Multi-Library SINDy: A Computationally Efficient Framework for SINDy (Draft)

Hirsa Kia[1]

[1]Department of Computer and Information Sciences, Temple University, Philadelphia, PA

October 14, 2024

Abstract We present Multi-Library SINDy, an enhanced framework for modeling nonlinear dynamical systems. Traditional SINDy approaches use a single feature library, which can be computationally expensive and inaccurate when modeling systems with diverse dynamics. Our method addresses this by using multiple tailored libraries, improving complexity and computational efficiency. Through simulations, Multi-Library SINDy achieves improved accuracy and faster execution, proving effective across systems with varying dynamical behaviors. This approach offers a scalable solution for modeling complex systems.

1 Introduction

Dynamic systems serve as a powerful tool for modeling complex interactions and temporal behaviors in natural and engineered systems. These systems, characterized by nonlinear, multiscale dynamics, present significant challenges in balancing computational efficiency and accuracy [1, 2]. Traditional regression techniques struggle with this balance, often oversimplifying the model or leading to overfitting [3, 4].

The Sparse Identification of Nonlinear Dynamical Systems (SINDy) framework addresses these challenges by using sparse regression to capture the essential dynamics of complex systems, offering a more accurate representation than conventional methods [5, 6]. However, SINDy's reliance on a constant-size feature library limits its efficiency and accuracy, particularly when dealing with varying system scales.

To overcome these limitations, we propose Multi-Library SINDy, an extension that uses multiple polynomial libraries tailored to different system dynamics. This approach provides better control over model complexity, reducing computational load while improving adaptability to distinct behaviors. Our theoretical analysis and empirical results show that Multi-Library SINDy retains the strengths of classical SINDy in terms of sparsity and interpretability, while significantly enhancing computational efficiency and precision, making it suitable for applications in fields like climate science, economics, and biomedical engineering.

1.1 SINDy Algorithm

The Sparse Identification of Nonlinear Dynamical Systems (SINDy) algorithm [1] utilizes sparse regression to identify governing equations from data. The algorithm begins with a polynomial library of candidate functions, from which a sparse subset is selected to describe the system dynamics. The general form of SINDy with a polynomial library is as follows:

$$\begin{pmatrix} \dot{x}_1(t_0) & \cdots & \dot{x}_n(t_0) \\ \dot{x}_1(t_1) & \cdots & \dot{x}_n(t_1) \\ \vdots & \ddots & \vdots \\ \dot{x}_1(t_m) & \cdots & \dot{x}_n(t_m) \end{pmatrix} = \begin{pmatrix} 1 & a_{1,1} & \cdots & a_{1,n} \\ 1 & a_{2,1} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & a_{m,1} & \cdots & a_{m,n} \end{pmatrix} \begin{pmatrix} w_{1,1} & \cdots & w_{1,n} \\ w_{2,1} & \cdots & w_{2,n} \\ \vdots & \ddots & \vdots \\ w_{m,1} & \cdots & w_{m,n} \end{pmatrix} \tag{1}$$

Here, $a_{i,j}$ represents the i-th time sample of the j-th function in the library (e.g., x^2, $\sin(x)$), and Σ is the matrix of weights. This equation can also be written more compactly as:

$$\dot{X} = \Phi\Sigma \tag{2}$$

To model the dynamics of the i-th feature, the corresponding equation is:

$$\begin{pmatrix} \dot{x}_i(t_1) \\ \dot{x}_i(t_2) \\ \vdots \\ \dot{x}_i(t_m) \end{pmatrix} = \begin{pmatrix} 1 & a_{1,1} & \cdots & a_{1,n} \\ 1 & a_{2,1} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & a_{m,1} & \cdots & a_{m,n} \end{pmatrix} \begin{pmatrix} w_{1,i} \\ w_{2,i} \\ \vdots \\ w_{m,i} \end{pmatrix} \tag{3}$$

In the original SINDy framework, Sequentially Thresholded Least Squares (STLS) was used to iteratively select the sparse terms from the polynomial library. The STLS algorithm operates in iterations, which can be described as follows [1]:

$$\begin{aligned} S^k &= \left\{1 \le j \le n : \left|x_j^k\right| \ge \lambda\right\} \\ x^{k+1} &= \underset{x \in \mathbb{R}^n : \mathrm{supp}(x) \subseteq S^k}{\operatorname{argmin}} \|Ax - b\|_2 \end{aligned} \tag{4}$$

The STLS algorithm iteratively thresholds the coefficients, eliminating small terms below a threshold λ, and re-solves the least squares problem on the remaining terms. This ensures sparsity in the model and computational efficiency by reducing the number of terms considered.

