
Kuramoto-Sivashinsky (KS)	$\frac{\partial u}{\partial t} = -0.5\frac{\partial}{\partial x}u^2 - \frac{\partial^2 u}{\partial x^2} - \frac{\partial^4 u}{\partial x^4}$	(5.42)	$T = [0, 150]$, $[X_1, X_2] = [0, 100.53]$, $\Delta x = 0.3927$, $\Delta t = 0.5$.

Table 5.2: A list of equations used for experiments of FourierIdent. For all the equations, we use the maximum power $\beta = 6$, maximum order of derivative $\alpha = 6$, and total number of features $L = 43$ as the dictionary's parameters.

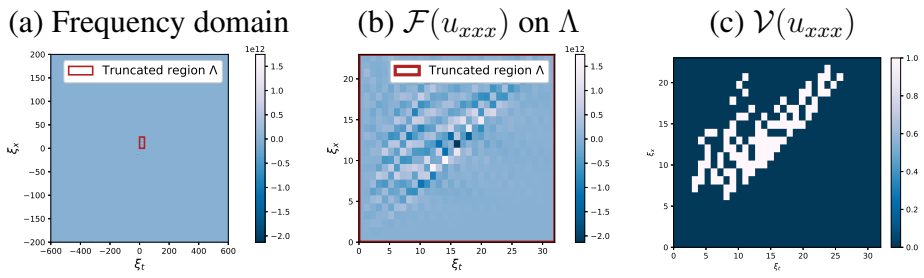


Figure 5.3: The KdV equation (Equation 5.41) with $\sigma_{NSR} = 0.3$ as an example. (a) The frequency domain and Λ (the red box). (b) Zoom of Λ and the frequency response $\mathcal{R}(|\mathcal{F}(u_{xxx})|)$. (c) The white region represents the core region of u_{xxx} further reduced from Λ. There is a big reduction in the size of the region.

5.6.1 Workflow of FourierIdent

We illustrate the procedure of FourierIdent step by step, from constructing a feature matrix to achieving the identified coefficients.

In [Step 1], we obtain the feature matrix (Equation 5.19) and determine the core regions of features. Figure 5.3 shows an example of the KdV equation (Equation 5.41) with $\sigma_{\text{NSR}} = 0.3$ noise. (a) shows the meaningful data region Λ (red box) in relation to the entire frequency domain, (b) shows the frequency response of u_{xxx} in Λ (zoomed), and (c) shows the core region of u_{xxx} restricted in Λ. Note the significant reduction of the size, which contributes to the efficiency of the method in (Equation 5.20).

In [Step 2], for each sparsity level $k = 1, \ldots, K$, we use Subspace Pursuit to obtain an initial candidate support $\mathcal{A}_k^0$ which may be further reduced to a smaller support through group trimming. Table 5.3 shows the first result of SP, $\mathcal{A}_k^0$, and how group trimming is iteratively applied to get $\mathcal{A}_k$ for the KS equation (Equation 5.42). We present the cases for $k \leq 5$, but we computed $\mathcal{A}_k$ until $k = 10$. When $k = 1$, the candidate support has one feature $(u^2)_{xxxxx}$, and the associated energy (Equation 5.27) is 0.8071. When $k = 3$, the correct support is found as the initial support $\mathcal{A}_3^0$, and no more feature is trimmed such that $\mathcal{A}_3 = \mathcal{A}_3^0$. When $k = 4$, the feature $(u^4)_{xxxxx}$ is removed during group trimming. When $k = 5$, the feature $(u^4)_{xxxxx}, (u^2)_{xxx}$ are removed during group trimming. The same support was obtained after group trimming when $3 < k \leq 10$. In the end, optimal k^* is

given from 3, and the associated support is chosen to be $\mathcal{A}^* = \{u_{xx}, u_{xxxx}, (u^2)_x\}$.

Sparsity	Support	Optimal k^* energy (Equation 5.27)
SP(1)	$\mathcal{A}_1^0 = \{(u^2)_{xxxxx}\} = \mathcal{A}_1$	0.8071
SP(2)	$\mathcal{A}_2^0 = \{(u^2)_{xxxxx}, (u^2)_{xxx}\} = \mathcal{A}_2$	4.1957
SP(3)	$\mathcal{A}_3^0 = \{u_{xx}, u_{xxxxx}, (u^2)_x\} = \mathcal{A}_3$	**0.1975**
SP(4)	$\mathcal{A}_4^0 = \{u_{xx}, u_{xxxxx}, (u^2)_x, (u^4)_{xxxx}\}$	
Score (Equation 5.22)	$s(u_{xx}) = 1, s(u_{xxxx}) = 0.59, s((u^2)_x) = 0.63,$ ~~$s((u^4)_{xxxxx}) = 0.01$~~	
Updated	$\mathcal{A}_4^1 = \{u_{xx}, u_{xxxxx}, (u^2)_x\} = \mathcal{A}_4$	**0.1975**
SP(5)	$\mathcal{A}_5^0 = \{u_{xx}, u_{xxxxx}, (u^2)_x(u^2)_{xxx}, (u^4)_{xxxxx}\}$	
Score (Equation 5.22)	$s(u_{xx}) = 1, s(u_{xxxx}) = 0.59, s((u^2)_x) = 0.65,$ ~~$s((u^4)_{xxxxx}) = 0.04, s((u^2)_x) = 0.02$~~	
Updated	$\mathcal{A}_5^1 = \{u_{xx}, u_{xxxx}, (u^2)_x\} = \mathcal{A}_5$	**0.1975**

Table 5.3: The KS equation in (Equation 5.42) with $\sigma_{\text{NSR}} = 0.5$ as an example. [Step 2] SP and group trimming. For each sparsity level k, SP(k) to represent Subspace Pursuit being applied on $\mathcal{S}(\boldsymbol{F})_{\mathcal{V}(u_t)}{}^{\dagger}$ with sparsity level k, we present the initial support and the converged support $\mathcal{A}_k$ after the group trimming. While no features are trimmed when $k \leq 3$, for $k > 3$ with the group trimming, it converged in one step. Note the same features $\mathcal{A}^*$ are found from $k = 3$ with the minimum energy (Equation 5.27).

In [Step 3], we pick the coefficient vector with the minimum energy based on the core region of features (Equation 5.27). Note that these coefficient energies are associated with each individual core region, and Table 5.4 presents this computation from the features of $\mathcal{A}^*$ and u_t, i.e., $\mathcal{V}(u_t), \mathcal{V}(u_{xx}), \mathcal{V}(u_{xxxx}), \mathcal{V}((u^2)_x)$, for the KS equation (Equation 5.42). The coefficient values are similar to each other, thanks to the consistency term (Equation 5.26) in (Equation 5.27), and we choose the best coefficient vector. In this example, $\mathcal{V}^* = \mathcal{V}(u_{xxxx})$ gives the lowest coefficient energy. The final output of this identification example is $u_t = -0.9701u_{xx} - 0.9916u_{xxxx} - 0.4785(u^2)_x$ and the true equation is $u_t = -u_{xx} - u_{xxxx} - 0.5(u^2)_x$ in (Equation 5.42).

Features in $\mathcal{A}_k^*$	Coefficients	Residual error (Equation 5.28)
Use $\mathcal{S}(\boldsymbol{F})_{\mathcal{V}(u_t)}$	$\boldsymbol{c}_{\mathcal{V}(u_t)} = [-0.8834, -0.881, -0.4443]$	0.2817
Use $\mathcal{S}(\boldsymbol{F})_{\mathcal{V}(u_{xx})}$	$\boldsymbol{c}_{\mathcal{V}(u_{xx})} = [-0.8912, -0.8921, -0.4464]$	0.2817
Use $\mathcal{S}(\boldsymbol{F})_{\mathcal{V}(u_{xxxx})}$	$\boldsymbol{c}_{\mathcal{V}(u_{xxxx})} = [-0.9701, -0.9916, -0.4785]$	**0.1468**
Use $\mathcal{S}(\boldsymbol{F})_{\mathcal{V}((u^2)_x)}$	$\boldsymbol{c}_{\mathcal{V}((u^2)_x)} = [-0.9642, -0.9706, -0.4867]$	0.1483
Identified equation : $u_t = -0.9701u_{xx} - 0.9916u_{xxxx} - 0.4785(u^2)_x$, using $\mathcal{V}^* = \mathcal{V}(u_{xxxx})$,		

Table 5.4: The KS equation in (Equation 5.42) with $\sigma_{\text{NSR}} = 0.5$. [Step 3] for each features of $\mathcal{A}^* \cup \{u_t\} = \{u_t, u_{xx}, u_{xxxx}, (u^2)_x\}$ computes the coefficients and find the minimum residual energy (Equation 5.28) feature. The minimum coefficient energy is given by $\mathcal{V}^* = \mathcal{V}(u_{xxxx})$ and the corresponding equation is identified.

5.6.2 Effect of the meaningful data region

We present the effect of the meaningful data region, using the KdV equation with $\sigma_{\text{NSR}} = 0.3$ as an example. Figure 5.4 shows the scale of features of $\boldsymbol{F}$, $\boldsymbol{F}_\Lambda$ ($\boldsymbol{F}$ restricted on Λ), $\mathcal{S}(\boldsymbol{F})$, $\mathcal{S}(\boldsymbol{F})_\Lambda$ and $\boldsymbol{W}$ which is the Weak-form feature in the physical domain as in WeakIdent [3]. The magnitude of the Fourier features have a wider range, and using the meaningful data region Λ helps to reduce this range by comparing $\boldsymbol{F}$ with $\boldsymbol{F}_\Lambda$ and $\mathcal{S}(\boldsymbol{F})$ with $\mathcal{S}(\boldsymbol{F})_\Lambda$. With Λ restriction, the shape of the graph looks similar to the physical values of Weak form. In addition, due to the symmetry, a quarter of data is used, i.e., the frequencies ξ_x, ξ_t with the same magnitude have similar behaviors. We take a smaller collection of the frequency modes which reduces the computational cost.

In Table 5.5, we compare the identification results with or without the restriction to the meaningful data region Λ. For the KdV equation (Equation 5.41) and the KS equation (Equation 5.42) on clean and noise data $\sigma_{\text{NSR}} = 0.3$, we show how the restriction to Λ helps to find the correct equation. Without noise, the restriction to Λ makes little differences. When the noise level increases, the results without restriction to Λ may give completely wrong results.

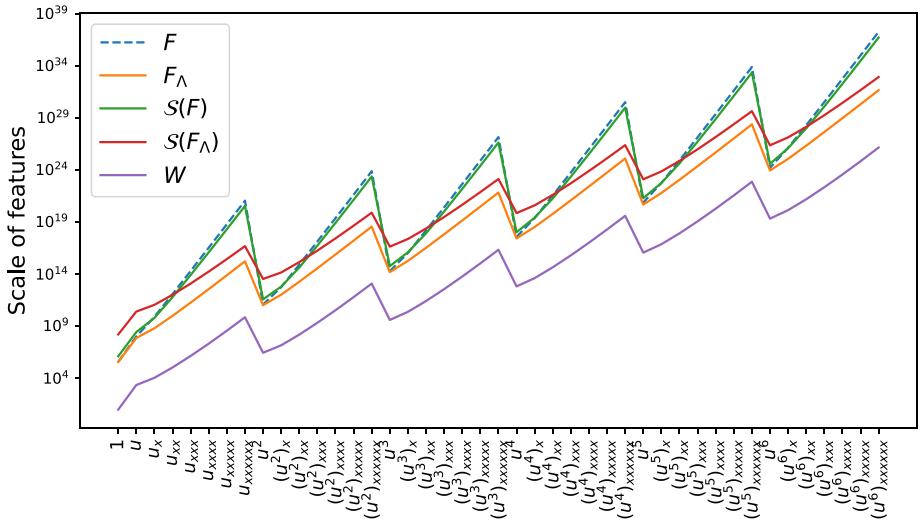


Figure 5.4: Effect of the meaningful data region Λ, illustrated by the KdV equation (Equation 5.41) with $\sigma_{\text{NSR}} = 0.3$. The x-axis provides a list of features and the y-axis represents the scale of features in terms of the ℓ_2-norm of the feature column. We compare $\boldsymbol{F}$ with $\boldsymbol{F}_\Lambda$, $\mathcal{S}(\boldsymbol{F})$ with $\mathcal{S}(\boldsymbol{F})_\Lambda$, and Fourier features with the Weak-form features in $\boldsymbol{W}$ in WeakIdent [3]. A restriction to the meaningful data region Λ helps to reduce the range of Fourier features and make the shape of the scale similar to that in the weak form.

5.6.3 Understanding the new energy

We introduce a new energy based on the core regions of features in subsection 5.4.2. In this subsection, we show the effect of the energy in (Equation 5.27) compared to the Cross-Validation (CV) error used in [204, 3]. In Figure 5.5, we present the result for the KS equation (Equation 5.42) with $\sigma_{\text{NSR}} = 0.8$. In Figure 5.5 (a), the yellow curve shows the new energy (Equation 5.27) value and the blue curve shows the CV error. The x-axis shows the initial sparsity as the input of SP, and the sparsity after group trimming is usually much smaller than the initial sparsity. The yellow and blue circles indicate the cases when the correct supports are found after group trimming. After the sparsity level $k = 3$, the CV errors are all similarly low. If we use the CV error, the wrong supports are identified when $k = 10, 13, 14, 15$. The yellow curve using the new energy is more consistent: the sparsity associated with a low energy corresponds to the correct support. Figure 5.5 (b) shows the values of the two terms in the energy (Equation 5.27): the green curve shows the fitting

KdV (Equation 5.41)	FourierIdent	$u_t = -1.0u_{xxx} - 0.5(u^2)_x$
$\sigma_{\text{NSR}} = 0$	with Λ	$u_t = -0.998u_{xxx} - 0.499(u^2)_x$
	without Λ	$u_t = -0.998u_{xxx} - 0.499(u^2)_x$
$\sigma_{\text{NSR}} = 0.3$	with Λ	$u_t = -1.012u_{xxx} - 0.499(u^2)_x$
	without Λ	$u_t = -151.987u_x$
KS (Equation 5.42)	FourierIdent	$u_t = -1.0u_{xx} - 1.0u_{xxxx} - 0.5(u^2)_x$
$\sigma_{\text{NSR}} = 0$	with Λ	$u_t = -1.0u_{xx} - 1.0u_{xxxx} - 0.5(u^2)_x$
	without Λ	$u_t = -0.999u_{xx} - 0.999u_{xxxx} - 0.5(u^2)_x$
$\sigma_{\text{NSR}} = 0.3$	with Λ	$u_t = -0.978u_{xx} - 0.977u_{xxxx} - 0.49(u^2)_x$
	without Λ	$u_t = -0.035(u^4)_x$

Table 5.5: Effect of applying FourierIdent with or without restriction to the meaningful data region Λ. With an increased level of noise, the benefit of having Λ is clearly demonstrated.

residual, and the purple curve shows the stability term. The yellow curve shows the sum of them, which is the same as the yellow curve in Figure 5.5 (a). While the residual curve (green curve) is minimized at the wrong sparsity $k = 1$, the total energy as the sum of the fitting residual and the stability term is minimized at the correct sparsity and support.

5.6.4 Increasing complexity

We experiment FourierIdent with an increasing complexity of the initial conditions. We use the KdV equation in (Equation 5.41) with different initial conditions (IC) where we can control the complexity:

$$u_0 = \sum_{r=1}^{R} \cos(rx + c_1) + \sin(rx + c_2), \tag{5.43}$$

where c_1 and c_2 are two random numbers and R denotes the total number of modes used in u_0. The larger the R is, the more complex the given data are in both physical and frequency domains. We use periodic spatial domain such that $x \in [-3.1416, 3.1293]$ with spatial spacing $\Delta x = 0.123$ and the time domain $t \in [0, 0.02]$ with temporal spacing $\Delta t = 3.9978 \times 10^{-5}$. In Figure 5.6 (a) - (e), we show clean data for the initial conditions with $R = 1$, 20, 30, 40, and 50 modes. It is shown that the pattern in (a) is relatively simple,

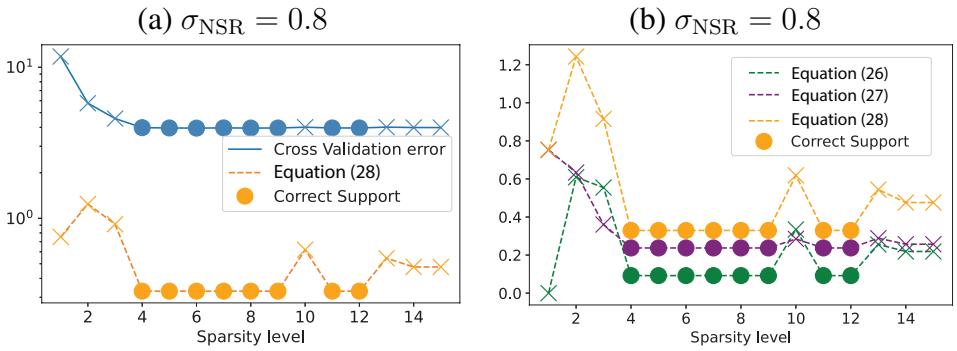


Figure 5.5: Benefits of using the energy in (Equation 5.27) to find the optimal k^*. (a) and (b) are both for KS equation with $\sigma_{\text{NSR}} = 0.8$. The x-axis represents the initial sparsity level. The dots represent the cases when the trimmed support is the correct one. In (a), the y-axis represents the values of Cross-Validation error and the energy in (Equation 5.27). When the given data are noisy, as in (a), the results with sparsity levels $10, 13, 14, 15$ give rise to small cross-validation errors but large energy defined in (Equation 5.27). In (b), the green curve shows the fitting residual in (Equation 5.25), and the purple curve shows the stability term in (Equation 5.26). The new energy as the sum of the fitting residual and the stability term is represented by the yellow curve.

and pattern in (e) is the most complex one. We use a simulated solution from each complexity mode from $R = 1$ to $R = 60$. Figure 5.6 (f)-(i) show the performance of FourierIdent compared with WeakIdent, for different level of noise such that $\sigma_{\text{NSR}} = 0.2, 0.25, 0.3, 0.35$. For each mode R ranging from 1 to 60, we simulate 20 different datasets with different random seeds. In each graph, each column contains 20 dots for the 20 independent experiments, and the height represents the e_2 identification error of Fourierident (blue) and WeakIdent (yellow).

