MANCHESTER 1824
The University of Manchester

Learning Chemical Reactions from Simulated Data

A thesis submitted to the University of Manchester for the degree of
Master of Philosophy
in the Faculty of Science and Engineering

2023

Ayana Mussabayeva
Department of Mathematics, School of Natural Sciences, Faculty of Science and Engineering

Contents

Word Count: 9244

List of figures

List of tables

Abstract

Catalytic chemical reactions are intricate processes that can be represented as a system of ordinary differential equations (ODEs) by deriving the rate laws for each of the species in a reaction. Different kinetic mechanisms can be associated with these systems of ODEs.

A kinetic mechanism illustrates how the reaction rate depends on the kinetic constants and the concentrations of the species. Analyzing the kinetics of a reaction is crucial for understanding the behaviour of the reaction and for improving its performance. Chemists aim to automate this process by enabling extraction of the kinetics from time-series concentrations data. The problem is formulated as a least squares problem. The data used for the experiments is generated numerically using a fourth order Runge-Kutta method. Different data sampling is used to check how the distribution of the concentration data affects the final result. Three sampling options are used in the experiments: equispaced sampling, points adaptively chosen by Runge-Kutta-Fehlberg 4(5) and Chebyshev points.

Apart from receiving a solution that fits the data trajectories, it is important to obtain a sparse solution. This can be achieved through the use of iterative thresholding algorithms or regularization techniques. Therefore, the sequentially thresholded least squares (STLSQ) algorithm and Lasso regularization have been utilized to extract interpretable system of ODEs from the generated data trajectories.

The results obtained demonstrate that Lasso regularization is more robust against numerical errors and more frequently identifies the correct "active" components (those with non-zero coefficients on the right-hand side of the ODEs) compared to STLSQ. It has also been observed that the choice of numerical approximation for derivatives and data sampling significantly impacts the results.

Declaration

No portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright Statement

1. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the "Copyright") and they have given the University of Manchester certain rights to use such Copyright, including for administrative purposes.

2. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the "Copyright") and they have given the University of Manchester certain rights to use such Copyright, including for dministrative purposes.

3. The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the "Intellectual Property") and any reproductions of copyright works in the thesis, for example graphs and tables ("Reproductions"), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

4. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see `http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=24420`), in any relevant Thesis restriction declarations deposited in the University Library, the University Library's regulations (see `http://www.library.manchester.ac.uk/about/regulations/`) and in the University's policy on Presentation of Theses.

1 Introduction

A chemical reaction is a transformative process in which one or more reactants undergo a conversion, resulting in the formation of one or more products. Some substances, known as catalysts, have the ability to facilitate or accelerate these reactions. Consequently, chemical reactions that involve catalysts are referred to as catalytic reactions. For example, a product P is obtained from initial substrate A catalyzed by cat:

$$\mathrm{A} \xrightarrow{\mathrm{cat}} \mathrm{P}. \tag{1.1}$$

Catalytic reactions are complex processes, as there may be different elementary reactions hiding under the above expression. The transformation of A into P proceeds through a series of elementary reactions that involve the collision of normally two species, for example:

$$\begin{aligned} \mathrm{A} + \mathrm{cat} &\underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} \mathrm{catA} \\ \mathrm{catA} &\underset{k_{-2}}{\overset{k_2}{\rightleftharpoons}} \mathrm{P} + \mathrm{cat}. \end{aligned} \tag{1.2}$$

It is seen from the above that there are two intermediate catalysts cat and catA involved in the reaction. The coefficients k_1, k_{-1}, k_2 and k_{-2} in the equation above determine how fast species react with each other. These constants are called kinetic constants. Catalytic reactions can be represented in the form of a kinetic mechanism, which is a series of elementary reactions. A kinetic mechanism shows the steps leading from the starting solutions to the formulation of intermediates and final products. In catalytic reactions a catalyst is regenerated at the end of a reaction, so it does not require a large amount of catalyst. A kinetic mechanism can be shown graphically as in Fig. 1.1. It is seen that the catalyst reacts with substrate A and the same catalyst is resulted at the end of the cycle, when catA reacts with P.

The kinetic mechanism and its equations show the dependency of the reaction's rate with the kinetic constants and species' concentrations. There are many different types of mechanisms. Even a simple transition from reactant A to product P can be interpreted using different kinetic mechanisms (see Fig. 1.2), so it is hard to analyse which one corresponds to some given set of time-series data.