The pseudocode of the STLS algorithm is presented in Algorithm 1:

Algorithm 1 Sequentially Thresholded Least Squares Algorithm

Inputs:
λ :=Threshold, m :=number of parameters, d := number of parameters, ϕ :=Library function matrix, ϵ := Loop control parameter, $\dot{x}$
Algorithm
$\Sigma = Zeroes[m, 1]$
for $i = 1 : d + 1$ **do** ▷ Least Squares loop
  $\Sigma \leftarrow (\phi^\top \phi)^{-1} \phi \dot{x}$
  for $j = 1 : d + 1$ **do** ▷ Thresholding loop
    if $\Sigma[j] \leq \lambda$ **then** $\Sigma[j] \leftarrow 0$, $\phi[:, j] \leftarrow \mathbf{0}$
    end if
    if $\max_k |\Sigma[k]^i - \Sigma[k]^{i-1}| \leq \epsilon$ **then** Break ▷ Convergence check loop
    end if
  end for
end for

Multi-Library SINDy extends the original approach by employing multiple polynomial libraries rather than relying on a single fixed library. This allows the model to adapt to different dynamic behaviors within the system, improving computational efficiency and modeling precision, particularly for systems with varying scales of dynamics.

In contrast to vanilla SINDy, where a constant-size feature library is applied across all dynamics, Multi-Library SINDy introduces flexibility by assigning specific libraries to different regions of the system, tailoring the complexity based on the dynamics at each scale. This segmented approach reduces computational overhead and improves the adaptability of the model to multiscale systems.

The overall algorithm for SINDy, incorporating multiple libraries, is presented in Algorithm 2:

Algorithm 2 Sparse Identification of Nonlinear Dynamics Algorithm

Inputs:
D :=dataset of positions, $\dot{D}$:=dataset of velocities, Λ :=Thresholds, M := degree of the polynomial models, ϵ := Loop control parameter
Algorithm
$n \leftarrow sizeColumn(D)$
for $i = 1 : n$ **do**
  $d \leftarrow \frac{(n+M[i])!}{n!M[i]!}$
  $\phi \leftarrow LibraryConstruction(D, M[i])$
  $\Sigma \leftarrow STLS(\Lambda[i], M[i], n, d, \phi, \epsilon, \dot{D}[:, i])$
end for

2 Analysis of the Proposed Method

2.1 Assumptions

1. We are only analyzing the polynomial library functions.
2. We assume that $m \geq n$, i.e. we are using a polynomial which has an order m, which bigger than the number of states n, and that the model order drop one at a time, meaning that the highest polynomial order that we have is m and the lowest order would be $m - n + 1$.
3. We assume that the number of data-points (which will determine the number of rows to our libraries) is lower bounded by the number of parameters that the model initially has, to prevent under-determined problem.
4. According to [7] using STLS, in the worst case scenario, we have the maximum number of iterations in optimization loop equal to the number of parameters in the model.

n	Number of states
m	Maximum polynomial degree
k	Number of data-points
$d_{n,i}$	Number of parameters of a polynomial library of degree i
K_i	Ratio of parameters of lower order i to the maximum order m
P	Ratio of dataset size to the number of parameters of the largest model

Table 1: Notations

2.2 Complexity Analysis

We show number of states by n, largest polynomial degree by m and k is the number of data-points. We define $K_i \doteq \frac{d_{n,i}}{d_{n,m}}$ as ***Ratio of Efficiency***, which can be described as the ratio of parameters of lower order i to the maximum order m. We also consider $k = p \cdot d_{n,m}$, where p is the ratio of dataset size to the number of parameters of the largest model.