This experiment shows that, for data with different complexity and with different noise levels, FourierIdent gives rise to smaller e_2 errors. Figures (f)-(i) shows that when $K > 10$, FourierIdent (marked in blue) performs better than WeakIdent (marked in yellow), when σ_{NSR} is large and the data have more frequency modes.

5.6.5 Increasing time for data collection

The next experiment shows how the increasing of time interval for data collection can improve the identification result. Figure 5.7 (a) and (b) show a realization of the given data

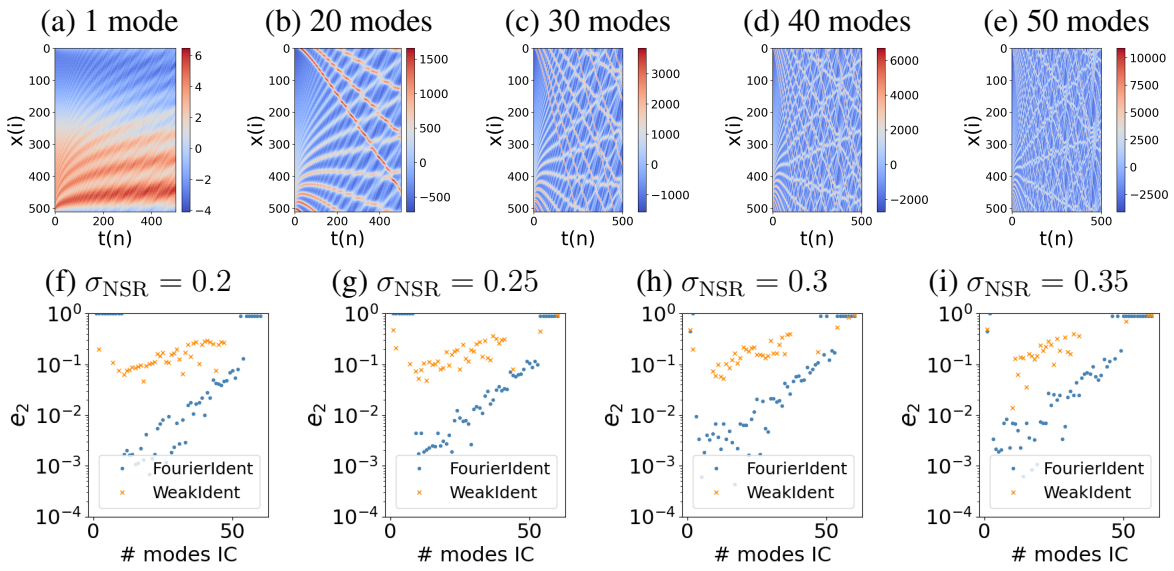


Figure 5.6: Influence of increasing complexity. (a) - (e) Clean data of the KdV equation (Equation 5.41) for the initial condition (Equation 5.43) with $R = 1$, 20, 30, 40, and 50. From (a) to (e), the patterns become more complex. (f) - (i) show comparison results between FourierIdeant and WeakIdent for different noise levels such that $\sigma_{\text{NSR}} \in \{0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35\}$. Each dot represents the e_2 identification error in each experiment. When $10 < R < 40$, FourierIdent gives rise to more accurate recovery results.

for the KS equation (Equation 5.42) with $\sigma_{\text{NSR}} = 0.5$. (a) is for $0 < t < 500$ and (b) is for $0 < t < 5,000$. The data in (b) clearly have more repetition of the pattern. Figure 5.7 (c)-(e) show experiments with varying noise levels such that $\sigma_{\text{NSR}} = 0.01, 0.5, 1.0$. For each graph, the x-axis represents the final time t_{end} used for the data, and y-axis shows the e_2 error of the identified coefficient. In (c), when the noise level is low, both FourierIdent and WeakIdent show good results with the e_2 error less than 10^{-3}. This is based on one experiment for each noise level. As the noise level increases, FourierIdent consistently gives better results in (d) and (e). (e) shows an extremely noisy case for $\sigma_{\text{NSR}} = 1$ when the given data are highly corrupted by noise. For both (d) and (e), as t_{end} gets bigger, both FourierIdent and WeakIdent give rise to decreasing errors , and the error of FourierIdent is lower. The core region is defined by the high frequency responses, thus with more data, the response will be stronger and the core region of feature can separate the noise better. FourierIdent is slightly worse than WeakIdent when $\sigma_{\text{SNR}} = 0.01$ compared to $\sigma_{\text{SNR}} = 0.5$ or 1.0. This shows the effect that FourierIent performs well due to having enough high-frequency responses when the data are exposed to large noise. In other words, FourierIdent is relatively more effective and robust under noise compared to methods using physical features, which is consistent with our observation in Figure 5.6.

5.6.6 FourierIdent comparison results

We test FourierIdent, and compare it with WeakIdent[3] and WSINDy [37]. We first present an example showing the identified equations, and then present more statistics about the recovery. In Table 5.6, we present the identified equation and the e_2 error for the equations listed in Table 5.2 with $\sigma_{\text{NSR}} = 0.3$. FourierIdent identifies the correct equation, i.e., the correct support and highly accurate coefficient values, under various scenarios.

We further show the recovery statistics for the equations in Table 5.2, in Figure 5.8, Figure 5.10, Figure 5.9, Figure 5.11, and Figure 5.12. We present the results with different noise-to-signal-ratio σ_{NSR} using box plots of the identification error for FourierIdent,

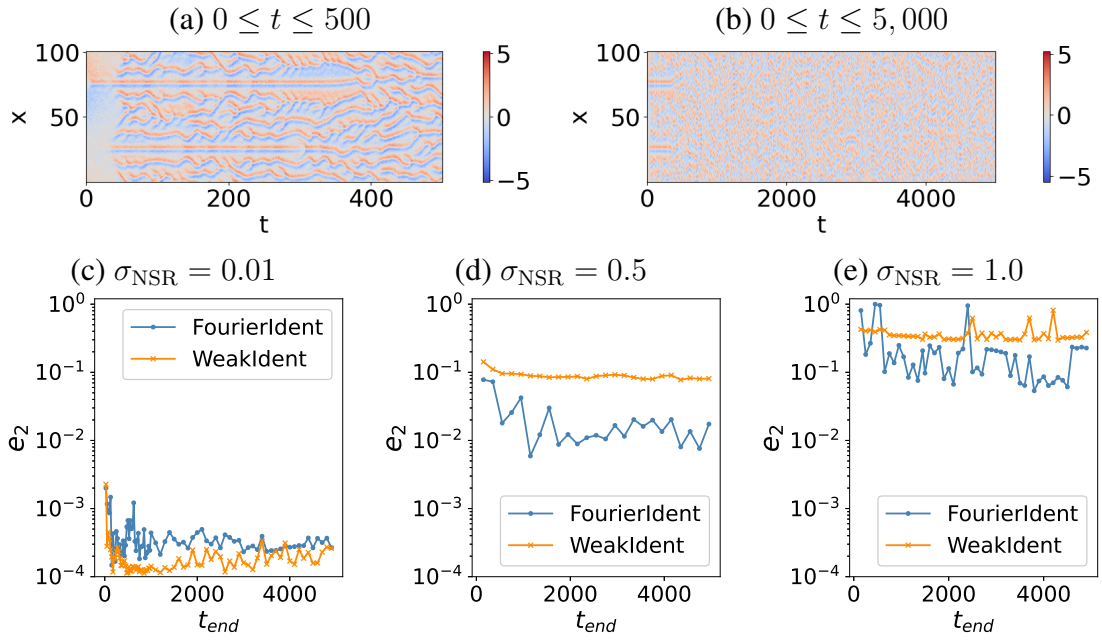


Figure 5.7: Influence of increasing time in data collection. (a) - (b): The given data of the KS equation (Equation 5.42) (a) for $0 < t < 500$, and (b) for $0 < t < 5,000$. (c)-(e): the x-axis represents the final time t_{end} for data collection, and the y-axis shows the e_2 error of the identified coefficient, for FourierIdent (blue) and WeakIdent (yellow). In (c), for low levels of noise $\sigma_{\text{NSR}} = 0.01$, both FourierIdent (blue) and WeakIdent (yellow) give small e_2 errors. In (d) and (e), when $\sigma_{\text{NSR}} = 0.5$ and 1, as t_{end} gets bigger, both FourierIdent and WeakIdent yield smaller errors, while FourierIdent gives rise to a smaller error than WeakIdent.

Equ	Method	Identified equation	e_2
Equation 5.38	FourierIdent	$u_t = +0.1u_{xx}$	0.002536
	WeakIdent	$u_t = +0.1u_{xx}$	0.003609
	WSINDy	$u_t = 0.098u_{xx}$	0.012311
Equation 5.39	FourierIdent	$u_t = -1.0u_x + 0.099u_{xx}$	0.000719
	WeakIdent	$u_t = -0.997u_x + 0.102u_{xx}$	0.003477
	WSINDy	$u_t = 0.434 + -0.793u_{xxxxxx} - 1.003u^2 + 0.096(u^2)_x + 0.470(u^2)_{xxxxxx}$	1.017465
Equation 5.40	FourierIdent	$u_t = +0.051u_{xx} + 0.249(u^2)_x$	0.006846
	WeakIdent	$u_t = +0.052u_{xx} + 0.248(u^2)_x$	0.011226
	WSINDy	$u_t = +0.048u_{xx} + 0.248(u^2)_x$	0.009981
Equation 5.41	FourierIdent	$u_t = -0.997u_{xxx} - 0.499(u^2)_x$	0.0031048
	WeakIdent	$u_t = -0.987u_{xxx} - 0.497(u^2)_x$	0.0121109
	WSINDy	$u_t = -0.977u_{xxx} - 0.4967(u^2)_x$	0.0203434
Equation 5.42	FourierIdent	$u_t = -0.977u_{xx} - 0.97u_{xxxx} - 0.488(u^2)_x$	0.02663459
	WeakIdent	$u_t = -0.95u_{xx} - 0.947u_{xxxx} - 0.476(u^2)_x$	0.051067
	WSINDy	$u_t = -0.9529u_{xx} - 0.9493u_{xxxx} - -0.4779(u^2)_x$	0.048433

Table 5.6: Identification results by FourierIdent, WeakIdent[3] and WSINDy [37] for the equations in Table 5.2 with $\sigma_{\mathrm{NSR}} = 0.3$ using one realization of the PDE solution.

WeakIdent[3] and WSINDy [37]. For each σ_{NSR}, the dataset is simulated 20 times with various random seeds of noise. FourierIdent is compatible with other methods in identifying a partial differential equation. It is robust to high levels of noise and capable of dealing with features with high-order derivatives.

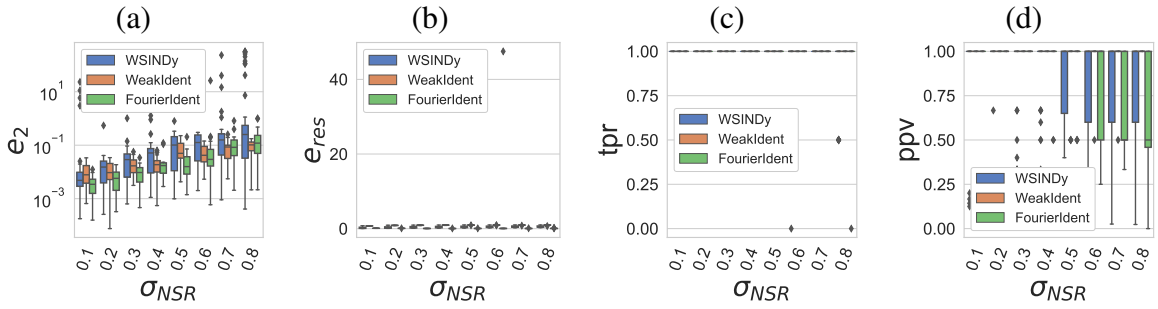


Figure 5.8: Heat equation (Equation 5.38). (a) e_2 error, (b) e_{res}, (c) TPR, and (d) PPV. For each noise level σ_{NSR}, we generate noise using 20 random seeds, and show a box plot for FourierIdent, WeakIdent, and WSINDy.

5.7 Discussion and Conclusion

We proposed FourierIdent, a method to identify differential equations in the frequency domain. We introduced denoising Fourier features by smoothing, and the meaningful data region and the core regions of features, and proposed an energy based on the core regions of features for coefficient identification. Different collections of high frequency responses

h]

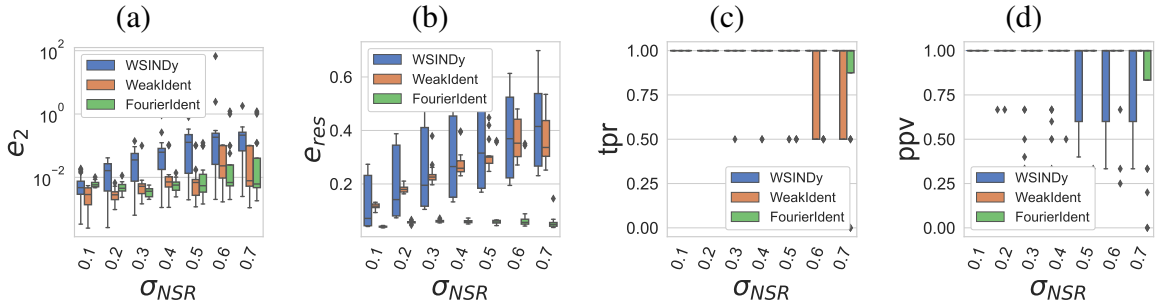


Figure 5.9: Transport equation (Equation 5.39) (a) e_2 error, (b) e_{res}, (c) TPR, and (d) PPV. For each noise level σ_{NSR}, we generate noise using 20 random seeds, and show a box plot for FourierIdent, WeakIdent, and WSINDy.

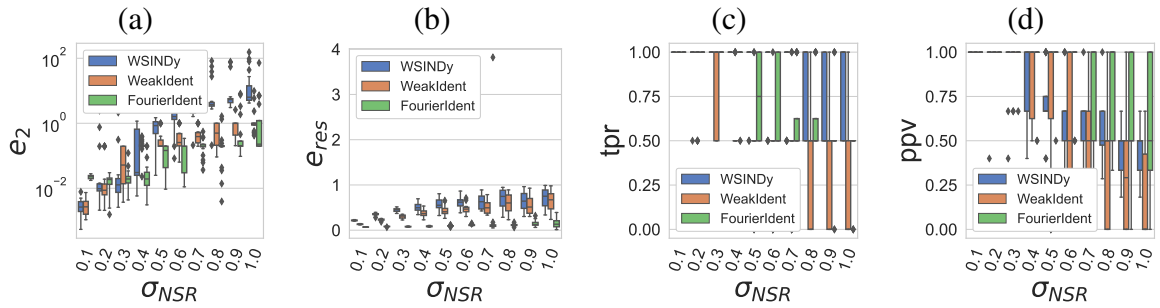


Figure 5.10: The Burgers' equation (Equation 5.40) (a) e_2 error, (b) e_{res}, (c) TPR, and (d) PPV. For each noise level σ_{NSR}, we generate noise using 20 random seeds, and show a box plot for FourierIdent, WeakIdent, and WSINDy.

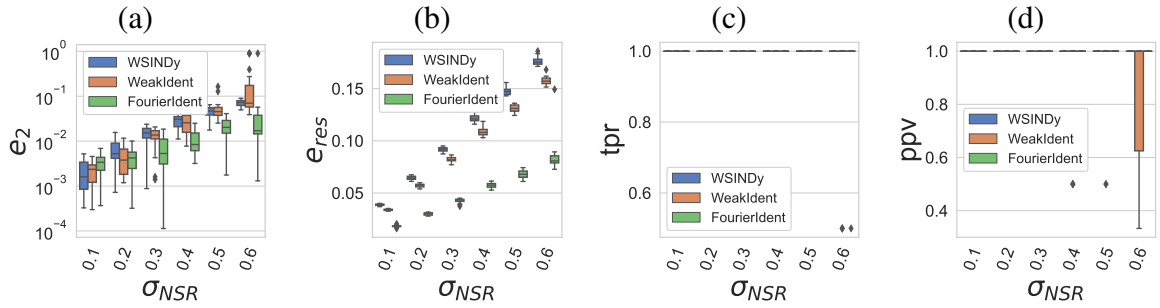


Figure 5.11: The KdV equation (Equation 5.41). (a) e_2 error, (b) e_{res}, (c) TPR, and (d) PPV. For each noise level σ_{NSR}, we generate noise using 20 random seeds, and show a box plot for FourierIdent, WeakIdent, and WSINDy.

are used to identify features and improve coefficient recovery. We present numerical experiments on various simulated datasets to compare FourierIdent with other state-of-art methods using weak formulations. FourierIdent is robust to noise and can handle higher-order derivatives. We show the benefits of FourierIdent on complex datasets simulated using different numbers of Fourier modes.

FourierIdent may be computationally expensive since it iteratively finds different collections of active features and solves the least square problem in many places, but this part can be parallelized. In this paper, we consider features in the form of the derivatives of monomials. Expanding the feature dictionary is a possible direction in our future work.

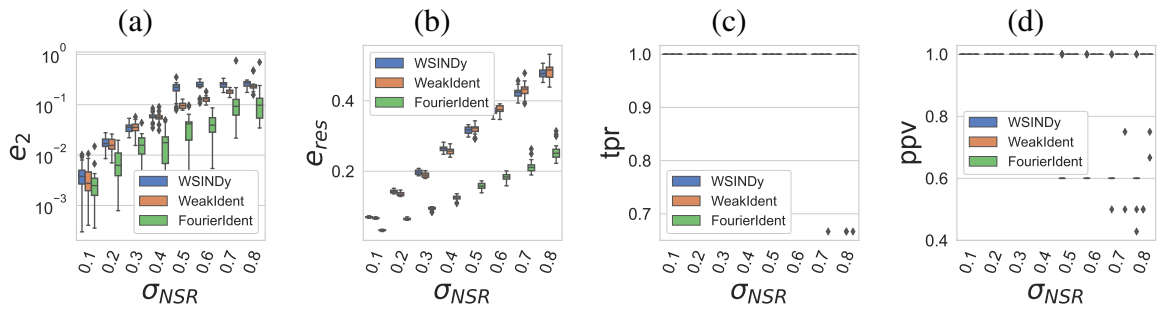


Figure 5.12: Comparison results on the KS equation (Equation 5.42) using multiple identification errors e_2 in (a), e_{res} in (b), TPR in (c), and PPV in (d). For each noise level σ_{NSR}, we generate noise using 20 random seeds and show a box plot for FourierIdent, WeakIdent, and WSINDy.