Thus, a simple transformation of A into P can be interpreted as different mechanisms, and it is essential to understand which is the correct one. A precise analysis of reaction's mechanism is necessary for its further usage, e.g. for its improvement, for using it in some

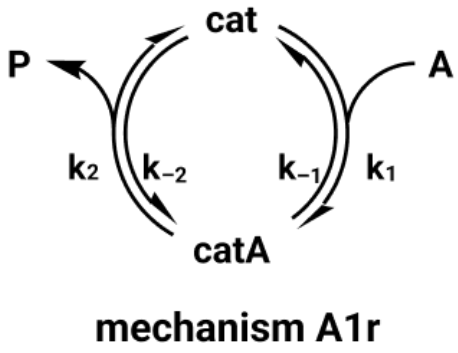

Figure 1.1. Scheme of A1r reaction mechanism

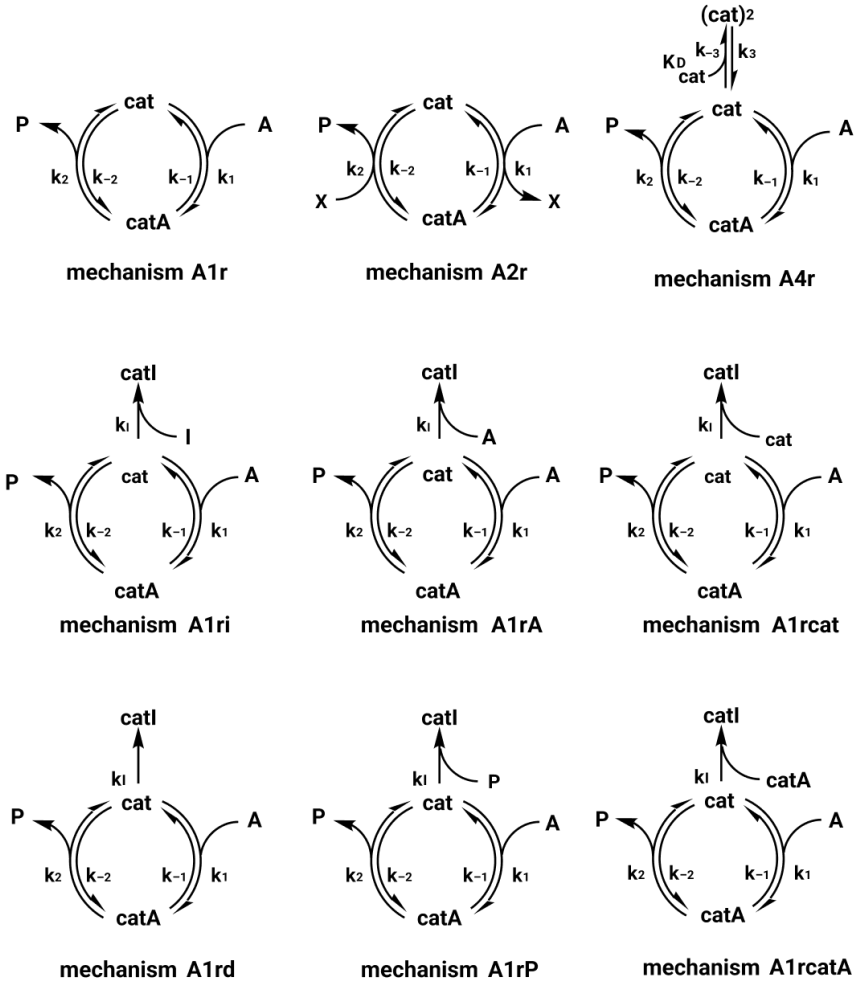

Figure 1.2. Diagrams of common kinetic mechanisms for one product and one reactant

industrial process or for stopping it in case if it is an undesired process.

Each kinetic mechanism can be represented as a system of ODEs. The differential equation for each element can be formed by observing the kinetic cycle and breaking it down into elementary steps. The direction of the step affects the sign of the step's product in the resulting ODE. For instance, the equation for the rate of change of the concentration of [cat] is formed by observing two reversible transitions (1.2). As both of the reactions are reversible, each can be represented in two elementary steps. Thus, the resulting ODE is shown as the sum of the four steps (see Fig.1.3). The kinetic constants k_1, k_{-1}, k_2, k_{-2} show how fast the transition

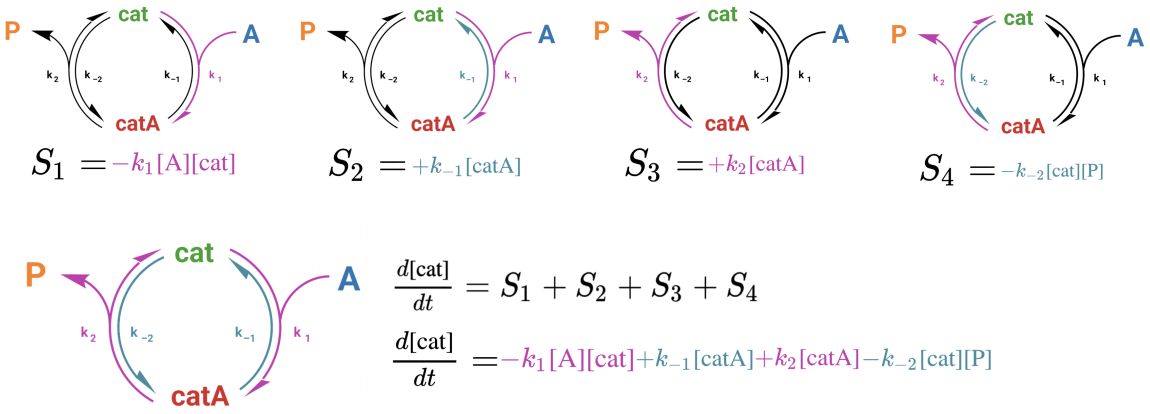

Figure 1.3. Deriving a differential equation from kinetic cycle's steps

of each step is.