The total number of parameters for the polynomials up to degree i can be achieved using summation of combinations with repetition, which would be:

$$d_{n,i} = \sum_{r=0}^{i} \binom{n+r-1}{r} = \frac{(n+i)!}{(n)!(i)!} \tag{5}$$

Where i is the degree of the proposed library. We consider only the changes made in the library construction and optimization loop, and we focus our analysis only on these two. The original case would have the number of operations equal to:

$$O_{n,m}^{original} = O_{matrix}^{original} + O_{Opt}^{original} = ((k + \frac{n}{2}) + \frac{n}{2} d_{n,m}) d_{n,m} \tag{6}$$

For the new method, we have the number of operations needed as:

$$O_{n,m}^{new} = O_{matrix}^{new} + O_{Opt}^{new} = \sum_{i=m-n+1}^{m} d_{n,i}(d_{n,i} + k) \tag{7}$$

We begin the analysis: naturally, we would want the following value to be positive, so we define *Efficiency* (E) by:

$$O_{n,m}^{original} - O_{n,m}^{new} \geq 0 \tag{8}$$

Which results in the following equation:

$$\frac{n}{2d_{n,m}} + \frac{n-2}{2} \geq \sum_{i=m-n+1}^{m-1} K_i^2 + PK_i \tag{9}$$

This equation can be used beforehand to check if for a specific task using this method would save us any computations. Full proof is provided in Appendix A.

Figure 1, shows efficient model orders for different number of states and $p = 1$. The figure shows that, as the number of features increases, the effectiveness of using multiple models increases. The figure also shows an increase in the range of efficient orders, e.g. in the case of 10 features, we have the option of having ten very large polynomial models with degrees from 21 to 31.

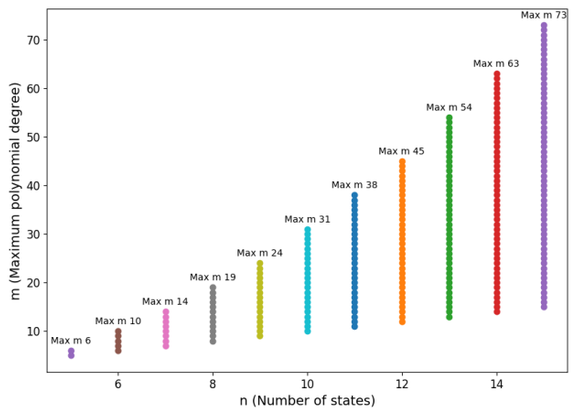

Figure 1: Efficient model orders for different number of states and $p = 1$

As a concluding argument in this section, it should be clear from the mathematical arguments before that if we are in the ROE range, instead of dropping down the model orders one at a time, we can drop more and have computational efficiency.

3 Case Study

3.1 Problem Statement

A toy system is created with the following dynamics. The values of the parameters k_1, k_2, k_3, k_4 and k_5 are presented in Table 2:

$$\frac{dx}{dt} = f(x), \quad x(0) = \begin{bmatrix} 0.2921 \\ 0.5351 \\ 0.2298 \\ 0.3736 \\ 0.5926 \end{bmatrix}, \quad x \in R^5 \tag{10}$$

$$f(x) = \begin{bmatrix} k_1 \cdot x_1^3 x_2^2 x_4 - 2k_2 \cdot x_3^2 x_4^2 x_5 \\ k_2 \cdot x_3^2 x_4^2 x_5 - k_3 x_1^2 x_5 x_3 - k_4 \cdot x_1 x_4 x_5 \\ k_3 \cdot x_1^2 x_5 x_3 - k_4 \cdot x_1 x_4 x_5 \\ k_4 \cdot x_1 x_4 x_5 \\ k_5 \cdot x_2 x_4 - x_1 \end{bmatrix} \tag{11}$$

Table 2: Parameter values of the process.

Parameter	Value
k_1	1.6828
k_2	1.8033
k_3	1.0563
k_4	1.9885
k_5	1.2025

3.2 Analysis

In vanilla SINDy, we select a library which contains all the possible combinations of terms. In this case it has to be a at least 6-th degree polynomial. Table 3 shows the number of parameters needed for different polynomial models.