CHAPTER 6
CONCLUSION AND DISCUSSION

This thesis presents a series of works of identification problems utilizing mathematical models and numerical algorithms. These works explore diverse research topics where folding patterns, objects, and differential equations are identified through innovative methodologies. The underlying intuition behind each model is both simple and novel, supported by comprehensive experiments demonstrating the benefits of each model. These models, built upon mathematical foundations, can be further applied to enhance our understanding and address a wide array of challenges in various domains.

The first proposed method, StemP, is a deterministic Stem-based Prediction algorithm for the RNA secondary structures prediction. By using mainly the Stem Length and Stem-Loop score, we explore the question of what information about the Stem is important for folding prediction. Extensive experiments have been conducted, including different types of sequences: short RNA sequences, tRNA sequences, and 5s rRNA sequences across different methods. One of the strengths of StemP is the simplicity and flexibility of the algorithm, and it gives a deterministic answer. This makes it easier to study folding structure energy more concretely. The significance of incorporating a Stem-Loop score for efficient computation and improved prediction accuracy has been established. Different values and types of Stem-Loop scores for different types of sequences are presented in this work. We investigated and found good values for better predictions. Future directions may involve refining the estimation of the range of such Stem-Loop scores by considering additional sequence properties or incorporating learning-based algorithms and Motif-based strategies to enhance vertex construction. In addition, a sophisticated hierarchical structure clique models as well as chemical reactions can be further considered. Further direction in this area also includes using StemP to predict RNA complex where multiple sequences are given,

and interaction between sequences is taken into account for finding potential stems.

The second presented work, Counting Object by Diffused Index with scalar and multi-dimensional seeds, is a diffusion-based, geometry-free, and learning-free method. The diffusion phase is based on an edge-weighted harmonic optimization model, using the g weight function or mask image and the seed image. An efficient algorithm, called Diffusion Algorithm, is proposed to obtain the diffused image. CODI-S is based on a Gaussian fitted curve to the histogram data of the diffused image such that the number of local maximum of this curve gives the number of objects in the image. For CODI-S, even with a small number of diffusion iterations, there is a large region with 100% accurate counting in the parameter space. CODI-M utilizes more flexible 4-dimensional seeds which can help to distinguish objects better. Typically, a longer iteration compared to CODI-S helps accurate and stable counting for CODI-M. CODI-M can also find each object location in the given image for object identification. This method can further separately count different size objects by clustering the set S of cluster size, as in Figure 3.5. In the numerical section of this chapter, we experimented with the proposed methods on various images, including cells, plants, fruits, and concert crowds. The results confirm that the proposed methods are geometry-free and are able to provide good counts in various cases in a very short amount of CPU time. The proposed method is flexible even if the boundary of the object is not clear nor fully enclosed. We compared with different existing methods, many of which only work for particular types of images considered in their paper. We also compared with methods that require a learning and training process. The proposed methods show comparable results in terms of accuracy. For an image with complex background, object extraction can be difficult. More work can be done in relation to the edge function or the mask to separate the objects and background in the future. In other words, there is more work to be done to provide a better mask such that the diffusion algorithm is able to separate the object from the background effectively in cases when the background is complicated.

The third and fourth works are in the field of identifying differential equations. In the

third work, we propose a new method, WeakIdent for identifying both PDEs and ODE systems from noisy data using a weak formulation in the physical domain. The proposed WeakIdent does not require prior knowledge of the governing features but uses all features up to a certain polynomial order and up to a certain order of derivative. We first use Subspace Pursuit to find a candidate support, then propose two novel techniques called narrow-fit and trimming to improve both the support identification and the coefficient recovery. A careful design of the test functions helps with the recovery, and a proper normalization of the columns in the feature matrix improves the results in the implementation of least-squares. The proposed WeakIdent requires at most L sparsity iterations (or including the sub-iteration of narrow-fit and trimming, at most $\frac{L^2}{2}$ iterations), where L is the number of features. At the same time, the trimming step improves the recovery and gives good results after a fraction of L is used to identify the correct support, as shown in Figure 4.4. Narrow fit based on highly dynamic regions also makes the computation more efficient, and with error normalization of the feature matrix, the coefficient recovery is improved. Comprehensive numerical experiments on various equations/systems are provided, showing the robust performance of WeakIdent compared to other state-of-the-art methods. The Weak form in general, is effective when the noise level is high. The last work, FurierIdent, is a method to identify differential equations in the frequency domain. We introduce a denoised Fourier feature using smoothing and core regions of features and propose an energy based on the core regions of features. Different collections of high-frequency responses are used to identify features and improve coefficient recovery. We present numerical experiments on various simulated datasets to show the identifiability of FourierIdent compared to other state-of-the-art methods using weak formulations. It is shown that FourierIdent is robust to noise and can handle higher-order derivatives. We show the benefit of FourierIdent on complex dynamics throughout datasets simulated using a different number of Fourier modes. FourierIdent may be computationally expensive since it iteratively finds different active collections of modes and solves the least square problems for many possibilities. Hence,

reducing the computation costs can be one of the future directions. FourierIdent considers features that are the derivatives of monomials for straightforward feature computation. Expanding the feature dictionary is another possible direction to take in the future. There are a lot of common aspects of WeakIdent and FourierIdent.

There are some similarities between WeakIdent and FourierIdent. SubSpace Pursuit, a greedy algorithm to minimize residual, is utilized for sparsity promotion for the purposes of stabilizing a support. Additionally, the Fourier feature in FourierIdent is motivated by the Weak Forms in WeakIdent to handle noise by carrying out the derivatives computation in the frequency domain instead. The adapted trimming techniques in each method play an important role in refining the support. The *Narrow-fit* in WeakIdent and *Core regions of features* in FourierIdent share a similar idea – only selecting those features located in a meaningful physical or frequency domain to enhance coefficients refinement. However, it's worth noting that identifying and understanding features and equations in the frequency domain itself is a more challenging task, as elaborated in chapter 5. The use of core region of features in FourierIdent is crucial in sparsity selection to address the sensitivity of features in the frequency domain. In terms of future directions for both works, there are a lot of routes to explore. For example, various additional types of differential features and equations could be discovered. Space or time-dependent equations might also be important to discover and identify, particularly in the case that the governing equation may change w.r.t time or space. Exploring governing equations when given data are from real lab experiments instead of simulated equations could also be interesting to discover.

Appendices

APPENDIX A

APPENDIX FOR chapter 2

A.1 StemP on 5s rRNA

A.1.1 Visualization of StemP results

In Figure A.1, we provide the prediction on sequences with Accession Number X67579, AF034620, X01590, AJ251080, V00336 and AE002087 by using StemP. Notice that there is no False Positive pairs found by StemP. The green dash line indicates the False Negative pairs.

A.1.2 12 set of 5s rRNA sequences

In Table A.1, we present the comprehensive prediction results in terms of Sensitivity, PPV and MCC for 12 5s rRNA Bacterial sequences. The sequences is from Gutell Lab [95] in [108]. For each sequence, the top result corresponds to the prediction that has highest MCC values. The best result corresponds to the best prediction that has SCR = 1. For each best prediction, we also provide the Standard Competing Rank (SCR) ("1224"), Dense Rank (DR) ("1223") and the cpu time. The top prediction gives MCC=0.922 as the average, and the average cpu time of 0.143 seconds.

A.1.3 StemP with or without GSL.

We present additional StemP results on 15 number of 5s rRNA sequences, with or without using the GSL which helps to reduce computation by identifying domain β and γ separately. The 15 sequences are from [96] and obtained from Gutell Lab [95]. We adopt the general parameters for 5S rRNA for Archaeal.

We consider StemP both with or without using GSL. Table A.2 shows that among 15

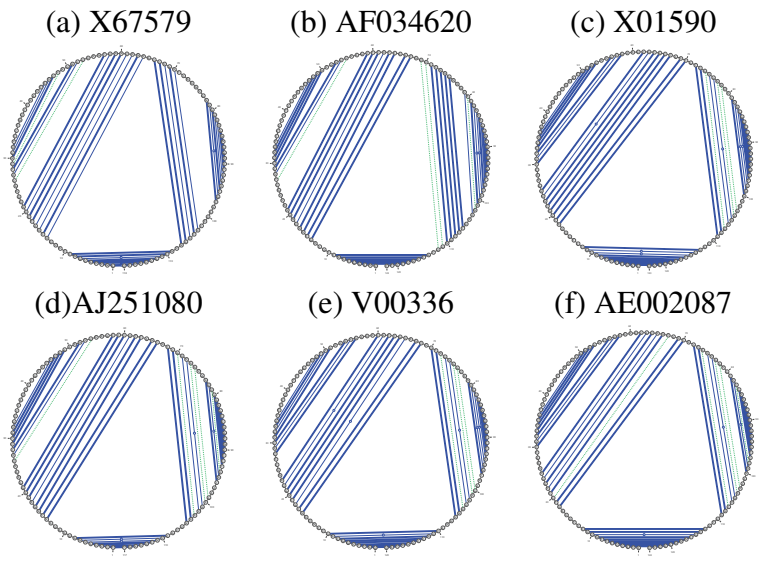


Figure A.1: StemP Results of SPA on 6 5s rRNA sequences. (a)-(f) show the best predicted folding structures by StemP on sequences with Accession Number X67579, AF034620, X01590, AJ251080, V00336 and AE002087. Notice that there is no False Positive pairs found by StemP. The green dash line indicates the False Negative pairs.

sequences, the best F_1 value for 9 sequences are improved when GSL is not used. For example, for sequence M36188, both top and best MCC has increased from 0 to a positive rate. However, both CPU and SCR increased when GSL is not used. When computer power is not limited, the true folding can be close to some clique structure, while GSL helps to reduce the candidate set in general.

A.1.4 Structural information

In Table A.3, we present how the parameters are learned for Archaeal organism 5s RNA sequences, using the data from Gutell Lab [95]. We count different variations of the stems.

A.2 StemP on sequences from Protein Data Bank

We present additional StemP results for RNA sequences with length larger than 50 from (PDB) [77]. In Table A.4, we compare the prediction with FOLD, MaxExpert, Probknot, MC and NAST.

Table A.1: StemP results on 12 number of 5s rRNA Bacterial sequences from Gutell Lab [95] in [108]. These are Bacterial Organism sequences. We use the general parameters of Bacterial Sequences.

	Top Results			Best results (PPV=100)			CPU
Accn	Sens	PPV	MCC	Sens	MCC	SCR/DR	
X08002	91.7	100.0	0.920	84.6	0.920	1/1	0.107
X08000	91.7	100.0	0.920	84.6	0.920	1/1	0.107
X02627	93.3	97.2	0.934	89.7	0.947	3/2	0.223
AJ131594	93.2	97.1	0.932	89.5	0.946	2/2	0.142
V00336	96.1	100.0	0.962	92.5	0.962	1/1	0.224
M10816	91.7	97.1	0.918	86.8	0.932	5/2	0.113
AJ251080	90.4	94.3	0.905	86.8	0.932	31/3	0.128
M25591	90.4	94.3	0.905	86.8	0.932	23/3	0.113
K02682	93.3	97.2	0.934	89.7	0.947	2/2	0.133
X04585	90.4	94.3	0.905	86.8	0.932	7/3	0.122
X02024	90.4	94.3	0.905	86.8	0.932	23/3	0.151
M16532	91.9	97.1	0.920	87.2	0.934	2/2	0.151
Average	92.0	96.9	0.922	87.7	0.936		0.143

Typically, these sequences have a large quantity of possible structures as well as stems of size as small as 1 or 2. For the case where the best prediction does not has SCR = 1, we provide the top prediction here, which is the highest MCC value among possible prediction with maximum number base pairs. These top MCC values are 2FK6 0.55, 3E5C 0.88, 1DK1 0.33, 1MMS 0.34, 3EGZ 0.72, 2QUS 0.00, 1KXK 0.81, 2DU3 0.49 and 2OIU 0.58. The highest accuracy of StemP result MCC is comparable to FOLD, MaxExpect, ProbKnot, MC and NAST in the majority.

A.3 StemP on tRNA

In Table A.5, we present the results of StemP for tRNA folding prediction for 27,010 number of tRNA sequences containing 15 subset of tRNA from The Gutell Lab [95]. In Ta-

Table A.2: StemP results of 15 different 5s rRNA sequences in [96] from Gutell Lab [95]. We use the general parameter and F_1-score as validation measurement. We present StemP with and without GSL restriction for comparison. Among 15 sequences, the best MCC value for 9 sequences improved not using the GSL, while the CPU time increased.

		Standard					without GSL					[96]
	Accn	Top	Best	SCR	DR	CPU	Top	Best	SCR	DR	CPU	
1	X07545	0.85	0.85	1	1	0.11	0.85	0.85	1	1	2.41	0.90
2	X14441	0.68	0.68	1	1	0.07	0.58	**0.84**	7	3	0.18	0.19
3	X72588	0.66	0.68	7	4	0.07	0.84	**0.84**	1	1	0.14	0.20
4	M10691	0.69	0.69	1	1	0.07	0.82	**0.82**	1	1	0.68	0.47
5	M36188	0.00	0.00	1	1	0.08	0.24	**0.44**	77	5	0.24	0.77
6	M26976	0.85	0.86	11	2	0.10	0.85	0.86	11	2	1.50	0.73
7	X62859	0.60	0.66	19	4	0.09	0.59	**0.77**	2	2	8.52	0.63
8	U67518	0.67	0.67	1	1	0.07	0.77	**0.77**	1	1	0.16	0.76
9	M34911	0.26	0.26	1	1	0.07	0.00	**0.60**	93	6	0.16	0.86
10	X62864	0.41	0.45	49	5	0.09	0.41	**0.62**	177	7	0.30	0.55
11	X72495	0.94	0.94	1	1	0.08	0.94	0.94	1	1	0.23	0.94
12	AE009942	0.62	0.62	1	1	0.09	0.77	**0.77**	1	1	1.11	0.89
13	M21086	0.89	0.89	1	1	0.10	0.89	0.89	1	1	1.32	0.88
14	X05870	0.90	0.90	1	1	0.09	0.90	0.90	1	1	0.34	0.88
15	X07692	0.89	0.89	1	1	0.10	0.89	0.89	1	1	1.86	0.87

Table A.3: For Archaeal organism from Gutell Lab [95], we counted different types of stems to make a general and the refined parameter bounds.

	Stem Length Variation (Count)					
Helix I	6 (47)	None (2)	5 (2)	$4_{[1/0]}1$ (1)	4 (1)	
Helix II	$2_{[0/1]}6$ (26)	8 (20)	$2_{[1/2]}5$ (3)	$1_{[1/2]}6$ (2)	$2_{[0/1]}1_{[2/1]}4$(1)	$2_{[0/1]}5$ (1)
Helix III	7 (27)	6 (18)	5 (4)	$2_{[1/1]}2$ (2)	$3_{[1/2]}2$ (1)	$3_{[2/2]}1$ (1)
Helix IV	$2_{[0/2]}4$ (44)	$2_{[2/4]}2$ (5)	$3_{[0/2]}4$ (4)			
Helix V	$1_{[1/1]}6_{[1/0]}2$ (39)	$1_{[1/1]}5_{[2/1]}2$ (4)	$1_{[1/1]}5$ (3)	$1_{[1/1]}8$ (2)	$8_{[1/0]}2$ (2)	$1_{[1/1]}7$(1)
	$1_{[1/1]}6_{[2/0]}1$(1)	8(1)				

ble A.5 (a) shows the SCR values of the best prediction of StemP, (b) MCC values of the top predictions, and (c) MCC values of the best predictions. The second column shows the total number of experiments for each organism. The third column shows the total number of results with the percentage in the parenthesis. The forth column shows additional number of folding results, and the combined percentage in parenthesis. The last column shows the CPU time in seconds computed in average. The bold numbers show when it is near 90%. Notice majority of sequences (many over 85% of the sequence) matched MCC 0.9 or higher. It is shown that there are more than 90% of the sequences that have maximum MCC within top 5 SCR.

(a) The SCR values for the best StemP prediction on tRNA

Organism	**#**	≤ 1	≤ 5	≤ 10	≤ 15	< 15
Alanine	4344	3261(75.1%)	325(82.6%)	108(85.0%)	257(**91.0**%)	393(9.0%)
Asparagine	1250	977(78.1%)	156(**90.6**%)	106 (99.1%)	5(99.5%)	6(0.4%)
Aspartic_Acid	1399	1111(79.4%)	177(**92.1**%)	66 (96.8%)	5(97.1%)	40(2.9%)
Cysteine	596	450(75.5%)	98(**91.9**%)	16(94.6%)	30(99.7%)	2(0.3%)
Glutamic_Acid	1895	1291(68.1%)	492(**94.1**%)	31(95.7%)	2(95.8%)	79(4.2%)
Glutamine	1151	846(73.5%)	229(**93.4**%)	22(95.3%)	4(95.7%)	50(4.3%)
Glycine	2493	1465(58.8%)	892(**94.5**%)	75(97.6%)	12(98.0%)	49(2.0%)
Histidine	987	554(56.1%)	345(**91.1**%)	49(96.1%)	13(97.4%)	26(2.7%)

Isoleucine	45584	046(88.8%)	393(**97.4**%)	23(97.9%)	31(98.6%)	65(1.4%)
Lysine	1566	938(59.9%)	381(84.2%)	42(86.9%)	17(**88.0**%)	188(12.0%)
Methionine	17891	228(68.6%)	273(83.9%)	137(**91.6**%)	15(92.4%)	136(7.6%)
Phenylalanine	26212	279(87.0%)	286(**97.9**%)	11(98.3%)	45(100.0%)	0(0.0%)
Proline	14111	197(84.7%)	139(**94.5**%)	25(96.3%)	16(97.4%)	36(2.6%)
Tryptophan	173	138(79.8%)	25(**94.2**%)	4 (96.5%)	2(97.7%)	4(2.3%)
Tyrosine	777	681(87.7%)	84(**98.5**%)	3(98.8%)	0(98.8%)	9(1.2%)

(b) The MCC values of the top SCR = 1 StemP prediction.