The same procedure can be done for each element in the reaction. For instance:

$$\frac{d[\mathrm{P}]}{dt} = k_2[\mathrm{catA}] - k_{-2}[\mathrm{cat}][\mathrm{P}],$$

is the rate of product formation, which depends on concentrations of [catA] and [cat][P] at time t. The kinetic coefficient k_2 has positive sign because of the forward direction of the transition from catA to P, while k_{-2} has negative sign because of opposite direction in the reversible step (see Fig. 1.3).

By deriving the equations for each substrate in the reaction, we can describe the mechanism A1r by a system of ODEs:

$$\begin{aligned}
\frac{d[\mathrm{A}]}{dt} &= -k_1[\mathrm{A}][\mathrm{cat}] + k_{-1}[\mathrm{catA}] \\
\frac{d[\mathrm{P}]}{dt} &= k_2[\mathrm{catA}] - k_{-2}[\mathrm{cat}][\mathrm{P}] \\
\frac{d[\mathrm{cat}]}{dt} &= -k_1[\mathrm{A}][\mathrm{cat}] + k_2[\mathrm{catA}] + k_{-1}[\mathrm{catA}] - k_{-2}[\mathrm{cat}][\mathrm{P}] \\
\frac{d[\mathrm{catA}]}{dt} &= k_1[\mathrm{A}][\mathrm{cat}] - k_2[\mathrm{catA}] - k_{-1}[\mathrm{catA}] + k_{-2}[\mathrm{cat}][\mathrm{P}].
\end{aligned} \tag{1.3}$$

Knowing that the catalytic reactions can be represented in a form of a system of ODEs, it is possible to formulate the problem mathematically. Further, if we were able to extract the information about the reaction directly from the system of ODEs, we would be able to identify the corresponding mechanism of the reaction. Analyzing the kinetics of a reaction is crucial for comprehending its nature and enhancing its performance. However, the current methods employed for kinetic analysis are highly limited as they often necessitate a manual comparison between the obtained time-series concentration profiles and the expected characteristics of plausible kinetic mechanisms. Among the most commonly used methods is the initial rates

method, which focuses on the rate change of a single reactant and assumes the concentrations of the remaining species to be constant. The primary drawback of this approach is that it tipycally provides only the information about the kinetics at the beginning of the reaction and does not consider the change of the reaction order. Over the last decades there have been several computational approaches proposed for biochemical data analysis. One of the early works in this area proposed correlation metric construction for time-series concentrations to draw the diagrams of chemical reactions [1]. This method required quite a lot of priori knowledge about the kinetic steps of the reaction, but as a result it provided possible kinetic cycles of the mechanism.

One of the most obvious ways to perform the kinetic analysis is to hold a number of experiments with different starting conditions (such as initial concentrations of reactants, temperature, etc). By observing the behaviour of the reaction with different starting conditions, chemists can define the type of mechanism and its kinetic properties. This is called the initial rates method. However, this classical method entails repeating the experiments, which can be both time-consuming and inefficient in terms of chemical reactant usage.

To overcome the described problem several methods such as reaction progress kinetic analysis (RPKA) [2] were proposed. This method compares the rate trajectories and tries to minimize the number of experiments held for identifying the kinetics. The disadvantage of the RPKA method is that it uses not the time-series concentrations, but rate data obtained by isothermal calorimetry, which is not suitable for some reactions.

A related method called variable time normalization analysis (VTNA) [3] can be used for inferring kinetics. It also performs the visual comparison of the chemical data, but unlike RPKA it uses reaction concentration profiles. Both RKPA and VTNA are quite simple and the main advantage is that these methods minimize the number of experiments held to define the kinetics of a reaction. However, these methods still require several experiments to be carried out. The more complex the mechanism, the more experiments are required. In addition to that, visual kinetic analysis does not obtain precise values of kinetic constants [4], but measures the elasticities of the species, which show how much the rate of the reaction changes with the change of concentration for each species. The main drawback of these existing methods is that they generalize the whole information present in the kinetics over several experiments with different initial value of just one reaction component. Classical initial rates method and visual kinetic analysis compare the given kinetics with several possible mechanistic hypotheses, but they fail to represent complex mechanisms.

Extraction of reaction kinetics can be performed using general-to-specific (or "top-down") and simple-to-general (or "bottom-up") iterative approaches [5]. These methods construct the dictionary of possible components in the equation. General-to-specific model selection starts with the full dictionary of all possible components and further reduces it iteratively. By contrast, the bottom-up approach chooses a single initial component of the equation and then adds iteratively new ones. Both of these methods can be successfully applied for inferring kinetics from time-series data [6]. The main drawback of these methods is their time-complexity and

inability to work with a large number of reacting species.