Table 3: Parameter count for different polynomial models

Model degree	Parameter count
2	21
3	56
4	126
5	252
6	462

3.2.1 Matrix Construction

Since the 6-th order polynomial contains 2-nd, 3-rd, 4-th and 5-th order polynomials, we calculate the 6-th order once and pick the first 21 parameters for the 2-nd order, 56 parameters for the 3-rd order, 126 parameters for the 4-th order and 252 parameters for the 5-th order polynomial. In total, 455k operations would be added with our multi library method (***Number of data points are shown with k***):

$$O_{increase} = k(455) > 462 * 455 = 462 * (252 + 126 + 56 + 21) = 210,210 \tag{12}$$

The latter part is based upon the fact that intuitively, to prevent under-determined least squares, we need at least data points equal to the number of parameters. Simple calculation for this specific example of 5 features, would reveal that the difference between each order would result in the number of parameters below:

$$\frac{(5+m)!}{(5)!(m)!} \tag{13}$$

3.2.2 Optimization

The tricky part comes down to the optimization part. In simplest words, in worst case scenario, the number of operations can be calculated as ***number of iterations, i***, times the ***number of parameters that need to be compared to sparsity factor, m***. In vanilla case, it is roughly equal to i*m*n:

$$O = (5 * 462)i = 2310i \tag{14}$$

The value for number of iterations is said to be at most equal to the number of parameters [7]. So in this example the number of operations is going to be equal to:

$$O_{original} = (5 * 462 * 462) + O_{rest} = 1,067,220 + O_{rest} \tag{15}$$

In our proposed method, this number would reduce to different parts. Let us illustrate in the example. In this example, n would be equal to 5. So the number of operations would equal to:

$$O_{new} = (462^2 + 252^2 + 126^2 + 56^2 + 21^2) + O_{increase} + O_{rest} = 296,401 + 210,210 + O_{rest} = 506,611 \tag{16}$$

Meaning that 52% of the original number operations are skipped now.

3.3 Simulations

The $\dot{X}$ trajectories were produced using MATLAB ODE45 and used as the input in the python implementation of Multi-Library SINDy. We created 660 datapoints, and used 462 of them as training data (exactly equal to the number of parameters for the largest model), which is equal to 70% of the dataset and

the rest as the testing data. For the simulations done below, we used a modified version of SINDy, in both single and multiple library which is called STRidge [8]:

$$\begin{aligned} S^k &= \left\{1 \le j \le n : \left|x_j^k\right| \ge \lambda\right\} \\ x^{k+1} &= \underset{x\in\mathbb{R}^n:\mathrm{supp}(x)\subseteq S^k}{\mathrm{argmin}} \|Ax-b\|_2 + \alpha\cdot\|x\|_2^2 \end{aligned} \tag{17}$$

The only difference betweein STRidge and STLS is the addition of L_2-Regularization term in STRidge. The reason we are using it is solely to numerically stabilize the matrix inversion operation. Table 4 shows the number of parameters (equivalently, the number of functions) each method uses to track the trajectories of $\dot{x_1}$ to $\dot{x_5}$.

Table 4: Number of functions in Single & Multiple library models

Variable	Single Library	Multiple Library
x_1	6	6
x_2	5	7
x_3	8	9
x_4	4	13
x_5	18	7

Performance of the two models is shown in the Table 5, where MSE of training and testing for each model is listed.

Table 5: Performance of Single & Multiple library models

	Variable	Single Library	Multiple Library
Training	x_1	1.1619e-13	1.1768e-14
	x_2	4.7109e-11	5.2924e-14
	x_3	5.1520e-13	1.2422e-12
	x_4	2.4936e-09	3.4302e-14
	x_5	3.1286e-15	5.8562e-14
Testing	x_1	8.7627e-11	2.0181e-11
	x_2	1.2358e-09	3.8429e-10
	x_3	2.2204e-09	1.4349e-09
	x_4	4.3251e-08	4.7891e-08
	x_5	9.5500e-09	9.0278e-09

Figs. 2 to 6 shows the actual and predicted trajectories of $\dot{x_1}$ to $\dot{x_5}$ in the training of both models.