Organism	**#**	≥ 0.95	≥ 0.90	≥ 0.85	≥ 0.80	< 0.80
Alanine	43442	645(60.9%)	508(72.6%)	123(75.4%)	20(75.9%)	1048(24.1%)
Asparagine	1250	865(69.2%)	109(77.9%)	4 (78.2%)	16(79.5%)	256(20.5%)
Aspartic_Acid	1399	966(69.0%)	190(82.6%)	4 (82.9%)	11(83.7%)	228(16.3%)
Cysteine	596	179(30.0%)	225(67.8%)	40(74.5%)	38(80.9%)	114(19.1%)
Glutamic_Acid	18951	065(56.2%)	309(72.5%)	11(73.1%)	307(89.3%)	203(10.7%)
Glutamine	1151	533(46.3%)	389(80.1%)	66(85.8%)	18(87.4%)	145(12.6%)
Glycine	24931	422(57.0%)	184(64.4%)	15(65.0%)	20(65.8%)	852(34.2%)
Histidine	987	407(41.2%)	192(60.7%)	62(67.0%)	29(69.9%)	297(30.1%)
Isoleucine	45583	093(67.9%)	807(85.6%)	21(86.0%)	58(87.3%)	579(12.7%)
Lysine	1566	751(48.0%)	148(57.4%)	0 (57.4%)	47(60.4%)	620(39.6%)
Methionine	17891	140(63.7%)	93 (68.9%)	13(69.6%)	17(70.6%)	526(29.4%)
Phenylalanine	2621	533(20.3%)	1386(73.2%)	39(74.7%)	241(83.9%)	422(16.1%)
Proline	1411	944(66.9%)	252(84.8%)	4 (85.0%)	5 (85.4%)	206(14.6%)
Tryptophan	173	123(71.1%)	7 (75.1%)	0 (75.1%)	0 (75.1%)	43 (24.9%)
Tyrosine	777	717(92.3%)	20 (94.9%)	2 (95.1%)	6 (95.9%)	32 (4.1%)

(c) The MCC values of the best StemP prediction.

Organism	**#**	≥ 0.95	≥ 0.90	≥ 0.85	≥ 0.80	< 0.80 **cpu**
Alanine	43442693(62.0%)	1141(88.3%)	233(**93.6**%)	134(86.7%)	134 (3.4%)	0.24
Asparagine	12501090(87.2%)	127 (**97.4**%)	5(97.8%)	16 (99.0%)	12(1.0%)	0.23
Aspartic_Acid	13991090(77.9%)	249 (**95.7**%)	6(96.1%)	7(96.6%)	47(3.4%)	0.22
Cysteine	596 247 (41.4%)	255 (84.2%)	26 (88.6%)	45 (**96.1**%)	23(3.9%)	0.09
Glutamic_Acid	18951412(74.5%)	328 (**91.8**%)	12 (92.5%)	35 (94.3%)	108(5.7%)	0.19
Glutamine	1151 556 (48.3%)	449 (87.3%)	75 (**93.8**%)	14 (95.0%)	57(5.0%)	0.11
Glycine	24932008(80.5%)	314 (**93.1**%)	46 (95.0%)	82 (98.3%)	43(1.7%)	0.14
Histidine	987 713 (72.2%)	105 (82.9%)	71 (**90.1**%)	27 (92.8%)	71(7.2%)	0.19
Isoleucine	45583267(71.7%)	970 (**93.0**%)	53 (94.1%)	94 (96.2%)	174 (3.8%)	0.18
Lysine	15661087(69.4%)	247 (86.8%)	25 (86.8%)	49 (**89.9**%)	158 (10.1%)	0.25
Methionine	17891399(78.2%)	118 (84.8%)	17 (85.7%)	38 (**87.9**%)	217 (12.1%)	0.17
Phenylalanine	2621 533 (20.3%)	1430(74.9%)	139(80.2%)	371(**94.4**%)	148 (5.6%)	0.04
Proline	14111023(72.5%)	310 (**94.5**%)	19 (95.8%)	5(96.2%)	54(3.8%)	0.19
Tryptophan	173 127 (73.4%)	11(79.8%)	0(79.8%)	23 (**93.1**%)	12 (6.9%)	0.16
Tyrosine	777 726 (**93.4**%)	28(97.0%)	5(97.7%)	4(98.2%)	14(1.8%)	0.15

Table A.5: StemP for 27,010 different tRNA sequences. (a) The SCR values for the best StemP prediction on tRNA. (b) The MCC values of the top SCR = 1 StemP prediction. (c) The MCC values of the best StemP prediction.

A.4 StemP comparison on unknown family.

In Table A.6, we present results of StemP on general sequences with unknown structures. This is the detailed version of Table 11 in the main paper. We also experimented 92 sequences with length in $[120, 130]$ using 5s Bacteria finding parameters in Table 8, resulting 7 sequences with $F_1 \geq 0.89$ and total average 0.37. Similar results on these sequences can be obtained using parameters for Archaeal or Eukaryotic in Table 6 in the main draft.

Table A.4: StemP for RNA sequences of length over 50 from Protein Data Bank. In the first column, superscript p indicates pseudo knots, superscript w indicates we includ wobble base pairs, and u includes UU base pairs. Superscript L indicates using $L = 2$ otherwise $L = 3$, and S indicates when SL is used. For 1KXK, the superscript N indicate that if SL is not imposed, it take 89s to get the same result. In the FOLD column, m represents picking the best MCC among multiple possible predictions: for 1DK1, the best MCC among (0.69, 0.21, 0.34), for 2QUS, among (0.93, 0.73), for 2DU3, among (0.42, 0.82), and for 2OIU, among (0.88, 0.50, 0.74, 0.71). In MaxExpect column for 3E5C, m represents picking the best MCC among multiple possible predictions (0.65, 0.97).

RNA	Size	StemP	SCR(m)	DR	CPU(s)	FOLD	MaxExpect	ProbKnot	MC	NAST
3E5C	53	**0.97**^{L}	9(69)	2	6.43	**0.94**^{m}	**0.97**^{m}	0.81	O	O
1MZP ^{u}	55	**0.89**	53(49)	4	0.12	0.36	0.36	0.36	X	O
1DK1 ^{w}	57	**0.83**^{S}	2(12)	2	10.61	0.69	0.72	0.82	O	O
1MMS	58	**0.87**^{S,L}	477(643)	6	7.80	0.71	0.73	0.69	O	X
3EGZ ^{p}	65	0.80	11(36)	3	0.60	**0.84**	0.81	0.81	X	O
2QUS ^{p}	69	**0.95**	465(250)	6	2.00	0.93	0.93	0.93	O	O
1KXK ^{w}	70	**0.96**^{S}	3(11)	2	11.16^{N}	0.81	0.81	0.79	O	O
2DU3 ^{p}	71	**0.90**^{S}	745(917)	6	3.608	0.82	0.47	0.49	X	O
2OIU ^{w}	71	**0.94**^{S}	7(25)	3	16.62	0.74	0.93	0.93	O	X
2FK6 ^{p}	79	**0.83**	10065(1991)	10	35.49	0.76	0.76	0.78	O	X

The comparison results of Ufold[4], RNAstructure[39], RNAsoft[112], e2efold[113], Eternafold[5], Linearfold[45], and Mfold[42] from Ufold [4], and results of MXfold2[109], SPOT-RNA[63], TORNADO[111], ContextFold[51], RNAfold[207, 104] are from [109] where sequence-wise cross validation for TS0 and family-wise cross validation for bpRNA-new respectively. These results use all data from the two sets, while StemP experiments are only for shorter sequences as mentioned. For longer sequences, not using SL poses great computational costs for StemP.

TS0	Count	%F1	%Sen	%PPV
StemP(Best)$[1, 90]$	468	**0.714**	**0.775**	**0.701**
Ufold		0.654	-	-
RNAstructure		0.532	-	-
RNAsoft		0.535	-	-
e2efold		0.189	-	-
Eternafold		0.563	-	-
Linearfold		0.551	-	-
Mfold		0.538	-	-
MXfold2		0.575	0.520	0.682
SPOT-RNA		0.597	0.652	0.578
TORNADO		0.561	0.554	0.609
ContextFold		0.575	0.583	0.595
RNAfold		0.446	0.631	0.508

bpRNA-new	Count	%F1	%Sen	%PPV
StemP(Best)$[1, 87]$	2483	**0.737**	**0.771**	**0.726**
Ufold		0.635	-	-
RNAstructure		0.629	-	-
RNAsoft		0.620	-	-

e2efold	0.036	-	-
Eternafold	0.647	-	-
Linearfold	0.633	-	-
Mfold	0.623	-	-
MXfold2	0.632	0.585	0.710
SPOT-RNA	0.596	0.599	0.619
TORNADO	0.620	0.636	0.638
ContextFold	0.554	0.595	0.539
RNAfold	0.617	0.552	0.720

Table A.6: Comparison on a TS0, a subset of bpRNA-1m dataset[110], used as a test set in [109] and bpRNA-new from [109, 110]. The comparison results of Ufold[4], RNAstructure[39], RNAsoft[112], e2efold[113], Eternafold[5], Linearfold[45], and Mfold[42] from Ufold [4], and results of MXfold2[109], SPOT-RNA[63], TORNADO[111], ContextFold[51], RNAfold[207, 104] are from [109]. These results are using all data from the two sets, while StemP experiments are only for shorter sequences as listed.

APPENDIX B

APPENDIX FOR chapter 4

B.1 Proof of Theorem 4.2.1

Proof. (a) Using the noisy data in the form $\hat{U}_i^n = U_i^n + \epsilon_i^n$ in (Equation (4.3)), the hth entry of $\boldsymbol{e}^{\text{noise}}$ can be expressed as

$$
\begin{aligned}
e_h^{\text{noise}} = \; & \Delta x \Delta t \sum_{(x_j,t^k)\in\Omega_{h(x_i,t^n)}} \\
& \left(\sum_{l\in \text{Supp}^*} (-1)^{\alpha_l} c_l \left((U_j^k + \epsilon_j^k)^{\beta_l} - (U_j^k)^{\beta_l} \right) \frac{\partial^{\alpha_l}\phi_h}{\partial x^{\alpha_l}}(x_j,t^k) + (\hat{U}_j^k - U_j^k)\frac{\partial \phi_h}{\partial t}(x_j,t^k) \right) \\
= \; & \Delta x \Delta t \sum_{(x_j,t^k)\in\Omega_{h(x_i,t^n)}} \\
& \left(\sum_{l\in \text{Supp}^*} (-1)^{\alpha_l} c_l \epsilon_j^k \left(\sum_{r=1}^{\beta_l} \binom{\beta_l}{r} (\epsilon_j^k)^{r-1} (U_j^k)^{\beta_l - r} \right) \frac{\partial^{\alpha_l}\phi_h}{\partial x^{\alpha_l}}(x_j,t^k) + \epsilon_j^k \frac{\partial \phi_h}{\partial t}(x_j,t^k) \right) \\
= \; & \Delta x \Delta t \sum_{(x_j,t^k)\in\Omega_{h(x_i,t^n)}} \\
& \left(\sum_{l\in \text{Supp}^*} (-1)^{\alpha_l} c_l \beta_l (U_j^k)^{\beta_l - 1} \frac{\partial^{\alpha_l}\phi_h}{\partial x^{\alpha_l}}(x_j,t^k) + \frac{\partial \phi_h}{\partial t}(x_j,t^k) \right) \epsilon_j^k \; + \mathcal{O}\left((\epsilon_j^k)^2 \right).
\end{aligned}
\tag{B.1}
$$

Hence,

$$
\|\boldsymbol{e}^{\text{noise}}\|_\infty = \max_h |e_h^{\text{noise}}| \leq \max_h \left[\epsilon \bar{S}_h^* |\Omega_h| \right] + \mathcal{O}\left(\epsilon^2\right)
$$

where

$$\bar{S}_h^* = \sup_{(x_j,t^k)\in\Omega_h} \left| \sum_{l\in \text{Supp}^*} (-1)^{\alpha_l} c_l \beta_l (U_j^k)^{\beta_l - 1} \frac{\partial^{\alpha_l}\phi_h}{\partial x^{\alpha_l}}(x_j,t^k) - \frac{\partial \phi_h}{\partial t}(x_j,t^k) \right|.$$

Setting $\bar{S}^* = \max_h \bar{S}_h^*$ as in (Equation 4.16) gives rise to our estimate in (Equation 4.15).

(b) From (Equation B.1), the leading term in e_h^{noise} is

$$\Delta x \Delta t \sum_{(x_j,t^k)\in\Omega_{h(x_i,t^n)}} \left(\sum_{l\in \text{Supp}^*} (-1)^{\alpha_l} c_l \beta_l (U_j^k)^{\beta_l - 1} \frac{\partial^{\alpha_l}\phi_h}{\partial x^{\alpha_l}}(x_j,t^k) + \frac{\partial \phi_h}{\partial t}(x_j,t^k) \right) \epsilon_j^k.$$

Based on our noise assumption, this leading term e_h^{noise} has mean 0, and variance $\sigma^2 S_h^*$ with S_h^* given in (Equation 4.17).

□

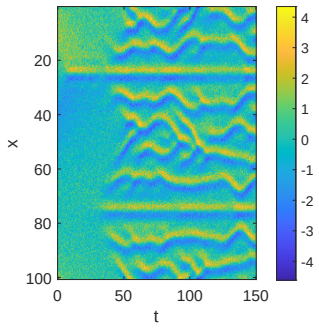
B.2 Additional results and comparisons

B.2.0 Additional results and comparisons for PDEs

In Figure B.1, we experiment on the KS equation (Equation (4.34)) with $\sigma_{\text{NSR}} = 0.5$. We compare WeakIdent with WPDE and RGG. Figure B.1 (a) shows noisy data $\hat{U}(x,t)$ and (b) gives the recovered equation with the E_2 error. WeakIdent finds correct support with a small error $E_2 = 0.08831$.

In Figure B.2, we show the identification results for the nonlinear Schrodinger equation (Equation 4.35). Table (c) shows the results from noise-free data with $\sigma_{\text{NSR}} = 0$, and Table (d) shows the noisy case with $\sigma_{\text{NSR}} = 0.1$. Figure B.2 (a) and (b) show the noisy data $\hat{U}(x,t)$ and $\hat{V}(x,t)$, and the tables (c) and (d) show the identified equations for WeakIdent, WPDE and RGG. WeakIdent finds the correct support with small errors in both the noise-free and noisy cases.

(a) $\hat{U}(\boldsymbol{x}, t)$



(b) $\sigma_{\text{NSR}} = 0.5$

True equation	$u_t = -1.00000u_{xx} - 1.00000u_{xxxx} - 0.50000(u^2)_x$	
WeakIdent	$u_t = -0.91387u_{xx} - 0.90906u_{xxxx} - 0.45686(u^2)_x$	$E_2 =$ **0.08831**
WPDE[36]	$u_t = -1.03383u_{xx} - 1.27316u_{xxxx} - 0.51246(u^2)_x$ $-0.24197u_{xxxxxx} - 0.11572(u^2)_{xxx}$	$E_2 = 0.25635$
RGG [38]	$u_t = 0.90654u_{xx} + 1.18313u_{xxxx} + 0.58744(u^2)_x$ $+0.00906u - 0.02481u_x + 0.34133u_{xxx} + 0.00175u^2 -$ $0.00952u^3$	$E_2 = 0.33155$

For RGG [38], 8 default features $\{uu_x, u_{xx}, u_{xxxx}, u, u_x, u_{xxx}, u^2, u^3\}$ and parameters $p_x = 4, p_t = 3, N_d = 100, D = (40, 20)$ are used, as in the case for transport equation (Equation 4.32) in Figure 4.5.

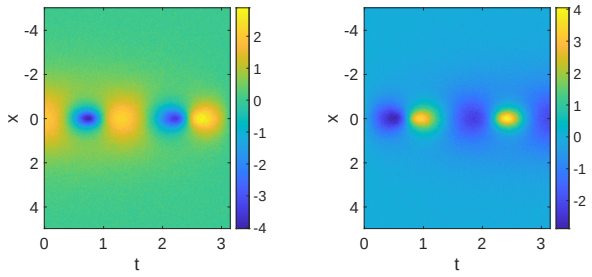
Figure B.1: KS equation (Equation 4.34) with $\sigma_{\text{NSR}} = 0.5$. (a) Given noisy data $\hat{U}(\boldsymbol{x}, t)$. (b) The identified equations using WeakIdent, WPDE[36] and RGG [38] where the E_2 error is given in the right column.

B.2.0 Additional results and comparisons for ODEs

In Figure B.3, we present the identification results for the ODEs in Table 4.2: the linear system (Equation 4.39), Van der Pol (Equation 4.40), Duffing (Equation 4.41), Lotka-Volterra (Equation 4.42), and Lorenz (Equation 4.43). We experiment with different noise levels with different methods, including WeakIdent, WODE[37], SINDy[22], SC [30], and ST [30]. Figure B.3 shows the median of the E_2 error, TPR and PPV over 50 experiments for each equation. Overall WeakIdent (light green curves) yields the lowest E_2 error in the first column, and the TPR and PPV values near 1, which demonstrates a good support recovery.