Some methods for reconstructing bio-chemical pathways solve inverse problems for dynamic models [7], [8]. Another possible approach is to use Bayesian inference in combination with Markov chain Monte Carlo method for uncertainty qualification when inferring the kinetics of a reaction [9]. However, most of these methods require a priori knowledge about the structure of a kinetic mechanism analyzed, which limits their applicability [10]. It is also noted, that even the most promising methods fail to infer kinetics of complex models. For instance, the accuracy in the reconstruction of the reaction topology depends on the number of chemical species and their reaction kinetics when using MIKANA (method to infer kinetics and network architecture) [11].

Generally, the idea is to have time-series data, obtained during the catalytic reaction, as an input. These data should be processed by the algorithm, which can output a system of ODEs, a scheme of a corresponding kinetic mechanism and its type. This concept is shown in Fig. 1.4.

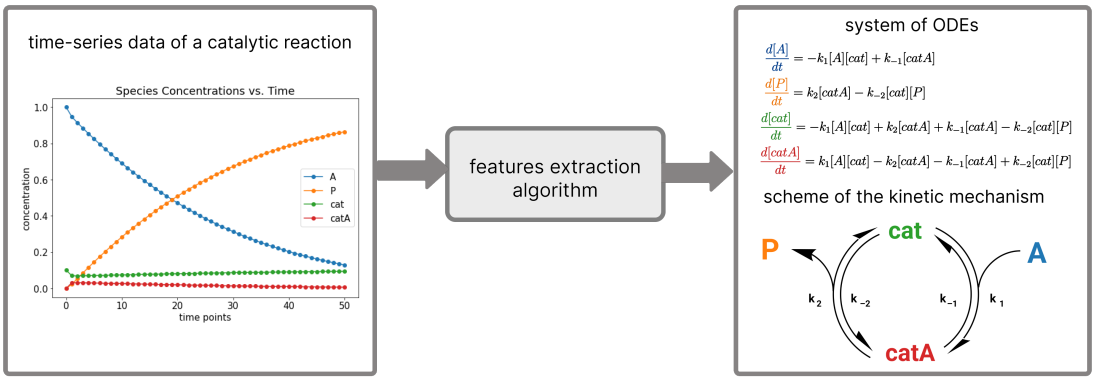

Figure 1.4. General concept of automating the kinetic mechanism identification

The application of neural networks can partly solve this problem and infer the type of mechanism from the measured data [12]. However, the main goal is not to just identify the mechanisms out of a “black box”, but to extract the differential equations, explaining the behaviour of the reaction.

1.1 Mathematical Formulation

Each ODE of the system can be written as:

$$\frac{d}{dt}x_j = F_j(x_1, \dots, x_n), \tag{1.4}$$

where $j \in \{1, 2, ..., n\}$, n is the number of species involved in a chemical reaction and x_j is the concentration of the j-th species changing over time t. The function $F_j(x_1, ..., x_n)$

represents the dynamic constraints that define the equation.

The input data of concentrations is then defined as a tall skinny data matrix X of size $[k \times n]$, where k is the number of data points. The number of ODEs in a system is the same as the number of species. Then, the input data can be written as:

$$X = \begin{bmatrix} x_1(t_1) & x_2(t_1) & \cdots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \cdots & x_n(t_2) \\ \vdots & \vdots & \vdots & \vdots \\ x_1(t_k) & x_2(t_k) & \cdots & x_n(t_k) \end{bmatrix} = \begin{bmatrix} | & | & \cdots & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \\ | & | & \cdots & | \end{bmatrix}. \tag{1.5}$$

The whole system of ODEs can be identified by introducing matrix $\dot{X}$:

$$\dot{X} = \begin{bmatrix} | & | & \cdots & | \\ F_1 & F_2 & \cdots & F_n \\ | & | & \cdots & | \end{bmatrix}. \tag{1.6}$$

The main goal is to recover each F_j, $j \in \{1, 2, ..., n\}$, from the measured data, representing $\dot{X}$ as a combination of candidate functions. The candidate functions are presented using a feature dictionary $\widetilde{X}$. Each column of $\widetilde{X}$ is a candidate function of the columns of X. For inferring kinetic mechanism problem, the feature dictionary stores original data vectors of X, its squared terms and its pairwise products. By defining $\bar{X}$ as a matrix storing pairwise products of the original data vectors, it can be said that:

$$\widetilde{X} = [X, \bar{X}, X^2] = \begin{bmatrix} | & ... & | & | & ... & | & | & ... & | \\ \mathbf{x}_1 & ... & \mathbf{x}_n & \mathbf{x}_1\mathbf{x}_2 & ... & \mathbf{x}_{n-1}\mathbf{x}_n & \mathbf{x}_1^2 & ... & \mathbf{x}_n^2 \\ | & ... & | & | & ... & | & | & ... & | \end{bmatrix}. \tag{1.7}$$