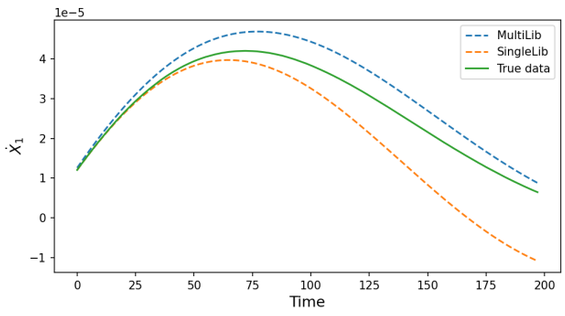

Figure 2: Testing of $\dot{x_1}$

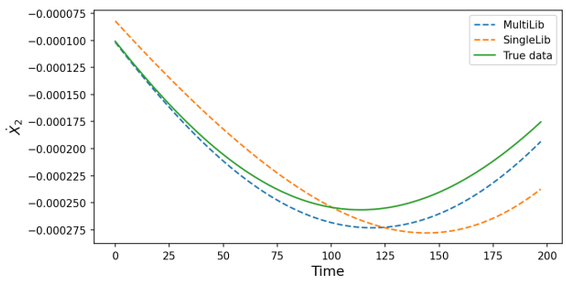

Figure 3: Testing of $\dot{x_2}$

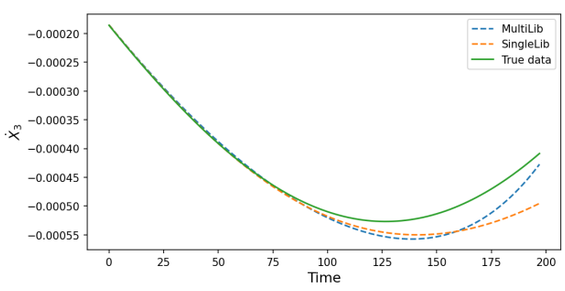

Figure 4: Testing of $\dot{x_3}$

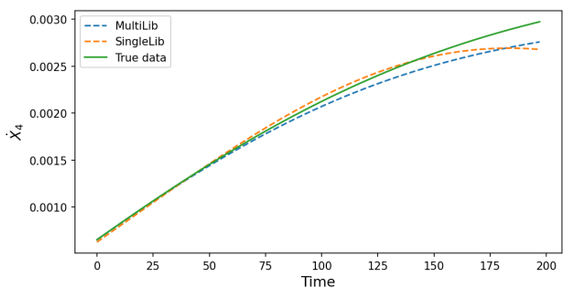

Figure 5: Testing of $\dot{x_4}$

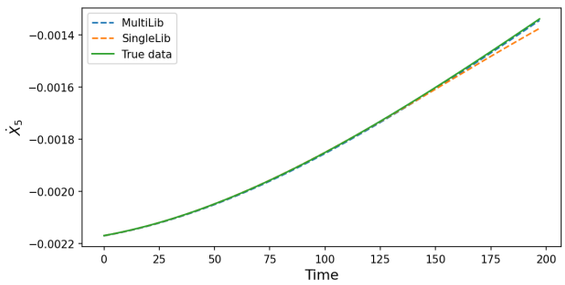

Figure 6: Testing of $\dot{x_5}$

4 Discussion

Evidently from the table 5 and Figs. 2 to 6, although we are saving computation, we are not experiencing any decrease in generalization. Intuitively, we think of Multiple libraries as an initialization method, that can direct SINDy towards a sparse model much faster and in a more efficient manner. The introduction of the Multi Library SINDy approach brings forth significant enhancements in modeling dynamic systems by utilizing multiple polynomial libraries tailored to specific system dynamics. This section discusses the implications, advantages, and potential for further research.

4.1 Enhanced Model Adaptability and Precision

The Multi Library SINDy framework enhances adaptability to diverse system dynamics by allowing specific libraries to target distinct behaviors within the system. This approach improves the precision of the model by fitting more appropriate functions to different parts of the data, thereby avoiding overfitting and underfitting issues common in single-library approaches. The case study presented demonstrates how this method can effectively capture complex interactions within a dynamic system without the need for excessive computational resources.

4.2 Computational Efficiency

Although managing multiple libraries introduces additional complexity, the overall computational efficiency is improved in comparison to traditional SINDy approaches. By reducing the dimensionality of the problem within each targeted library, the optimization processes are significantly simplified, leading to faster convergence and reduced computation time. This balance between complexity and efficiency is critical in scenarios where real-time modeling and prediction are required.

4.3 Application Scope

The Multi Library SINDy is particularly advantageous in fields where systems exhibit multifaceted dynamics, such as climate science, economics, and biomedical engineering. For instance, in climate modeling, different libraries can be designed to model atmospheric, oceanic, and terrestrial dynamics separately, each with its specific characteristics. However, challenges remain in extending this approach to extremely noisy or chaotic systems where defining appropriate libraries upfront may be difficult.