In Table B.1, we present the detailed results for the data in Figure 4.11(d). The noise level is $\sigma_{\text{NSR}} = 0.1$. Table B.1 (a) shows the dynamics. The noisy data for each of the

(a) $\hat{U}(\boldsymbol{x},t)$ (b) $\hat{V}(\boldsymbol{x},t)$



(c) $\sigma_{\text{NSR}} = 0$

True equation	$u_t = +0.50000v_{xx} + 1.00000v^3 + 1.00000u^2v$ $v_t = -0.50000u_{xx} - 1.00000uv^2 - 1.00000u^3$	
WeakIdent	$u_t = +0.50000v_{xx} + 1.00000v^3 + 1.00000u^2v$ $v_t = -0.50000u_{xx} - 1.00000uv^2 - 1.00000u^3$	$E_2 =$ **9.4887e-08**
WPDE[36]	$u_t = +0.50000v_{xx} + 1.00000v^3 + 1.00000u^2v$ $v_t = -0.50000u_{xx} - 1.00000uv^2 - 1.00000u^3$	$E_2 = 1.3254e-07$
RGG [38]	$u_t = +0.49996v_{xx} + 0.99986v^3 + 0.99988u^2v$ $u_t = -0.50004u_{xx} - 0.99990u^3 - 0.99988uv^2$	$E_2 = 1.2391e-4$

(d) $\sigma_{\text{NSR}} = 0.1$

WeakIdent	$u_t = +0.49872v_{xx} + 0.98737v^3 + 1.00001u^2v$ $v_t = -0.49977u_{xx} - 1.01550uv^2 - 0.98785u^3$	$E_2 =$**0.011**
WPDE[36]	$u_t = +0.49868v_{xx} + 0.98722v^3 + 1.00068u^2v$ $v_t = -0.06604 + 0.10805v^2 + 0.14935v_{xx}^2 - 0.04445u_{xxxx} +$ $0.04995uv_{xx} - 0.98964uv^2 - 0.14989uv_{xx}^2 - 0.01231uv_{xx}^3 +$ $0.08142u^2 - 0.03497u^2v^2 + 0.06269u^2v_{xx}^2 - 0.01519u^2v_{xx}^4 -$ $0.97427u^3 - 0.09657u_{xx}^3 - 0.01753u^3v^2 + 0.01367u^3v_{xx}^2$	$E_2 =$0.388
RGG [38]	$u_t = +0.09242(u^2)_x + 0.13394u_{xx} + 0.59688u +$ $0.20176u_x - 3.29871u^2 + 2.49062u^3 - 0.42728(v^2)_x +$ $0.17625v_{xx} - 2.52277v + 0.27970v_x - 11.82022v^2 +$ $5.53290v^3 - 5.43222uv + 10.31760u^2v + 16.92442uv^2$ $u_t = +0.09242(u^2)_x + 0.13394u_{xx} + 0.59688u +$ $0.20176u_x - 3.29871u^2 + 2.49062u^3 - 0.42728(v^2)_x +$ $0.17625v_{xx} - 2.52277v + 0.27970v_x - 11.82022v^2 +$ $5.53290v^3 - 5.43222uv + 10.31760u^2v + 16.92442uv^2$	$E_2 =$17.668

For RGG [38], we add an additional dictionary to include the correct features. We use a dictionary of 19 features: $\{(u^2)_x, u_{xx}, u_{xxxx}, u, u_x, u_{xxx}, u^2, u^3, (v^2)_x, v_{xx}, v_{xxxx}, v, v_x, v_{xxx}, v^2, v^3, uv, u^2v, uv^2\}$.

Figure B.2: Nonlinear Schrodinger equation (Equation 4.35) with two variables. The given noisy data $\hat{U}(\boldsymbol{x},t)$ and $\hat{V}(\boldsymbol{x},t)$ are shown in (a) and (b) respectively. Table (c) and (d) show the identified equations using WeakIdent, WPDE and RGG with $\sigma_{\text{NSR}} = 0$ and $\sigma_{\text{NSR}} = 0.1$.

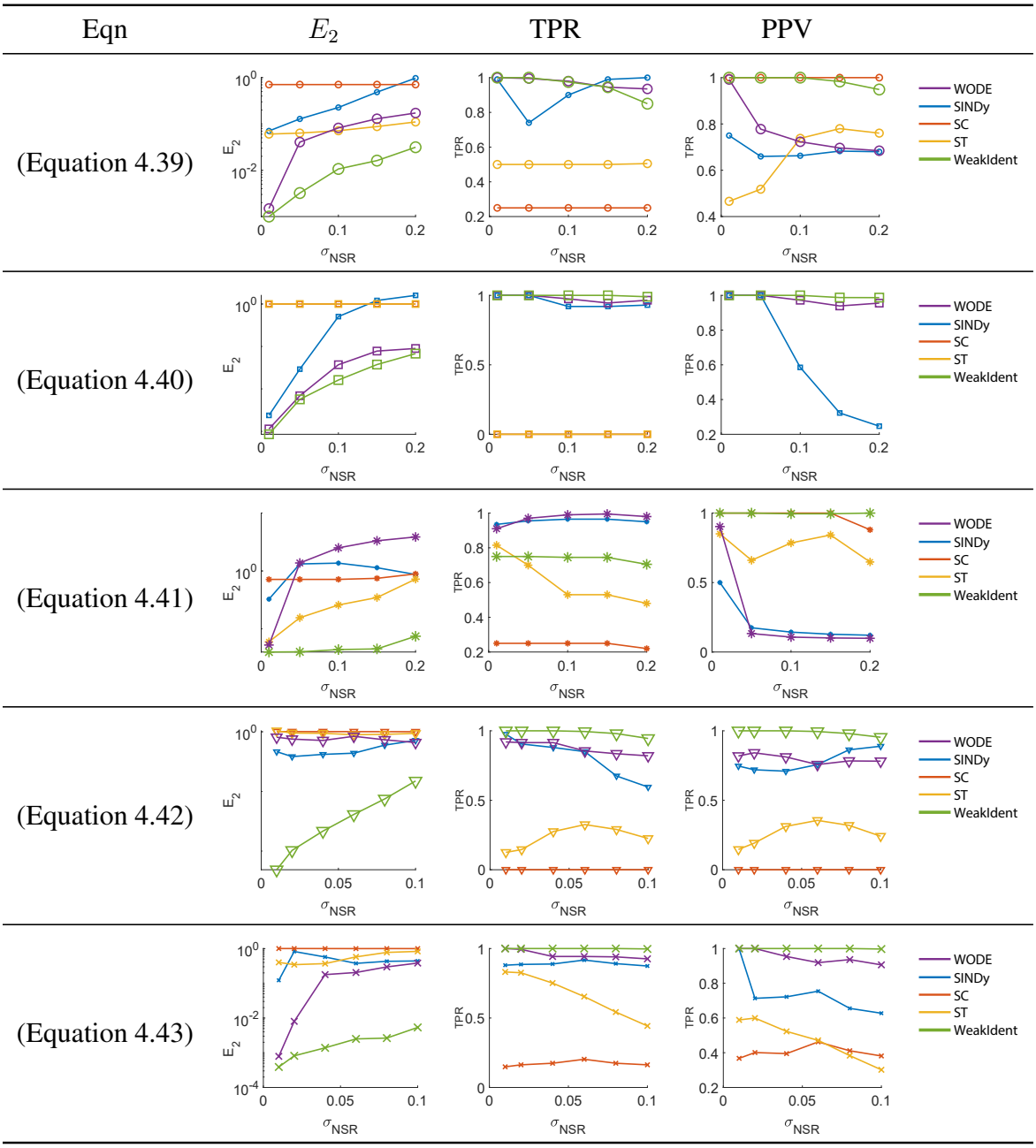
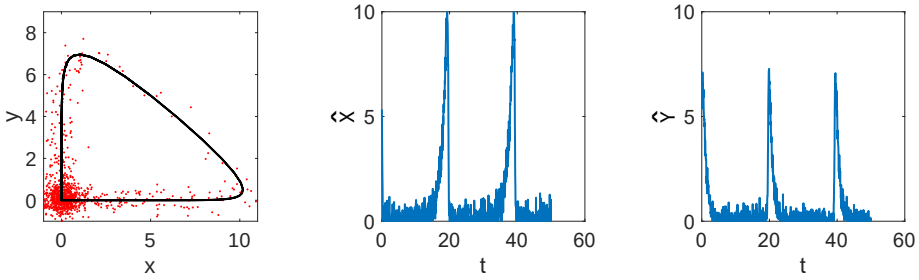


Figure B.3: WeakIdent results for the identification of ODEs listed in Table 4.2, with the noise level σ_{NSR} from 0 to 0.1. Each graph shows the median over 50 experiments on each equation using WODE (purple), SINDy (blue), SC (red), ST (yellow), and WeakIdent (green). Each column shows the E_2 error, the TPR and PPV values. The green curve of WeakIdent gives the lowest E_2 error and the TPR and PPV values are close to 1.

dependent variables $\hat{X}$ and $\hat{Y}$ are shown in (b) and (c) respectively. Table B.1 shows the identified systems by WeakIdent, WODE, SINDy, SC and ST with the E_2 and E_{dyn} errors, TPR and PPV. WeakIdent gives the most accurate recovery.

(a) Equ.(Equation 4.42) (b) $\hat{X}$ (c) $\hat{Y}$



Method	Equation(s)	E_2	E_{dyn}	TPR	PPV
True equation	$\dot{x} = +0.66667x - 1.33333xy$ $\dot{y} = -1.00000y + 1.00000xy$				
WeakIdent	$\dot{x} = +0.61271x - 1.21729xy$ $\dot{y} = -0.97002y + 0.92339xy$	**0.07**	**0.65**	**1.00**	**1.00**
WODE[37]	$\dot{x} = +0.62699x - 0.20740x^2y$ $\dot{y} = -0.91948y + 0.91515xy$	0.73	2.34	0.75	0.75
SINDy[198]	$\dot{x} = +0.65909x - 1.05803xy$ $\dot{y} = +0.61390xy$	0.58	5.57	0.75	1.00
SC[30]	$\dot{x} = -0.05204y^3$ $\dot{y} = 0$	1.00	112.17	0.00	0.00
ST[30]	$\dot{x} = -0.05204y^3$ $\dot{y} = +0.91700xy + 0.01157xy^2 - 0.05996x^2y$	0.88	3.86	0.25	0.25

Table B.1: The Lotka-Volterra equation (Equation 4.42) with $\sigma_{\text{NSR}} = 0.1$. We use the same data as in Figure 4.11(d). We present the comparisons between WeakIdent and WODE [37], SINDy [198], SC, ST [30].

Table B.2 shows the detailed results for the Lorenz system (Equation 4.43). The data set is the same as the one in Figure 4.11(e) with $\sigma_{\text{NSR}} = 0.1$.. The noisy data $\hat{X}, \hat{Y}, \hat{Z}$ are displayed in (a), (b), (c) respecitvely. Table B.2 shows the recovered equations. Table B.2 provides more details associated with Figure B.4 where we present statistical comparisons using 50 experiments for various noise level when σ_{NSR} varies from 0.01 to 0.1. The E_2 error by WeakIdent is lower with less variations, and the TPR and PPV values are closer to

1 compared to other methods.

B.2.1 Test functions and feature construction

We give an outline of the construction of test functions in (Equation 4.10). Specifically, we discuss how to choose the parameters m_x, m_t, p_x, p_t (simply m and p below) according to [36]:

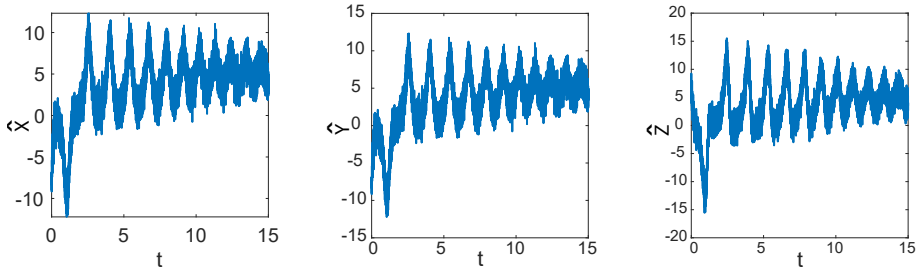
(1) Frequency consideration: Given the data $\{U_i^n\}$, we consider the Fourier transform of data in each dimension. For example, $\mathcal{F}_x(U)$ is the Fourier transform of U is the spatial domain. We next find a junction point k_x^* by fitting the cumulative sum of the vectorized data $|\mathcal{F}_x(U)|$ by a piecewise linear polynomial with one junction point. The k_x^* minimizes the L_2 fitting error.

(2) Fit with a Gaussian distribution: The test function ϕ is matched to Gaussian for a denoising effect, i.e., $\phi_p(x) = C\left(1 - \left(\frac{x}{m\Delta x}\right)^2\right)^p \approx \rho_\sigma(x)$ where $\rho_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}(\frac{x}{\sigma})^2}$ and $\sigma = \frac{m\Delta x}{\sqrt{2p+3}}$. Here $\phi_p(x)$ matches ρ_σ up to the third moment such that $|\hat{\phi}_p(\xi) - \hat{\rho}_\sigma(\xi)| \leq \mathcal{O}(|\xi|^4 (m\Delta x)^4 p^{-3})$ and C is a constant such that $||\phi_p||_1 = 1$ [36]. Here $\hat{\phi}_p$ and $\hat{\rho}_\sigma$ denotes the Fourier transform of ϕ_p and ρ_σ respectively. To suppress the noise, the high frequency components of data with the mode larger than k_x^* or smaller than $-k_x^*$ are set to be within the 5% tail of the Gaussian. This gives $\frac{2\pi}{\mathbb{N}_x \Delta x} k_x^* = \frac{2}{\sigma}$ from the property of cumulative distribution function of $\hat{\rho}_\sigma = \hat{\rho}_{1/\sigma}$, and relating this to p and m gives the first condition: $\frac{2\pi}{\mathbb{N}_x \Delta x} k_x^* = \hat{\tau} \frac{\sqrt{2p+3}}{m\Delta x}$, where $\hat{\tau}$ is a parameter [36].

(3) Vanishing of ϕ on the boundary To guarantee the decay of ϕ in each spatial domain, p and m are set to satisfy the second condition: $\phi_p((m-1)\Delta x) \leq 10^{-10}$ and $p > \alpha_x + 1$ where α_x is the highest order derivative in the x direction for all features. Using the first and the second conditions above, p and m are determined.

The computation of the features in (Equation 4.9) is done by convolution in each

(a)$\hat{X}$ (b) $\hat{Y}$ (c)$\hat{Z}$



(d)

Method	Equation(s)	E_2	TPR	PPV
True equation	$\dot{x} = +5.00000y - 5.00000x$ $\dot{y} = -1.00000y + 15.00000x - 1.00000xz$ $\dot{z} = -2.00000z + 1.00000xy$			
WeakIdent	$\dot{x} = +4.97854y - 4.97406x$ $\dot{y} = -0.97267y + 14.88230x - 0.99353xz$ $\dot{z} = -1.96169z + 0.98105xy$	**0.011**	**1.00**	**1.00**
WODE[37]	$\dot{x} = -4.12201 + 2.77136y + 0.39253y^2 - 2.20585x - 0.89285xy + 0.45249x^2$ $\dot{y} = -11.16919 + 3.46066z - 0.27237z^2 + 3.89361y - 0.35785yz + 6.97398x - 0.47972xz - 0.55649xy + 1.16625x^2$ $\dot{z} = -12.63547 + 0.71043y - 1.07591x + 1.36546xy - 0.72687x^2$	0.018	1	0.88
SINDy[198]	$\dot{x} = +4.94736y - 4.91610x$ $\dot{y} = -0.99564y + 14.88425x - 0.99267xz$ $\dot{z} = +0.29463 - 2.01828z + 1.00702xy$	1.188	0.86	0.3
SC[30]	$\dot{x} = +0.02392xy^2$ $\dot{y} = -0.00906xz^2$ $\dot{z} = -0.27194z^2 + 0.01103z^3 + 0.05702xyz$	1.000	0.00	0.00
ST[30]	$\dot{x} = +0.03245x^2y$ $\dot{y} = -0.00906xz^2$ $\dot{z} = +0.02673y^2z$	1.000	0.00	0.00

Table B.2: The Lorenz equation (Equation 4.43) with $\sigma_{\text{NSR}} = 0.2$. This experiment shows the comparisons of WeakIdent, WODE, SINDy, SC and ST using the same data set from Figure 4.11(e). WeakIdent gives rise to the best recovery.