The number of column vectors in this features dictionary is $m = \frac{n^2+3n}{2}$, as the features dictionary includes n species, n element-wise squared vectors and the unique pairwise Hadamard products of n species. Not all of the possible features are "active" components, i.e. not all of them will be presented in the right-hand side of the ODEs. There are m possible components (of features), but there are only s_{ex} number of the correct active components, which should be presented in the right-hand side of the system. As not all of the m features should appear in the ODE, it is necessary to determine their state by introducing the vectors of coefficients $\mathbf{w} = [w_1 \quad w_2 \quad ... \quad w_m]$ for each equation. Those features which are not active will have zero coefficients. The active features should have the coefficients which can be interpreted as kinetic constants. For the whole system of ODEs we get a matrix of weights W of size $[m \times n]$, which satisfies the following expression:

$$\dot{X} = \widetilde{X}W. \tag{1.8}$$

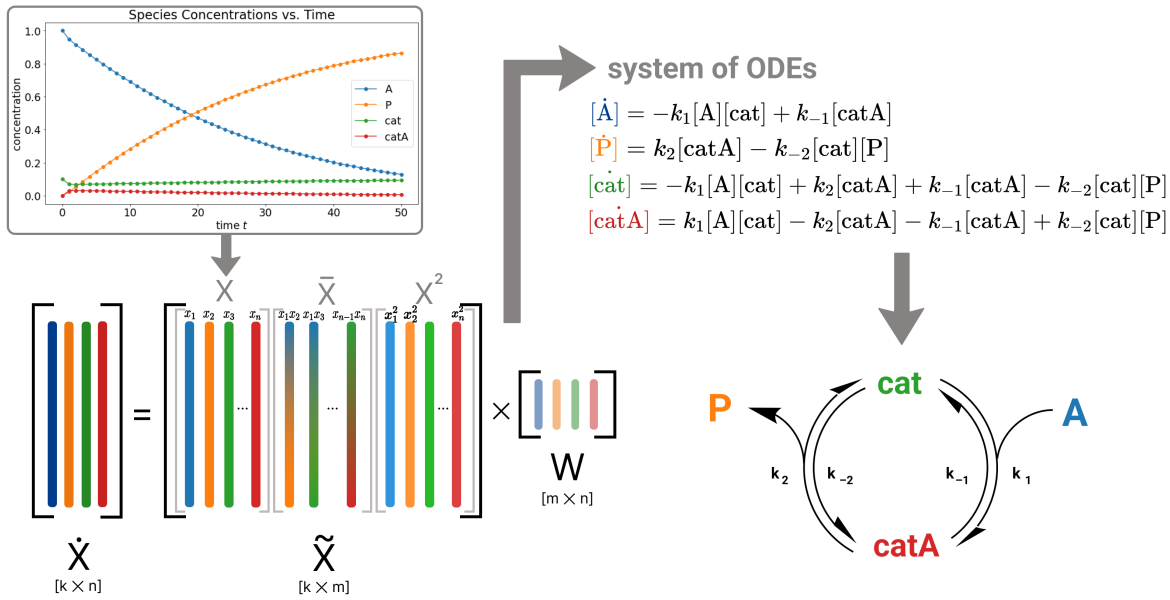

Figure 1.5. General scheme for ODEs extraction from time-series data

The general scheme for ODEs extraction process from chemical data is represented in Fig. 1.4.

The matrices X and $\widetilde{X}$ are constructed using the input data trajectories of chemical concentrations. But in order to find the coefficients W, firstly it is necessary to obtain the values of the derivatives matrix $\dot{X}$ (1.6), which is also a skinny matrix of the same size as X. To do so, $\dot{X}$ could be approximated numerically. The matrix Y is introduced for the numerical approximation of $\dot{X}$, which can be obtained e.g. using finite difference method or spline approximation:

$$Y \approx \dot{X}. \tag{1.9}$$

Finally, it can be said that the goal is to find the values of the sparse matrix W, such that:

$$Y \approx \widetilde{X}W. \tag{1.10}$$

1.2 Mechanism A1r

In the following experiments we focus on the simplest mechanism with two intermediates called Mechanism A1r. The diagram of this mechanism is presented in Fig. 1.1 and its behaviour is described by (1.3).

When investigating the equations more closely we can spot that there are linear relations between the equations:

$$
\begin{aligned}
[\dot{\mathrm{A}}] &= -k_1[\mathrm{A}][\mathrm{cat}] + k_{-1}[\mathrm{catA}] \\
[\dot{\mathrm{P}}] &= k_2[\mathrm{catA}] - k_{-2}[\mathrm{cat}][\mathrm{P}] \\
[\dot{\mathrm{cat}}] &= -k_1[\mathrm{A}][\mathrm{cat}] + k_{-1}[\mathrm{catA}] + k_2[\mathrm{catA}] - k_{-2}[\mathrm{cat}][\mathrm{P}] \\
[\dot{\mathrm{catA}}] &= k_1[\mathrm{A}][\mathrm{cat}] - k_{-1}[\mathrm{catA}] - k_2[\mathrm{catA}] + k_{-2}[\mathrm{cat}][\mathrm{P}].
\end{aligned}
\tag{1.11}
$$