4.4 Real-World Applications

There are numerous potential applications for Multi Library SINDy in real-world scenarios. In financial forecasting, for example, the method can be used to model different economic variables with distinct dynamics, providing more accurate predictions than traditional methods. Similarly, in biomedical engineering, it could help model the nonlinear interactions in biological systems under different physiological conditions.

4.5 Future Work

Future research could focus on several areas to enhance the Multi Library SINDy approach. Algorithmic improvements, such as automated library selection and dynamic library updating based on incoming data, could provide more flexibility and accuracy. Extending the approach to include non-polynomial libraries, such as Fourier series or wavelets, might also be explored to better capture oscillatory or multi-scale phenomena. Additionally, integrating machine learning techniques could offer improvements in handling large datasets and complex dynamics.

4.6 Comparison with Other Methods

Comparing Multi Library SINDy to other dynamic system modeling techniques, such as neural networks or ensemble methods, reveals its unique strengths in interpretability and computational simplicity. While neural networks may provide high accuracy, they often lack the transparency needed for scientific analysis. Multi Library SINDy offers a compelling balance between accuracy and interpretability, making it suitable for scientific applications where understanding the underlying dynamics is crucial.

5 Conclusions

In conclusion, the Multi Library SINDy method represents a significant advancement in the sparse identification of dynamical systems, providing a flexible and efficient tool for modeling complex systems across various domains. Its ability to adapt to specific system characteristics without a significant increase in computational complexity makes it a promising approach for both theoretical studies and practical applications.

References

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

[2] Huimei Ma, Xiaofan Lu, and Linan Zhang. Extracting parametric dynamics from time-series data. *Nonlinear Dynamics*, 111(16):15177–15199, 2023.

[3] Feng Jiang, Lin Du, Fan Yang, and Zi-Chen Deng. Regularized least absolute deviation-based sparse identification of dynamical systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 33(1), 2023.

[4] William A Brock. Nonlinearity and complex dynamics in economics and finance. In *The economy as an evolving complex system*, pages 77–97. CRC Press, 2018.

[5] SL Brunton and JN Kutz. *Data-driven science and engineering: machine learning, dynamical systems, and control: by SL Brunton and JN Kutz, 2019, Cambridge, Cambridge University Press, 472 pp.,£ 49.99 (hardback), ISBN 9781108422093. Level: postgraduate. Scope: textbook.*, volume 60. Taylor & Francis, 2019.

[6] Eurika Kaiser, J Nathan Kutz, and Steven L Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A*, 474(2219):20180335, 2018.

[7] Linan Zhang and Hayden Schaeffer. On the convergence of the sindy algorithm. *Multiscale Modeling & Simulation*, 17(3):948–972, 2019.

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

[9] Herbert Robbins. A remark on stirling's formula. *The American mathematical monthly*, 62(1):26–29, 1955.

A Proof

The number of parameters needed for a polynomial degree r would be:

$$\binom{n+r-1}{r} = \frac{(n+r-1)!}{r!(n-1)!} \tag{18}$$

Hence the total number of parameters for the polynomials up to degree i can be achieved using summation of combinations with repetition, which would be:

$$d_{n,i} = \sum_{r=0}^{i} \binom{n+r-1}{r} = \frac{(n+i)!}{(n)!(i)!} \tag{19}$$

Where i is the degree of the proposed polynomial and n is the number of features. $d_{n,i}$ is the indication of feature library column length.

Since our suggested library only makes changes in the library construction and optimization loop, we focus our analysis only on these two. The original case would have the number of operations equal to:

$$O_{matrix}^{original} = k \cdot d_{n,m} \quad , \quad O_{Opt}^{original} = n \cdot \sum_{i=1}^{d_{n,m}} i = n\frac{d_{n,m}(d_{n,m}+1)}{2} \tag{20}$$

$$O_{n,m}^{original} = O_{matrix}^{original} + O_{Opt}^{original} = k \cdot d_{n,m} + n\frac{d_{n,m}(d_{n,m}+1)}{2} = ((k+\frac{n}{2}) + \frac{n}{2}d_{n,m})d_{n,m} \tag{21}$$

Where k is the number of datapoints.