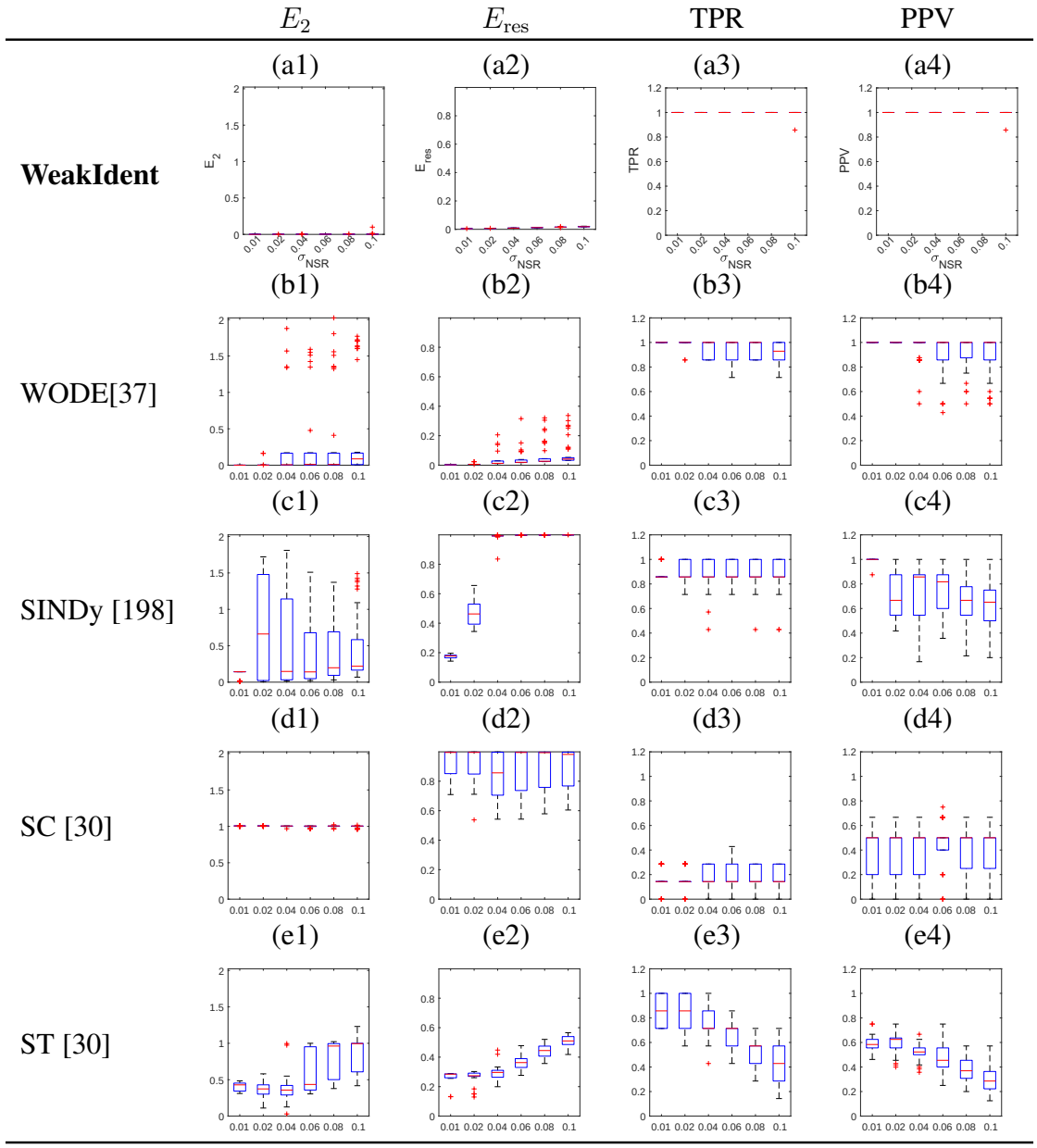


Figure B.4: The Lorenz equation ((Equation 4.43)), statistical comparisons: WeakIdent (a1)-(a4), WODE [37] (b1)-(b4), SINDy [198] (c1)-(c4), SC [30](d1)-(d4) and ST [30](e1)-(e4). The E_2, E_{res} errors , TPR and PPV are shown from 50 experiments for each $\sigma_{\text{NSR}} \in \{0.01, 0.02, , ..., 0.1\}$ using box-plots. The E_2 error given by WeakIdent is lower than others with less variations, and the TPR and PPV by WeakIdent are closer to 1 compared to other methods.

dimension:

$$w_{h(x_i,t^n),l} = (-1)^{\alpha_l}\left(U * \frac{\partial^{\alpha_l}\phi}{\partial x^{\alpha_l}}\right)(x_i,t^n) = (-1)^{\alpha_l}\sum_{k=n-m_t}^{n+m_t}\sum_{j=i-m_x}^{i+m_x} U_j^k \frac{\partial^{\alpha_l}}{\partial x^{\alpha_l}}\phi(x_j - x_i, t^k - t^n),$$

and there is a similar form for $b_{h(x_i,t^n)}$ for each (x_i,t^n) in the domain. FFT is applied to compute the convolution. For the convolutions in the x direction when $t = t^n$, we use the vector $\boldsymbol{\phi} = (\phi(-m_x\Delta x, t^n), ..., \phi(m_x\Delta x, t^n))$ for convolution. Near the boundary, we pad the data by zeros for the computation of convolutions.

APPENDIX C

APPENDIX FOR chapter 5

In this Appendix, we present some more details of the FourierIdent.

C.1 Effect of the core region of feature u_t

In Figure C.1, we present the difference between the core region of feature u_t from FourierIdent and the high dynamic region in physical domain given by uu_x as in WeakIdent [3]. The KdV equation (Equation 5.41) with $\sigma_{NSR} = 0.3$ is used, and it is shown in (a). We introduce $\boldsymbol{e}_{\text{given}}$ to describe the point-wise relative error of an approximated discrete system: for each frequency mode $h = \mathcal{H}(\xi_x, \xi_t)$,

$$e_{given}(\xi_x, \xi_t) = \frac{\mathcal{S}(\boldsymbol{F})_h \cdot \boldsymbol{c}_{true} - \mathcal{S}(\boldsymbol{b})_h}{\mathcal{S}(\boldsymbol{b})_h}. \tag{C.1}$$

It approximates the error of a constructed discrete system at this mode. Here $\mathcal{S}(\boldsymbol{F})_h$ and $\mathcal{S}(\boldsymbol{b})_h$ represent the $\mathcal{H}(\xi_x, \xi_t)$-th row of $\mathcal{S}(\boldsymbol{F})$ and $\mathcal{S}(\boldsymbol{c})$, and $\boldsymbol{c}_{true}$ represents the true coefficient vector.

Figure C.1 (b) shows the error of the core regions of feature u_t, and (c) shows the highly dynamic region selected by WeakIdent. Both FourierIdent and WeakIdent choose the region where the residual error e_{given} tends to be small, while FourierIdent's choice is more of a partial region depending on the high response modes in $\mathcal{V}(u_t)$. The pattern of the highly dynamic regions in the physical domain exhibits more variation in terms of scales from 10^{-4} to 10^{1} with more points located in $\mathbb{H}$ (Highly dynamic region of WeakIdent). In contrast, FourierIdent has fewer points located in $\mathcal{V}_{u_t}$ with less variations, which makes identification problems more challenging in the frequency domain.

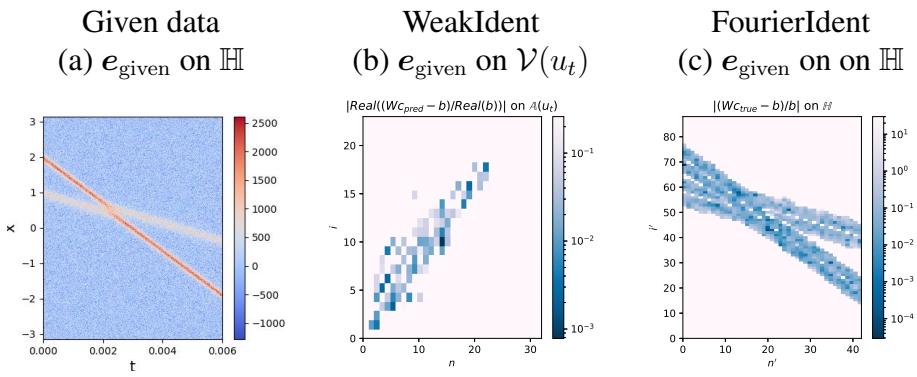


Figure C.1: Observation of the given error $|\frac{\boldsymbol{W}\boldsymbol{c}_{true}-\boldsymbol{b}}{\boldsymbol{b}}|$ in the highly dynamic region in the physical domain (WeakIdent) v.s. $\boldsymbol{e}_{\text{given}}$ on the core regions in the frequency domain (FourierIdent). (a) represents given data of the KdV equation in (Equation 5.41) with $\sigma_{NSR} = 0.3$. (b) shows the Highly dynamic region in blue (the mild dynamic region in white). (b) shows the given error $\boldsymbol{e}_{\text{given}}$ of the FourierIdent feature matrix (real part) on active response modes $\mathcal{V}(u_t)$ in blue (otherwise in white). Note that the results for the feature matrix in the imaginary part are similar to those in the real part. The patterns of active response mode are relatively similar in FourierIdent, while those of highly dynamic regions in the physical domain (WeakIdent) depends on the given data. The given error of the feature matrix in the frequency domain indicates the connection between the core regions and highly dynamic region.

C.2 Core region of features in finding an optimal coefficient

The right-hand side of (Equation 5.28) can be understood as measuring how well a predicted coefficient vector fits a certain collection of modes. We show an example of the different potential collections of responses with $\boldsymbol{e}_{\text{given}}$ in (Equation C.1) in Figure C.2, where (b), (c), and (d) provide the region of these potential responses in the frequency domain. It is shown that (a) consists of a larger area than an individual core regions region or the union of core regions region. The dark blue region in (a), while not included in any of (b)-(d), may be considered as those regions with mild dynamics in the physical domain. They are a good approximation based on $\boldsymbol{e}_{\text{given}}$ but may not be meaningful in identifying correct support. Hence, in Step 3 of FourierIdent, the $\mathcal{V}^*$ is carefully selected to provide more accurate identification results.

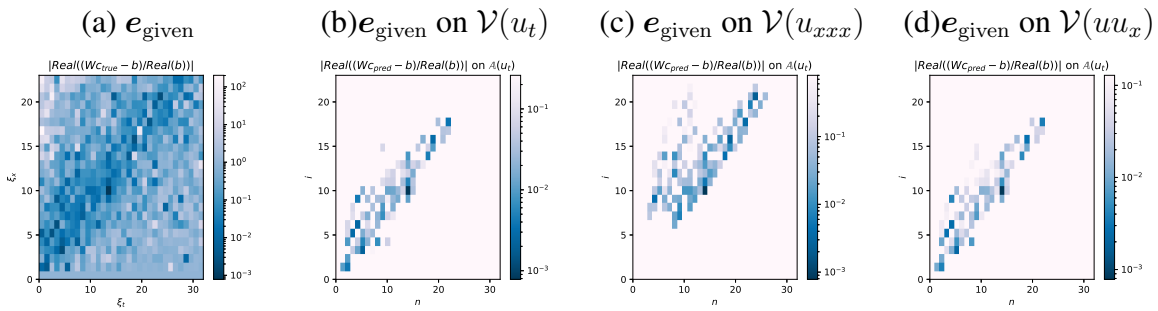


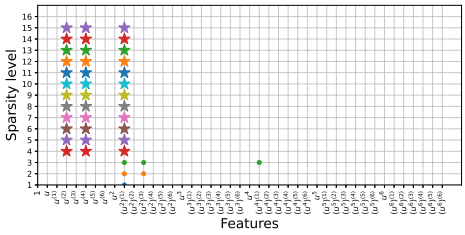
Figure C.2: Visualization of the patterns of active frequency modes. We take the KdV equation with $\sigma_{NSR} = 0.3$ as an example. We show the pattern of active response modes in green on each true feature in the examples. In each figure (b) - (d), the x and y axis represent the frequency mode in Λ. The different values of numbers represent the relative residual of the feature matrix in this mode. The white region represents the frequency mode that is not active in a particular feature u_t (in (b)), u_{xxx} (in (c)), and uu_x (in (d)).

C.3 Effect of SP and Group trimming

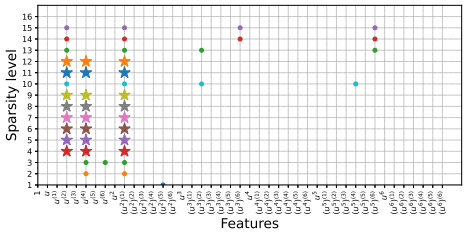
As in the example of subsection 5.6.1, SP and group trimming gives a stable recovery of the support.

In Figure C.3, we use two examples to show the effect of the proposed group trimming method. In this example, We take the KS equation in (Equation 5.42) with clean data and noisy data with $\sigma_{NSR} = 0.8$. The first row provides results from FourierIdent. In addition, we show results from WeakIdent in the same experiment in the second row. In each figure, the x-axis and y-axis represent an individual feature and a sparsity level. A symbol of a circle or star is shown if a feature has been selected after group trimming. These circles will be replaced by stars if the reduced support matches with the exact true support. In this example, the true features are $(u^2)_x, u_{xx}, u_{xxxx}$. It is shown that both FourierIdent and WeakIdent can reduce larger support to the correct one in the majority of the cases with or without noise.

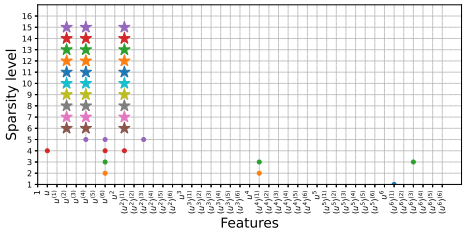
(a) Equation 5.42 (FourierIdent) $\sigma_{NSR} = 0$



(b) Equation 5.42 (FourierIdent) $\sigma_{NSR} = 0.8$



(c) Equation 5.42 (WeakIdent) $\sigma_{NSR} = 0$



(d) Equation 5.42 (WeakIdent) $\sigma_{NSR} = 0.8$

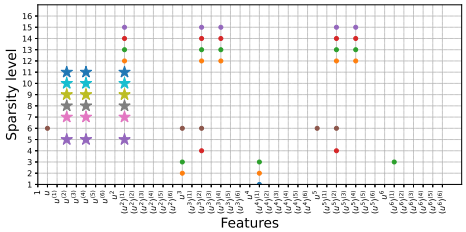


Figure C.3: Benefit of Group trimming. Visualize reduced supports from different sparsity levels using Group trimming in FourierIdent v.s. using Trimming in WeakIdent on Clean and Noisy datasets. We take the KS equation in (Equation 5.42) with $\sigma_{NSR} = 0$ and $\sigma_{NSR} = 0.8$ as two examples. Each row provides the features in reduced support from sparsity level $k = 1, 2, ..., 15$. If a feature is shown as a star, the current support is reduced to correct support. It is shown that Group Trimming in FourierIdent is as stable as individual feature trimming in WeakIdent.

REFERENCES

[1] M. Tang, K. Hwang, and S. H. Kang, "Stemp: A fast and deterministic stem-graph approach for rna secondary structure prediction," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2023.

[2] M. Tang, M. Yashtini, and S. H. Kang, "Counting objects by diffused index: Geometry-free and training-free approach," *Journal of Visual Communication and Image Representation*, vol. 86, p. 103 527, 2022.

[3] M. Tang, W. Liao, R. Kuske, and S. H. Kang, "Weakident: Weak formulation for identifying differential equation using narrow-fit and trimming," *Journal of Computational Physics*, p. 112 069, 2023.

[4] L. Fu, Y. Cao, J. Wu, Q. Peng, Q. Nie, and X. Xie, "Ufold: Fast and accurate rna secondary structure prediction with deep learning," *Nucleic acids research*, vol. 50, no. 3, e14–e14, 2022.

[5] H. K. Wayment-Steele *et al.*, "Rna secondary structure packages evaluated and improved by high-throughput experiments," *BioRxiv*, pp. 2020–05, 2021.

[6] S. Chourasiya and G. U. Rani, "Automatic red blood cell counting using watershed segmentation," *Hemoglobin*, vol. 14, p. 17, 2014.

[7] B. Venkatalakshmi and K. Thilagavathi, "Automatic red blood cell counting using hough transform," in *2013 IEEE Conference on Information & Communication Technologies*, IEEE, 2013, pp. 267–271.

[8] M. Fritz, B. Leibe, B. Caputo, and B. Schiele, "Integrating representative and discriminant models for object category detection," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, IEEE, vol. 2, 2005, pp. 1363–1370.

[9] C. G. Loukas, G. D. Wilson, B. Vojnovic, and A. Linney, "An image analysis-based approach for automated counting of cancer cell nuclei in tissue sections," *Cytometry Part A: the journal of the International Society for Analytical Cytology*, vol. 55, no. 1, pp. 30–42, 2003.

[10] H. Idrees, I. Saleemi, C. Seibert, and M. Shah, "Multi-source multi-scale counting in extremely dense crowd images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2547–2554.

[11] D. Onoro-Rubio and R. J. López-Sastre, "Towards perspective-free object counting with deep learning," in *European conference on computer vision*, Springer, 2016, pp. 615–629.

[12] M. Marsden, K. McGuinness, S. Little, C. E. Keogh, and N. E. O'Connor, "People, penguins and petri dishes: Adapting object counting models to new visual domains and object types without forgetting," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8070–8079.

[13] W. Xie, J. A. Noble, and A. Zisserman, "Microscopy cell counting and detection with fully convolutional regression networks," *Computer methods in biomechanics and biomedical engineering: Imaging & Visualization*, vol. 6, no. 3, pp. 283–292, 2018.

[14] Y. Wang and Y. Zou, "Fast visual object counting via example-based density estimation," in *2016 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2016, pp. 3653–3657.

[15] L. H. Favela, "The dynamical renaissance in neuroscience," *Synthese*, vol. 199, no. 1, pp. 2103–2127, 2021.

[16] J. Bongard and H. Lipson, "Automated reverse engineering of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences*, vol. 104, no. 24, pp. 9943–9948, 2007.

[17] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *science*, vol. 324, no. 5923, pp. 81–85, 2009.

[18] J. T. Nardini *et al.*, "Learning equations from biological data with limited time samples," *Bulletin of mathematical biology*, vol. 82, no. 9, pp. 1–33, 2020.

[19] E. Baake, M. Baake, H. Bock, and K. Briggs, "Fitting ordinary differential equations to chaotic data," *Physical Review A*, vol. 45, no. 8, p. 5524, 1992.

[20] M. Bär, R. Hegger, and H. Kantz, "Fitting partial differential equations to space-time dynamics," *Physical Review E*, vol. 59, no. 1, p. 337, 1999.

[21] T. G. Müller and J. Timmer, "Parameter identification techniques for partial differential equations," *International Journal of Bifurcation and Chaos*, vol. 14, no. 06, pp. 2053–2060, 2004.

[22] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the national academy of sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

[23] L. Zhang and H. Schaeffer, "On the convergence of the sindy algorithm," *Multiscale Model. Simul.*, vol. 17, pp. 948–972, 2019.

[24] K. Kaheman, J. N. Kutz, and S. L. Brunton, "Sindy-pi: A robust algorithm for parallel implicit sparse identification of nonlinear dynamics," *Proceedings. Mathematical, Physical, and Engineering Sciences*, vol. 476, 2020.

[25] S. H. Rudy, A. Alla, S. L. Brunton, and J. N. Kutz, "Data-driven identification of parametric partial differential equations," *SIAM J. Appl. Dyn. Syst.*, vol. 18, pp. 643–660, 2019.

[26] J.-C. Loiseau, B. R. Noack, and S. L. Brunton, "Sparse reduced-order modelling: Sensor-based dynamics to full-state estimation," *Journal of Fluid Mechanics*, vol. 844, pp. 459–490, 2018.

[27] Y. Guan, S. L. Brunton, and I. V. Novosselov, "Sparse nonlinear models of chaotic electroconvection," *Royal Society Open Science*, vol. 8, 2021.