It can be concluded that for the mechanism A1r:

$$
\begin{aligned}
[\dot{\mathrm{cat}}] &= [\dot{\mathrm{A}}] + [\dot{\mathrm{P}}] \\
[\dot{\mathrm{catA}}] &= -[\dot{\mathrm{cat}}].
\end{aligned}
\tag{1.12}
$$

When rewriting the system of ODEs in matrix form, the exact matrix of weights for mechanism A1r is a $[4 \times 14]$ matrix:

$$
W_{\text{ex}} = \begin{bmatrix}
0 & 0 & 0 & k_{-1} & 0 & -k_1 & 0 & 0 & \dots & 0 \\
0 & 0 & 0 & k_2 & 0 & 0 & -k_{-2} & 0 & \dots & 0 \\
0 & 0 & 0 & k_2 + k_{-1} & 0 & -k_1 & -k_{-2} & 0 & \dots & 0 \\
0 & 0 & 0 & -k_2 - k_{-1} & 0 & k_1 & k_{-2} & 0 & \dots & 0
\end{bmatrix}.
\tag{1.13}
$$

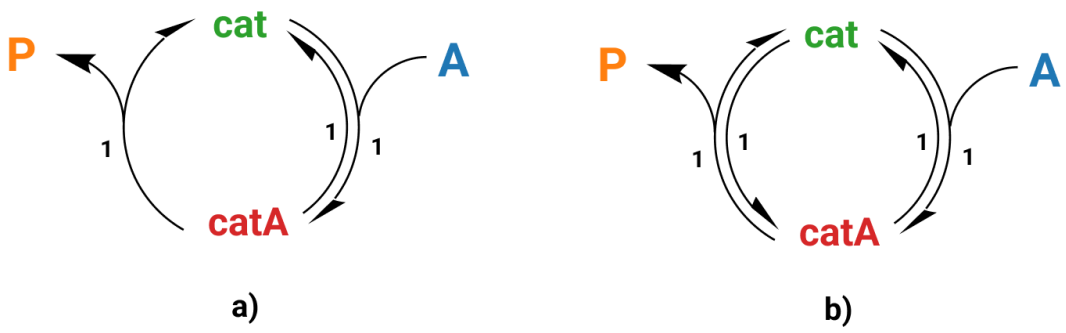

Figure 1.6. Mechanism A1r: a) kinetic constants: $k_1 = 1$, $k_{-1} = 1$, $k_2 = 1$. $k_{-2} = 0$ b) kinetic constants: $k_1 = 1$, $k_{-1} = 1$, $k_2 = 1$. $k_{-2} = 1$

There are two sets of kinetic constants chosen for data generation. The first set is $k_1 = 1$, $k_{-1} = 1$, $k_2 = 1$. $k_{-2} = 0$. The kinetic cycle with these constants is shown in Fig. 1.6(a). Substituting these constants into (1.3) we get:

$$
\begin{aligned}
[\dot{\mathrm{A}}] &= [\mathrm{catA}] - [\mathrm{A}][\mathrm{cat}] \\
[\dot{\mathrm{P}}] &= [\mathrm{catA}]
\end{aligned}
\tag{1.14}
$$

$$
\begin{aligned}
[\dot{\text{cat}}] &= 2[\text{catA}] - [\text{A}][\text{cat}] \\
[\dot{\text{catA}}] &= -2[\text{catA}] + [\text{A}][\text{cat}].
\end{aligned}
$$

In this case the exact number of active features (the number of components with non-zero coefficients) is $s_{\text{ex}} = 7$.

The second set of kinetic constants is $k_1 = 1$, $k_{-1} = 1$, $k_2 = 1$. $k_{-2} = 1$. The kinetic cycle with this set of constants is shown in Fig. 1.6(b). By inserting these into (1.3), the following equations with $s_{\text{ex}} = 10$ active components are obtained:

$$
\begin{aligned}
[\dot{\text{A}}] &= [\text{catA}] - [\text{A}][\text{cat}] \\
[\dot{\text{P}}] &= [\text{catA}] - [\text{cat}][\text{P}] \\
[\dot{\text{cat}}] &= 2[\text{catA}] - [\text{A}][\text{cat}] - [\text{cat}][\text{P}] \\
[\dot{\text{catA}}] &= -2[\text{catA}] + [\text{A}][\text{cat}] + [\text{cat}][\text{P}].
\end{aligned}
\tag{1.15}
$$

Knowing the system of ODEs, it is possible to generate data concentrations using numerical methods, such as Runge-Kutta, LSODA [13], etc.

2 Simulating Chemical Reactions by Numerical Integration

In order to conduct the experiments within this work, and so that we are able to measure the error precisely, synthetically generated data can be used, rather than the experimentally-obtained time- series concentrations obtained by chemists during a reaction. By knowing the pre-defined system of ODEs for a particular kinetic mechanism, one can generate the species concentrations by using different numerical integration methods, such as Euler methods or Runge-Kutta methods.