First term is the number of operations required for matrix construction, in which column size is the number of parameters we have and the row size should be at least equal to the number of parameters.

Second term is for optimization loop, in which the iteration is over the models, and for every iteration we would have the number of parameters to go over in an optimization loop and maximum number of iterations in each model is equal to the number of parameters in the model.

For the new method, we have the number of operations in matrix construction as:

$$O_{matrix}^{new} = k \sum_{i=m-n+1}^{m} d_{n,i} \tag{22}$$

The number of operations in optimization loop as:

$$O_{Opt}^{new} = \sum_{i=m-n+1}^{m} \sum_{j=1}^{d_{n,i}} j = \sum_{i=m-n+1}^{m} \frac{d_{n,i}(d_{n,i}+1)}{2} \leq \sum_{i=m-n+1}^{m} d_{n,i}^2 \tag{23}$$

In which again we consider that for each model, we go through every parameter in that model and the adaptive number of runs for every model is at most equal to the number of parameters it would have to eliminate. Overall:

$$O_{n,m}^{new} = O_{Opt}^{new} + O_{matrix}^{new} = \sum_{i=m-n+1}^{m} d_{n,i}(d_{n,i}+k) \tag{24}$$

We begin the analysis: naturally, we would want the following value to be positive, so we define *Efficiency* (E) by:

$$\left(k + \frac{n}{2}\right) d_{n,m} + \frac{n}{2} d_{n,m}^2 - \sum_{i=m-n+1}^{m-1} d_{n,i}(d_{n,i}+k) \geq 0 \tag{25}$$

$$\left(k + \frac{n}{2}\right) d_{n,m} + \frac{n}{2} d_{n,m}^2 - d_{n,m}^2 - k d_{n,m} - \sum_{i=m-n+1}^{m-1} d_{n,i}(d_{n,i}+k) \geq 0 \tag{26}$$

$$\frac{1}{d_{n,m}} \left(\sum_{i=m-n+1}^{m-1} \frac{n}{2(n-1)} d_{n,m} - k d_{n,i} + \left(\frac{n-2}{n-1}\right) d_{n,m}^2 - d_{n,i}^2 \right) \geq 0 \tag{27}$$

We define $K_i \doteq \frac{d_{n,i}}{d_{n,m}}$ as ***Ratio of Efficiency***, which can be described as the ratio of parameters of lower order i to the maximum order m:

$$K_i = \frac{d_{n,i}}{d_{n,m}} \Rightarrow \sum_{i=m-n+1}^{m-1} \frac{n}{2(n-1)} - k K_i + \frac{n-2}{2(n-2)} d_{n,m} - K_i d_{n,i} \geq 0 \tag{28}$$

$$\sum_{i=m-n+1}^{m-1} \frac{n}{2(n-1)} - k K_i + d_{n,m} \left(\frac{n-2}{2(n-1)} - K_i^2 \right) \geq 0 \tag{29}$$

$$\sum_{i=m-n+1}^{m-1} -d_{n,m} K_i^2 - k K_i + \frac{n}{2(n-1)} + \frac{n-2}{2(n-1)} d_{n,m} \tag{30}$$

Consider $k = p \cdot d_{n,m}$, where p is the ratio of dataset size to the number of parameters of the largest model:

$$P \cdot d_{n,m} = k \Rightarrow d_{n,m} \sum_{i=m-n+1}^{m-1} \left(-k_i^2 - P K_i + \frac{n-2}{2(n-1)} \right) + \frac{n}{2} \geq 0 \tag{31}$$

$$\frac{n}{2} + \frac{n-2}{2} d_{n,m} - d_{n,m} \sum_{i=m-n+1}^{m-1} K_i^2 + P K_i \geq 0 \tag{32}$$

$$\frac{n}{2 d_{n,m}} + \frac{n-2}{2} \geq \sum_{i=m-n+1}^{m-1} K_i^2 + P K_i \tag{33}$$

To calculate $d_{n,i}$, Stirling approximation [9] can also be used To extract an analytical form of the factorials:

$$d_{n,i} = \frac{(n+i)!}{n! i!} \approx \frac{(n+i)^{n+i+\frac{1}{2}}}{n^{n+\frac{1}{2}} i^{i+\frac{1}{2}}} \cdot \frac{1}{\sqrt{2\pi}} = d_{n,i} \tag{34}$$