[28] K. P. Champion, S. L. Brunton, and J. N. Kutz, "Discovery of nonlinear multiscale systems: Sampling strategies and embeddings," *SIAM Journal on Applied Dynamical Systems*, vol. 18, no. 1, pp. 312–333, 2019.

[29] S. H. Kang, W. Liao, and Y. Liu, "Ident: Identifying differential equations with numerical time evolution," *Journal of Scientific Computing*, vol. 87, no. 1, pp. 1–27, 2021.

[30] Y. He, S. H. Kang, W. Liao, H. Liu, and Y. Liu, "Robust pde identification from noisy data," *arXiv preprint arXiv:2006.06557*, 2020.

[31] G. Tran and R. Ward, "Exact recovery of chaotic systems from highly corrupted data," *Multiscale Modeling & Simulation*, vol. 15, no. 3, pp. 1108–1129, 2017.

[32] H. Schaeffer, G. Tran, R. Ward, and L. Zhang, "Extracting structured dynamical systems using sparse optimization with very few samples," *Multiscale Modeling & Simulation*, vol. 18, no. 4, pp. 1435–1461, 2020.

[33] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Data-driven discovery of partial differential equations," *Science Advances*, vol. 3, no. 4, e1602614, 2017.

[34] K. Wu and D. Xiu, "Numerical aspects for approximating governing equations using data," *Journal of Computational Physics*, vol. 384, pp. 200–221, 2019.

[35] D. Gurevich, P. A. K. Reinbold, and R. O. Grigoriev, "Robust and optimal sparse regression for nonlinear pde models.," *Chaos*, vol. 29 10, p. 103 113, 2019.

[36] D. A. Messenger and D. M. Bortz, "Weak sindy for partial differential equations," *Journal of Computational Physics*, p. 110 525, 2021.

[37] D. A. Messenger and D. M. Bortz, "Weak sindy for partial differential equations," *Journal of Computational Physics*, vol. 443, p. 110 525, 2021.

[38] P. A. Reinbold, D. R. Gurevich, and R. O. Grigoriev, "Using noisy or incomplete data to discover models of spatiotemporal dynamics," *Physical Review E*, vol. 101, no. 1, p. 010 203, 2020.

[39] D. H. Mathews and D. H. Turner, "Prediction of rna secondary structure by free energy minimization," *Current opinion in structural biology*, vol. 16, no. 3, pp. 270–278, 2006.

[40] R. Lorenz, M. T. Wolfinger, A. Tanzer, and I. L. Hofacker, "Predicting RNA secondary structures from sequence and probing data," *Methods*, vol. 103, pp. 86–98, 2016.

[41] M. Zuker and P. Stiegler, "Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information.," *Nucleic acids research.*, 1981.

[42] M. Zuker, "Mfold web server for nucleic acid folding and hybridization prediction," *Nucleic acids research*, vol. 31, no. 13, pp. 3406–3415, 2003.

[43] M. N. Osman, R. Abdullah, and N. AbdulRashid, "Rna secondary structure prediction using dynamic programming algorithm—a review and proposed work," in *2010 International Symposium on Information Technology*, IEEE, vol. 2, 2010, pp. 551–556.

[44] Y. Frid and D. Gusfield, "An improved four-russians method and sparsified four-russians algorithm for rna folding," *Algorithms for Molecular Biology : AMB*, vol. 11, 2016.

[45] L. Huang *et al.*, "Linearfold: Linear-time approximate rna folding by 5'-to-3' dynamic programming and beam search," *Bioinformatics*, vol. 35, pp. i295–i304, 2019.

[46] S. Bellaousov and D. H. Mathews, "Probknot: Fast prediction of RNA secondary structure including pseudoknots," *Rna*, vol. 16, no. 10, pp. 1870–1880, 2010.

[47] Y. Wu *et al.*, "Improved prediction of RNA secondary structure by integrating the free energy model with restraints derived from experimental probing data," *Nucleic acids research*, vol. 43, no. 15, pp. 7247–7259, 2015.

[48] R. Lorenz, D. Luntzer, I. L. Hofacker, P. F. Stadler, and M. T. Wolfinger, "Shape directed RNA folding," *Bioinformatics*, vol. 32, no. 1, pp. 145–147, 2016.

[49] L. Zhang, H. Zhang, D. H. Mathews, and L. Huang, "Threshknot: Thresholded probknot for improved rna secondary structure prediction," *arXiv: Biomolecules*, 2019.

[50] H. Zhang, L. Zhang, D. H. Mathews, and L. Huang, "Linearpartition: Linear-time approximation of rna folding partition function and base-pairing probabilities," *Bioinformatics*, vol. 36, pp. i258–i267, 2020.

[51] S. Zakov, Y. Goldberg, M. Elhadad, and M. Ziv-Ukelson, "Rich parameterization improves RNA structure prediction," *Journal of Computational Biology*, vol. 18, no. 11, pp. 1525–1542, 2011.

[52] E. Bindewald, T. Kluth, and B. A. Shapiro, "Cylofold: Secondary structure prediction including pseudoknots," *Nucleic acids research*, vol. 38, no. suppl_2, W368–W372, 2010.

[53] O. Perriquet, H. Touzet, and M. Dauchet, "Finding the common structure shared by two homologous RNAs," *Bioinformatics*, vol. 19, no. 1, pp. 108–116, 2003.

[54] M. S. Swenson *et al.*, "Gtfold: Enabling parallel RNA secondary structure prediction on multi-core desktops," *BMC research notes*, vol. 5, no. 1, pp. 1–6, 2012.

[55] K. Sato, Y. Kato, M. Hamada, T. Akutsu, and K. Asai, "IPknot: Fast and accurate prediction of RNA secondary structures with pseudoknots using integer programming," *Bioinformatics*, vol. 27, no. 13, pp. i85–i93, 2011.

[56] H. H. Tsang and K. C. Wiese, "SARNA-predict: Accuracy improvement of RNA secondary structure prediction using permutation-based simulated annealing," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 7, no. 4, pp. 727–740, 2008.

[57] https://en.wikipedia.org/wiki/List_of_RNA_structure_prediction_software, [Online; accessed 11-November-2021].

[58] M. Hamada, K. Sato, and K. Asai, "Improving the accuracy of predicting secondary structure for aligned RNA sequences," *Nucleic acids research*, vol. 39, no. 2, pp. 393–402, 2011.

[59] M. Hamada, K. Sato, H. Kiryu, T. Mituyama, and K. Asai, "Centroidalign: Fast and accurate aligner for structured RNAs by maximizing expected sum-of-pairs score," *Bioinformatics*, vol. 25, no. 24, pp. 3236–3243, 2009.

[60] K. Sato, Y. Kato, T. Akutsu, K. Asai, and Y. Sakakibara, "DAFS: Simultaneous aligning and folding of RNA sequences via dual decomposition," *Bioinformatics*, vol. 28, no. 24, pp. 3218–3224, 2012.

[61] T. Zhang, J. Singh, T. Litfin, J. Zhan, K. K. Paliwal, and Y. Zhou, "Rnacmap: A fully automatic pipeline for predicting contact maps of rnas by evolutionary coupling analysis," *Bioinformatics*, 2021.

[62] Z. Li and Y. Yu, "Protein secondary structure prediction using cascaded convolutional and recurrent neural networks," *arXiv preprint arXiv:1604.07176*, 2016.

[63] J. Singh, J. Hanson, K. Paliwal, and Y. Zhou, "Rna secondary structure prediction using an ensemble of two-dimensional deep neural networks and transfer learning," *Nature communications*, vol. 10, no. 1, pp. 1–13, 2019.

[64] S. Zhang *et al.*, "A deep learning framework for modeling structural features of rna-binding protein targets," *Nucleic acids research*, vol. 44, no. 4, e32–e32, 2016.

[65] Y. Ji, X. Xu, and G. D. Stormo, "A graph theoretical approach for predicting common rnasecondary structure motifs including pseudoknots in unaligned sequences," *Bioinformatics*, vol. 20, no. 10, pp. 1591–1602, 2004.

[66] D. Knisley, J. Knisley, C. Ross, and A. Rockney, "Classifying multigraph models of secondary rna structure using graph-theoretic descriptors," *International Scholarly Research Notices*, vol. 2012, 2012.

[67] A. Legendre, E. Angel, and F. Tahi, "Rcpred: Rna complex prediction as a constrained maximum weight clique problem," *BMC bioinformatics*, vol. 20, no. 3, pp. 53–62, 2019.

[68] D. Fera *et al.*, "Rag: Rna-as-graphs web resource," *BMC bioinformatics*, vol. 5, no. 1, p. 88, 2004.

[69] M. Hamada, K. Tsuda, T. Kudo, T. Kin, and K. Asai, "Mining frequent stem patterns from unaligned rnasequences," *Bioinformatics*, vol. 22, no. 20, pp. 2480–2487, 2006.

[70] C. Gaspin and E. Westhof, "An interactive framework for rnasecondary structure prediction with a dynamical treatment of constraints," *Journal of molecular biology*, vol. 254, no. 2, pp. 163–174, 1995.

[71] E. Tomita, A. Tanaka, and H. Takahashi, "The worst-case time complexity for generating all maximal cliques and computational experiments," *Theoretical Computer Science*, vol. 363, no. 1, pp. 28–42, 2006.

[72] C. Bron and J. Kerbosch, "Algorithm 457: Finding all cliques of an undirected graph," *Commun. ACM*, vol. 16, no. 9, pp. 575–577, Sep. 1973.

[73] D. Eppstein, M. Löffler, and D. Strash, "Listing all maximal cliques in sparse graphs in near-optimal time," in *Algorithms and Computation: 21st International Symposium, ISAAC 2010, Jeju Island, Korea, December 15-17, 2010, Proceedings, Part I*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 403–414, ISBN: 978-3-642-17517-6.

[74] N. K. Tanner, O. Cordin, J. Banroques, M. Doere, and P. Linder, "The Q motif: A newly identified motif in DEAD box helicases may regulate atp binding and hydrolysis," *Molecular cell*, vol. 11, no. 1, pp. 127–138, 2003.

[75] O. Cordin, N. K. Tanner, M. Doere, P. Linder, and J. Banroques, "The newly discovered Q motif of DEAD-box RNA helicases regulates RNA-binding and helicase activity," *The EMBO journal*, vol. 23, no. 13, pp. 2478–2487, 2004.

[76] M. Parisien and F. Major, "The MC-Fold and MC-Sym pipeline infers RNA structure from sequence data," *Nature*, vol. 452, no. 7183, pp. 51–55, 2008.

[77] F. C. Bernstein *et al.*, "The protein data bank: A computer-based archival file for macromolecular structures," *Journal of molecular biology*, vol. 112, no. 3, pp. 535–542, 1977.

[78] R. Müller and M. E. Nebel, "Combinatorics of rnasecondary structures with base triples," *Journal of Computational Biology*, vol. 22, no. 7, pp. 619–648, 2015.

[79] E. De Leonardis *et al.*, "Direct-coupling analysis of nucleotide coevolution facilitates RNA secondary and tertiary structure prediction," *Nucleic acids research*, vol. 43, no. 21, pp. 10 444–10 455, 2015.

[80] A. Micsonai *et al.*, "Bestsel: A web server for accurate protein secondary structure prediction and fold recognition from the circular dichroism spectra," *Nucleic acids research*, vol. 46, no. W1, W315–W322, 2018.

[81] J. Yang and Y. Zhang, "I-tasser server: New development for protein structure and function predictions," *Nucleic acids research*, vol. 43, no. W1, W174–W181, 2015.

[82] H. M. Berman *et al.*, "The nucleic acid database. a comprehensive relational database of three-dimensional structures of nucleic acids.," *Biophysical journal*, vol. 63, no. 3, p. 751, 1992.

[83] B. Coimbatore Narayanan *et al.*, "The nucleic acid database: New features and capabilities," *Nucleic acids research*, vol. 42, no. D1, pp. D114–D122, 2013.

[84] D. H. Mathews, M. D. Disney, J. L. Childs, S. J. Schroeder, M. Zuker, and D. H. Turner, "Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure," *Proceedings of the National Academy of Sciences*, vol. 101, no. 19, pp. 7287–7292, 2004.

[85] Z. J. Lu, J. W. Gloor, and D. H. Mathews, "Improved RNA secondary structure prediction by maximizing expected pair accuracy," *Rna*, vol. 15, no. 10, pp. 1805–1813, 2009.

[86] M. A. Jonikas *et al.*, "Coarse-grained modeling of large RNA molecules with knowledge-based potentials and structural filters," *Rna*, vol. 15, no. 2, pp. 189–199, 2009.

[87] J. S. Reuter and D. H. Mathews, "RNAstructure: Software for RNA secondary structure prediction and analysis," *BMC bioinformatics*, vol. 11, no. 1, pp. 1–9, 2010.

[88] C. Laing and T. Schlick, "Computational approaches to 3D modeling of RNA," *Journal of Physics: Condensed Matter*, vol. 22, no. 28, p. 283 101, 2010.

[89] K. Darty, A. Denise, and Y. Ponty, "VARNA: Interactive drawing and editing of the RNA secondary structure," *Bioinformatics*, vol. 25, no. 15, p. 1974, 2009.

[90] P. C. Bevilacqua, L. E. Ritchey, Z. Su, and S. M. Assmann, "Genome-wide analysis of RNA secondary structure," *Annual review of genetics*, vol. 50, pp. 235–266, 2016.

[91] I. Tinoco Jr and C. Bustamante, "How RNA folds," *Journal of molecular biology*, vol. 293, no. 2, pp. 271–281, 1999.

[92] S. Washietl, I. L. Hofacker, and P. F. Stadler, "Fast and reliable prediction of noncoding RNAs," *Proceedings of the National Academy of Sciences*, vol. 102, no. 7, pp. 2454–2459, 2005.

[93] J. Ren, B. Rastegari, A. Condon, and H. H. Hoos, "Hotknots: Heuristic prediction of RNA secondary structures including pseudoknots," *Rna*, vol. 11, no. 10, pp. 1494–1504, 2005.

[94] Z. Tan, Y. Fu, G. Sharma, and D. H. Mathews, "TurboFold II: RNA structural alignment and secondary structure prediction informed by multiple homologs," *Nucleic acids research*, vol. 45, no. 20, pp. 11 570–11 581, 2017.

[95] J. J. Cannone *et al.*, "The comparative RNA web (CRW) site: An online database of comparative sequence and structure information for ribosomal, intron, and other RNAs," *BMC bioinformatics*, vol. 3, no. 1, p. 2, 2002.

[96] S. Poznanović, F. Barrera-Cruz, A. Kirkpatrick, M. Ielusic, and C. Heitsch, "The challenge of RNA branching prediction: A parametric analysis of multiloop initiation under thermodynamic optimization," *Journal of structural biology*, vol. 210, no. 1, p. 107 475, 2020.

[97] M. Szymanski, M. Z. Barciszewska, V. A. Erdmann, and J. Barciszewski, "5S ribosomal RNA database," *Nucleic Acids Research*, vol. 30, no. 1, pp. 176–178, 2002.

[98] N. B. Leontis, J. Stombaugh, and E. Westhof, "Motif prediction in ribosomal RNAs lessons and prospects for automated motif prediction in homologous RNA molecules," *Biochimie*, vol. 84, no. 9, pp. 961–973, 2002.

[99] A. O. Harmanci, G. Sharma, and D. H. Mathews, "Turbofold: Iterative probabilistic estimation of secondary structures for multiple RNA sequences," *BMC bioinformatics*, vol. 12, no. 1, pp. 1–22, 2011.

[100] Z. Tan, G. Sharma, and D. H. Mathews, "Modeling rna secondary structure with sequence comparison and experimental mapping data.," *Biophysical journal*, vol. 113 2, pp. 330–338, 2017.

[101] K. Wiese, A. Deschenes, and A. Hendriks, "Rnapredict—an evolutionary algorithm for rna secondary structure prediction," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 5, no. 1, pp. 25–41, 2008.

[102] S. R. Eddy, "How do RNA folding algorithms work?" *Nature biotechnology*, vol. 22, no. 11, pp. 1457–1458, 2004.

[103] S. Lalwani, R. Kumar, and N. Gupta, "An efficient two-level swarm intelligence approach for RNA secondary structure prediction with bi-objective minimum free energy scores," *Swarm and Evolutionary Computation*, vol. 27, pp. 68–79, 2016.

[104] R. Lorenz *et al.*, "Viennarna package 2.0," *Algorithms for molecular biology*, vol. 6, no. 1, pp. 1–14, 2011.

[105] S. Srikamdee, W. Wattanapornprom, and P. Chongstitvatana, "Rna secondary structure prediction with coincidence algorithm," in *2016 16th International Symposium on Communications and Information Technologies (ISCIT)*, IEEE, 2016, pp. 686–690.

[106] I. L. Hofacker, S. H. Bernhart, and P. F. Stadler, "Alignment of RNA base pairing probability matrices," *Bioinformatics*, vol. 20, no. 14, pp. 2222–2227, 2004.

[107] I. L. Hofacker, M. Fekete, and P. F. Stadler, "Secondary structure prediction for aligned RNA sequences," *Journal of molecular biology*, vol. 319, no. 5, pp. 1059–1066, 2002.

[108] A. B. Bellamy-Royds and M. Turcotte, "Can clustal-style progressive pairwise alignment of multiple sequences be used in RNA secondary structure prediction?" *BMC bioinformatics*, vol. 8, no. 1, pp. 1–19, 2007.

[109] K. Sato, M. Akiyama, and Y. Sakakibara, "Rna secondary structure prediction using deep learning with thermodynamic integration," *Nature communications*, vol. 12, no. 1, pp. 1–9, 2021.

[110] P. Danaee, M. Rouches, M. Wiley, D. Deng, L. Huang, and D. Hendrix, "Bprna: Large-scale automated annotation and analysis of rna secondary structure," *Nucleic acids research*, vol. 46, no. 11, pp. 5381–5394, 2018.

[111] E. Rivas, R. Lang, and S. R. Eddy, "A range of complex probabilistic models for rna secondary structure prediction that includes the nearest-neighbor model and more," *RNA*, vol. 18, no. 2, pp. 193–212, 2012.