2.1 Runge–Kutta Method

The Runge–Kutta 4th order method is a popular method also called as classic Runge–Kutta method or RK45.

The basic idea of the method is to approximate the solution of the ODE at a discrete set of points in time. At each time step, the method uses a weighted average of four function evaluations to estimate the solution at the next time step. Knowing that each governing equation of the system can be written as an ODE (1.4), we can rewrite it in a vector form:

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}_1, \dots, \mathbf{x}_n) \tag{2.1}$$

Then, by knowing the initial concentration $\mathbf{x}^{(0)}$ for each ODE it is possible to find the next concentration values of $\mathbf{x}$ as:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \frac{h}{6}(\mathbf{s}_1 + 2\mathbf{s}_2 + 2\mathbf{s}_3 + \mathbf{s}_4), \tag{2.2}$$

where h is the chosen initial step value, which can be be estimated optimally by calculating partial derivatives at t_0 [14]. The step value h depends on k, since it can be calculated as:

$$h_k = t_{k+1} - t_k. \tag{2.3}$$

Nevertheless in order to keep the notation simple, the initial step value is used as $h = h_k$. The coefficients $\mathbf{s}_1$, $\mathbf{s}_2$, $\mathbf{s}_3$, $\mathbf{s}_4$ can be found as:

$$\mathbf{s}_1 = f(\mathbf{x}^{(k)}, t_k) \tag{2.4}$$
$$\mathbf{s}_2 = f(\mathbf{x}^{(k)} + \frac{h}{2}\mathbf{s}_1, t_k + \frac{h}{2}) \tag{2.5}$$
$$\mathbf{s}_3 = f(\mathbf{x}^{(k)} + \frac{h}{2}\mathbf{s}_2, t_k + \frac{h}{2}) \tag{2.6}$$
$$\mathbf{s}_4 = f(\mathbf{x}^{(k)} + h\mathbf{s}_3, t_k + h). \tag{2.7}$$

The coefficients in the weighted average (2.2) are chosen such that the method has fourth-order accuracy, meaning that the error in the solution approximately decreases by a factor of 16 as the time step is halved. In other words, the global error of the computed solution after several steps should be of order 4, while the truncation error accumulated at each iteration can be of order 5.

If the above expressions are expanded as Taylor series and substituted to (2.2), the error accumulated at each iteration is cancelled out to the 5th order, so it can be written as:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \frac{h}{6}(\mathbf{s}_1 + 2\mathbf{s}_2 + 2\mathbf{s}_3 + \mathbf{s}_4) + O(h^5). \tag{2.8}$$

It should also be noted that there can be many possible ways to compute $\mathbf{x}^{(k+1)}$ in such a way thatit results in the global error being of order 4. So, roughly speaking, Runge-Kutta of 4th order is not a single method, but a general name for a number of methods.

Runge-Kutta 4(5) or RK45 method is based on the Dormand-Prince (4,5) pair [15]. It is a predictor-corrector method, which calculates four coefficients, but also introduces two additional terms to correct the solution for the 5th order method:

$$\mathbf{s}_1 = f(\mathbf{x}^{(k)}, t_k), \tag{2.9}$$
$$\mathbf{s}_2 = hf(\mathbf{x}^{(k)} + \frac{1}{4}\mathbf{s}_1, t_k + \frac{1}{4}h), \tag{2.10}$$
$$\mathbf{s}_3 = hf(\mathbf{x}^{(k)} + \frac{3}{32}\mathbf{s}_1 + \frac{9}{32}\mathbf{s}_2, t_k + \frac{3}{8}h), \tag{2.11}$$
$$\mathbf{s}_4 = hf(\mathbf{x}^{(k)} + \frac{1932}{2197}\mathbf{s}_1 - \frac{7200}{2197}\mathbf{s}_2 + \frac{7296}{2197}\mathbf{s}_3, t_k + \frac{12}{13}h), \tag{2.12}$$
$$\mathbf{s}_5 = hf(\mathbf{x}^{(k)} + \frac{439}{216}\mathbf{s}_1 - 8\mathbf{s}_2 + \frac{3680}{513}\mathbf{s}_3 - \frac{845}{4104}\mathbf{s}_4, t_k + h), \tag{2.13}$$
$$\mathbf{s}_6 = hf(\mathbf{x}^{(k)} - \frac{8}{27}\mathbf{s}_1 + 2\mathbf{s}_2 - \frac{3544}{2565}\mathbf{s}_3 + \frac{1859}{4104}\mathbf{s}_4 - \frac{11}{40}\mathbf{s}_5). \tag{2.14}$$