[112] M. Andronescu, R. Aguirre-Hernandez, A. Condon, and H. H. Hoos, "Rnasoft: A suite of rna secondary structure prediction and design software tools," *Nucleic acids research*, vol. 31, no. 13, pp. 3416–3422, 2003.

[113] X. Chen, Y. Li, R. Umarov, X. Gao, and L. Song, "Rna secondary structure prediction by learning unrolled algorithms," *arXiv preprint arXiv:2002.05810*, 2020.

[114] E. Rivas and S. R. Eddy, "A dynamic programming algorithm for rna structure prediction including pseudoknots," *Journal of molecular biology*, vol. 285, no. 5, pp. 2053–2068, 1999.

[115] S. H. Kang and R. March, "Variational models for image colorization via chromaticity and brightness decomposition," *IEEE Transactions on Image Processing*, vol. 16, no. 9, pp. 2251–2261, 2007.

[116] M. Yashtini, S. H. Kang, and W. Zhu, "Efficient alternating minimization methods for variational edge-weighted colorization models," *Advances in Computational Mathematics*, vol. 45, no. 3, pp. 1735–1767, 2019.

[117] M. Yashtini and S. H. Kang, "A fast relaxed normal two split method and an effective weighted tv approach for euler's elastica image inpainting," *SIAM Journal on Imaging Sciences*, vol. 9, no. 4, pp. 1552–1581, 2016.

[118] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[119] Y. Chen, W. W. Hager, M. Yashtini, X. Ye, and H. Zhang, "Bregman operator splitting with variable stepsize for total variation image reconstruction," *Computational optimization and applications*, vol. 54, no. 2, pp. 317–342, 2013.

[120] W. Hager, C. Ngo, M. Yashtini, and H.-C. Zhang, "An alternating direction approximate newton algorithm for ill-conditioned inverse problems with application to parallel mri," *Journal of the Operations Research Society of China*, vol. 3, no. 2, pp. 139–162, 2015.

[121] M. R. Hestenes, "Multiplier and gradient methods," *Journal of optimization theory and applications*, vol. 4, no. 5, pp. 303–320, 1969.

[122] C. Wu and X.-C. Tai, "Augmented lagrangian method, dual methods, and split bregman iteration for rof, vectorial tv, and high order models," *SIAM Journal on Imaging Sciences*, vol. 3, no. 3, pp. 300–339, 2010.

[123] J. Yang, Y. Zhang, and W. Yin, "A fast alternating direction method for tvl1-l2 signal reconstruction from partial fourier data," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 288–297, 2010.

[124] M. Yashtini and S. H. Kang, "Alternating direction method of multiplier for euler's elastica-based denoising," in *International Conference on Scale Space and Variational Methods in Computer Vision*, Springer, 2015, pp. 690–701.

[125] Y. Chen, K. Biddell, A. Sun, P. A. Relue, and J. D. Johnson, "An automatic cell counting method for optical images," in *Proceedings of the First Joint BMES/EMBS Conference. 1999 IEEE Engineering in Medicine and Biology 21st Annual Conference and the 1999 Annual Fall Meeting of the Biomedical Engineering Society (Cat. N*, IEEE, vol. 2, 1999, 819–vol.

[126] M. Baygin, M. Karakose, A. Sarimaden, and E. Akin, "An image processing based object counting approach for machine vision application," *arXiv preprint arXiv:1802.05911*, 2018.

[127] E. Lu, W. Xie, and A. Zisserman, "Class-agnostic counting," in *Asian conference on computer vision*, Springer, 2018, pp. 669–684.

[128] I. H. Laradji, N. Rostamzadeh, P. O. Pinheiro, D. Vazquez, and M. Schmidt, "Where are the blobs: Counting by localization with point supervision," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 547–562.

[129] T. W. Ayalew, J. R. Ubbens, and I. Stavness., "Unsupervised domain adaptation for plant organ counting," in *Computer Vision – ECCV 2020 Workshops*, A. Bartoli and A. Fusiello, Eds., Cham: Springer International Publishing, 2020, pp. 330–346, ISBN: 978-3-030-65414-6.

[130] M. V. Giuffrida, A. Dobrescu, P. Doerner, and S. A. Tsaftaris, "Leaf counting without annotations using adversarial unsupervised domain adaptation," in *2019*

IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE, 2019, pp. 2590–2599.

[131] A. K. Nellithimaru and G. A. Kantor, "Rols: Robust object-level slam for grape counting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[132] Q. Wang, J. Gao, W. Lin, and X. Li, "Nwpu-crowd: A large-scale benchmark for crowd counting and localization," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 6, pp. 2141–2149, 2020.

[133] W. Liu, M. Salzmann, and P. Fua, "Context-aware crowd counting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5099–5108.

[134] H. Tulsani, S. Saxena, and N. Yadav, "Segmentation using morphological watershed transformation for counting blood cells," *IJCAIT*, vol. 2, no. 3, pp. 28–36, 2013.

[135] X. Guo and F. Yu, "A method of automatic cell counting based on microscopic image," in *2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics*, IEEE, vol. 1, 2013, pp. 293–296.

[136] M. Maitra, R. K. Gupta, and M. Mukherjee, "Detection and counting of red blood cells in blood cell images using hough transform," *International journal of computer applications*, vol. 53, no. 16, 2012.

[137] H. Berge, D. Taylor, S. Krishnan, and T. S. Douglas, "Improved red blood cell counting in thin blood smears," in *2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, IEEE, 2011, pp. 204–207.

[138] S. Kothari, Q. Chaudry, and M. D. Wang, "Automated cell counting and cluster segmentation using concavity detection and ellipse fitting techniques," in *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, IEEE, 2009, pp. 795–798.

[139] J. Gall and V. Lempitsky, "Class-specific hough forests for object detection," in *Decision forests for computer vision and medical image analysis*, Springer, 2013, pp. 143–157.

[140] S. Maji and J. Malik, "Object detection using a max-margin hough transform," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2009, pp. 1038–1045.

[141] O. Barinova, V. Lempitsky, and P. Kholi, "On detection of multiple object instances using hough transforms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1773–1784, 2012.

[142] C. Chen, S. Li, H. Qin, and A. Hao, "Structure-sensitive saliency detection via multilevel rank analysis in intrinsic feature space," *IEEE Transactions on Image Processing*, vol. 24, no. 8, pp. 2303–2316, 2015.

[143] C. Chen, S. Li, Y. Wang, H. Qin, and A. Hao, "Video saliency detection via spatial-temporal fusion and low-rank coherency diffusion," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3156–3170, 2017.

[144] T. Falk *et al.*, "U-net: Deep learning for cell counting, detection, and morphometry," *Nature methods*, vol. 16, no. 1, pp. 67–70, 2019.

[145] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[146] R. M. Rad, P. Saeedi, J. Au, and J. Havelock, "Cell-net: Embryonic cell counting and centroid localization via residual incremental atrous pyramid and progressive upsampling convolution," *IEEE Access*, vol. 7, pp. 81 945–81 955, 2019.

[147] S. He, K. T. Minn, L. Solnica-Krezel, M. A. Anastasio, and H. Li, "Deeply-supervised density regression for automatic cell counting in microscopy images," *Medical Image Analysis*, vol. 68, p. 101 892, 2021.

[148] J. Paul Cohen, G. Boucher, C. A. Glastonbury, H. Z. Lo, and Y. Bengio, "Count-ception: Counting by fully convolutional redundant counting," in *Proceedings of the IEEE International conference on computer vision workshops*, 2017, pp. 18–26.

[149] N. Jiang and F. Yu, "A two-path network for cell counting," *IEEE Access*, vol. 9, pp. 70 806–70 815, 2021.

[150] N. Jiang and F. Yu, "A cell counting framework based on random forest and density map," *Applied Sciences*, vol. 10, no. 23, p. 8346, 2020.

[151] N. Jiang and F. Yu, "A foreground mask network for cell counting," in *2020 IEEE 5th International Conference on Image, Vision and Computing (ICIVC)*, IEEE, 2020, pp. 128–132.

[152] Y. Guo, O. Krupa, J. Stein, G. Wu, and A. Krishnamurthy, "Sau-net: A unified network for cell counting in 2d and 3d microscopy images," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2021.

[153] J. P. Cohen, G. Boucher, C. A. Glastonbury, H. Z. Lo, and Y. Bengio, "Count-ception: Counting by fully convolutional redundant counting," in *International Conference on Computer Vision Workshop on BioImage Computing*, 2017.

[154] X. Ding, Q. Zhang, and W. J. Welch, "Classification beats regression: Counting of cells from greyscale microscopic images based on annotation-free training samples," in *CAAI International Conference on Artificial Intelligence*, Springer, 2021, pp. 662–673.

[155] J. Rodriguez-Vazquez, A. Alvarez-Fernandez, M. Molina, and P. Campoy, "Zenithal isotropic object counting by localization using adversarial training," *Neural Networks*, vol. 145, pp. 155–163, 2022.

[156] M. Ester, H. P. Kriegel, J. Sander, and X. Xu., "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96, Portland, Oregon: AAAI Press, 1996, pp. 226–231.

[157] S. H. Kang, B. Sandberg, and A. M. Yip, "A regularized k-means and multiphase scale segmentation," *Inverse Problems & Imaging*, vol. 5, no. 2, p. 407, 2011.

[158] V. Lempitsky and A. Zisserman, "Learning to count objects in images," *Advances in neural information processing systems*, vol. 23, pp. 1324–1332, 2010.

[159] Y. Xie, F. Xing, X. Shi, X. Kong, H. Su, and L. Yang, "Efficient and robust cell detection: A structured regression approach," *Medical image analysis*, vol. 44, pp. 245–254, 2018.

[160] C. Arteta, V. Lempitsky, and A. Zisserman, "Counting in the wild," in *European conference on computer vision*, Springer, 2016, pp. 483–498.

[161] L. Fiaschi, U. Köthe, R. Nair, and F. A. Hamprecht, "Learning to count with regression forest and structured labels," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, IEEE, 2012, pp. 2685–2688.

[162] P. Kainz, M. Urschler, S. Schulter, P. Wohlhart, and V. Lepetit, "You should use regression to detect cells," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, 2015, pp. 276–283.

[163] J. Lonsdale *et al.*, "The genotype-tissue expression (gtex) project," *Nature genetics*, vol. 45, no. 6, pp. 580–585, 2013.

[164] C. Arteta, V. Lempitsky, J. A. Noble, and A. Zisserman, "Learning to detect cells using non-overlapping extremal regions," in *International conference on medical image computing and computer-assisted intervention*, Springer, 2012, pp. 348–356.

[165] Y. Al-Kofahi, W. Lassoued, W. Lee, and B. Roysam, "Improved automatic detection and segmentation of cell nuclei in histopathology images," *IEEE Transactions on Biomedical Engineering*, vol. 57, no. 4, pp. 841–852, 2009.

[166] J. Byun, M. R. Verardo, B. Sumengen, G. P. Lewis, B. Manjunath, and S. K. Fisher, "Automated tool for the detection of cell nuclei in digital microscopic images: Application to retinal images," *Mol Vis*, vol. 12, no. 105-07, pp. 949–60, 2006.

[167] B. Parvin, Q. Yang, J. Han, H. Chang, B. Rydberg, and M. H. Barcellos-Hoff, "Iterative voting for inference of structural saliency and characterization of subcellular events," *IEEE Transactions on Image Processing*, vol. 16, no. 3, pp. 615–623, 2007.

[168] Y. Xie, F. Xing, X. Kong, H. Su, and L. Yang, "Beyond classification: Structured regression for robust cell detection using convolutional neural network," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, 2015, pp. 358–365.

[169] C. Arteta, V. Lempitsky, J. A. Noble, and A. Zisserman, "Detecting overlapping instances in microscopy images using extremal region trees," *Medical image analysis*, vol. 27, pp. 3–16, 2016.

[170] K. Zuiderveld, "Contrast limited adaptive histogram equalization," *Graphics gems*, pp. 474–485, 1994.

[171] J. A. Cruz *et al.*, "Multi-modality imagery database for plant phenotyping," *Machine Vision and Applications*, vol. 27, no. 5, pp. 735–749, 2016.

[172] P. Pawara, A. Boshchenko, L. R. Schomaker, and M. A. Wiering, "Deep learning with data augmentation for fruit counting," in *International Conference on Artificial Intelligence and Soft Computing*, Springer, 2020, pp. 203–214.

[173] M. Rahnemoonfar and C. Sheppard, "Deep count: Fruit counting based on deep simulated learning," *Sensors*, vol. 17, no. 4, p. 905, 2017.

[174] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[175] O. R. Vincent, O. Folorunso, *et al.*, "A descriptive algorithm for sobel image edge detection," in *Proceedings of informing science & IT education conference (InSITE)*, vol. 40, 2009, pp. 97–107.

[176] G. Shrivakshan and C. Chandrasekar, "A comparison of various edge detection techniques used in image processing," *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 5, p. 269, 2012.

[177] Y.-J. Cha, K. You, and W. Choi, "Vision-based detection of loosened bolts using the hough transform and support vector machines," *Automation in Construction*, vol. 71, pp. 181–188, 2016.

[178] L. Baker, S. Mills, T. Langlotz, and C. Rathbone, "Power line detection using hough transform and line tracing techniques," in *2016 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, IEEE, 2016, pp. 1–6.

[179] J. Ding, N. Xue, Y. Long, G.-S. Xia, and Q. Lu, "Learning roi transformer for detecting oriented objects in aerial images," *arXiv preprint arXiv:1812.00155*, 2018.

[180] G.-S. Xia *et al.*, "Dota: A large-scale dataset for object detection in aerial images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3974–3983.

[181] C. Arteta, V. Lempitsky, and A. Zisserman, "Counting in the wild," in *European conference on computer vision*, Springer, 2016, pp. 483–498.

[182] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE transactions on Information Theory*, vol. 55, no. 5, pp. 2230–2249, 2009.

[183] H. G. Bock, "Recent advances in parameteridentification techniques for ode," *Numerical treatment of inverse problems in differential and integral equations*, pp. 95–121, 1983.

[184] H. Schaeffer, "Learning partial differential equations via data discovery and sparse optimization," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 473, no. 2197, p. 20 160 446, 2017.

[185] S. Zhang and G. Lin, "Robust data-driven discovery of governing physical laws with error bars," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 474, no. 2217, p. 20 180 305, 2018.

[186] Z. Chen, V. Churchill, K. Wu, and D. Xiu, "Deep neural network modeling of unknown partial differential equations in nodal space," *J. Comput. Phys.*, vol. 449, p. 110 782, 2022.

[187] T. Qin, K. Wu, and D. Xiu, "Data driven governing equations approximation using deep neural networks," *Journal of Computational Physics*, vol. 395, pp. 620–635, 2019.

[188] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature communications*, vol. 9, no. 1, pp. 1–10, 2018.

[189] M. Raissi and G. E. Karniadakis, "Hidden physics models: Machine learning of nonlinear partial differential equations," *Journal of Computational Physics*, vol. 357, pp. 125–141, 2018.

[190] K. Wu and D. Xiu, "Data-driven deep learning of partial differential equations in modal space," *J. Comput. Phys.*, vol. 408, p. 109 307, 2020.

[191] H. Xu, H. Chang, and D. Zhang, "Dl-pde: Deep-learning based data-driven discovery of partial differential equations from discrete and noisy data," *arXiv preprint arXiv:1908.04463*, 2019.

[192] H. Xu, H. Chang, and D. Zhang, "Dlga-pde: Discovery of pdes with incomplete candidate library via combination of deep learning and genetic algorithm," *J. Comput. Phys.*, vol. 418, p. 109 584, 2020.

[193] Y. He, S. H. Kang, W. Liao, H. Liu, and Y. Liu, "Numerical identification of nonlocal potential in aggregation," *Communications in Computational Physics*, 2022.

[194] S. Rudy, A. Alla, S. L. Brunton, and J. N. Kutz, "Data-driven identification of parametric partial differential equations," *SIAM Journal on Applied Dynamical Systems*, vol. 18, no. 2, pp. 643–660, 2019.

[195] Z. Chen, K. Wu, and D. Xiu, "Methods to recover unknown processes in partial differential equations using data," *ArXiv*, vol. abs/2003.02387, 2020.

[196] Å. Björck, "Least squares methods," *Handbook of numerical analysis*, vol. 1, pp. 465–652, 1990.

[197] Å. Björck, "Error analysis of least squares algorithms," in *Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms*, Springer, 1991, pp. 41–73.

[198] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Sparse identification of nonlinear dynamics with control (sindyc)," *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 710–715, 2016.

[199] M. van Berkel, G. Vandersteen, E. Geerardyn, R. Pintelon, H. Zwart, and M. de Baar, "Frequency domain sample maximum likelihood estimation for spatially dependent parameter estimation in pdes," *Automatica*, vol. 50, no. 8, pp. 2113–2119, 2014.

[200] J. Goos, J. Lataire, E. Louarroudi, and R. Pintelon, "Frequency domain weighted nonlinear least squares estimation of parameter-varying differential equations," *Automatica*, vol. 75, pp. 191–199, 2017.

[201] A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis, "Adaptive activation functions accelerate convergence in deep and physics-informed neural networks," *Journal of Computational Physics*, vol. 404, p. 109 136, 2020.

[202] Z. Zhang and Y. Liu, "A robust framework for identification of pdes from noisy data," *Journal of Computational Physics*, vol. 446, p. 110 657, 2021.

[203] H. Zhao and Y. Zhong, "How much can one learn from a single solution of a pde?" *arXiv preprint arXiv:2206.05336*, 2022.

[204] Y. He, S.-H. Kang, W. Liao, H. Liu, and Y. Liu, "Robust identification of differential equations by numerical techniques from a single set of noisy observation," *SIAM Journal on Scientific Computing*, vol. 44, no. 3, A1145–A1175, 2022.

[205] L. N. Trefethen, *Spectral methods in MATLAB*. SIAM, 2000.

[206] A.-K. Kassam and L. N. Trefethen, "Fourth-order time-stepping for stiff pdes," *SIAM Journal on Scientific Computing*, vol. 26, no. 4, pp. 1214–1233, 2005.

[207] I. L. Hofacker, "Vienna rna secondary structure server," *Nucleic acids research*, vol. 31, no. 13, pp. 3429–3431, 2003.