Then the value of the solution at the given point is found as:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \frac{25}{216}\mathbf{s}_1 + \frac{1408}{2565}\mathbf{s}_3 + \frac{2197}{4104}\mathbf{s}_4 - \frac{1}{5}\mathbf{s}_5, \tag{2.15}$$

$$\mathbf{x}'^{(k+1)} = \mathbf{x}^{(k)} + \frac{16}{135}\mathbf{s}_1 + \frac{6656}{12825}\mathbf{s}_3 + \frac{28561}{56430}\mathbf{s}_4 - \frac{9}{50}\mathbf{s}_5 + \frac{2}{55}\mathbf{s}_6 \tag{2.16}$$

The expected error can be now calculated as:

$$R = \frac{1}{h}|\mathbf{x}'^{(k+1)} - \mathbf{x}^{(k+1)}|. \tag{2.17}$$

It is important to choose the time step size that is small enough to accurately approximate the solution, but not so small that the computation becomes too expensive. Runge-Kutta-Fehlberg [16] or RKF45 uses step size control techniques during the computation. The main difference between the Runge-Kutta 4(5) and Runge-Kutta-Fehlberg methods is that the former is a fixed-step method, while the latter is an adaptive step-size method that estimates the error to adjust the step size in each iteration. The main idea of Runge-Kutta-Fehlberg is to start with a moderate step size and then, depending on how large is the expected error, either keep or recalculate the solution. If the error estimate exceeds the tolerance, the step size is adjusted to reduce the error. The specific adjustment of the step can vary depending on the algorithm's implementation. If the expected error R is less than or equal to the chosen tolerance ε then the current solution is kept and the algorithm moves to the next step with the step size δh. The adjustment factor δ is calculated as:

$$\delta = (\frac{\varepsilon}{R})^p, \tag{2.18}$$

where p defines how significantly the step size is adjusted based on the error and is usually chosen to be between 0.25 and 0.9. In our case $p = 0.25$, which is a classical choice [16]. If the value of p is set to be larger, then the step size is changed more drastically.

The RKF45 method proves to be valuable in solving mildly stiff ODEs. These are cases where the solution undergoes relatively rapid changes in certain parts of the time interval and slower changes in others. Traditional fixed-step methods may not be efficient in such situations, as they often require small step sizes throughout the entire domain to ensure accuracy. In contrast, the adaptive step size of the RKF45 method enables larger steps in regions where the solution changes slowly and smaller steps in regions with rapid changes, resulting in more efficient computational processes.

2.2 Data Generation

The RKF45 method provides adaptive stepping, meaning that the number of points for the solution is changing with the change of the error tolerance value ε. Figure 2.1 represents the number of points for which RKF45 evaluated the solution depending on the tolerance value.

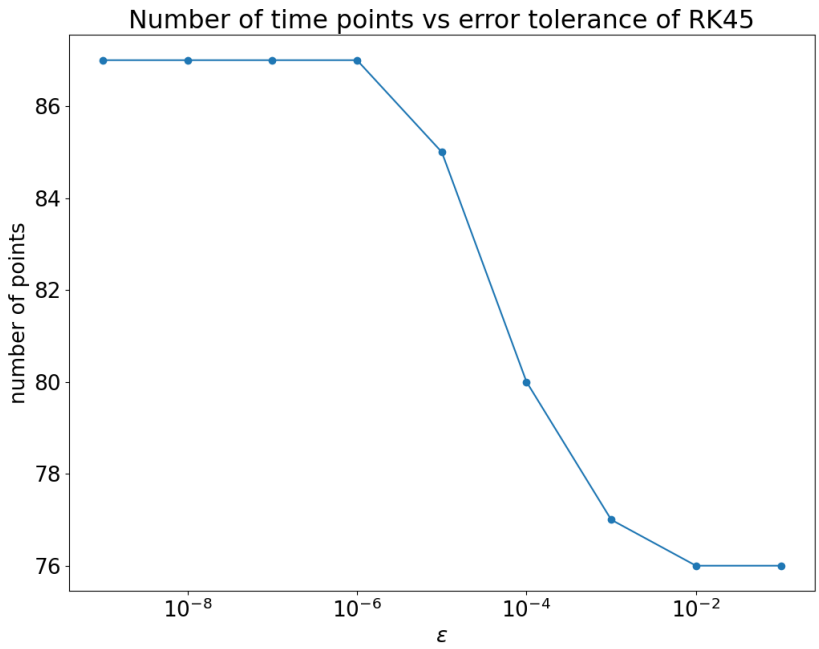

Figure 2.1. Number of points of the generated data depending on the error tolerance of RK45

In order to obtain an accurate solution during the data generation, the tolerance is fixed to be $\varepsilon = 10^{-6}$, since the number of solution points does not change after decreasing the error further. The RKF45 implementation given in [17] provides 86 data points when solving the system of ODEs with $\varepsilon = 10^{-6}$.

For data generation each ODE of the pre-defined system of ODEs for mechanism 1 is solved using the RKF45 method. As a result the data vector $\mathbf{x}$ for each chemical component is generated.

By taking initial values as $[\mathrm{A}]_0 = 1$, $[\mathrm{P}]_0 = 0$, $[\mathrm{cat}]_0 = 0.1$ and $[\mathrm{catA}]_0 = 0$, we generate the data trajectories for the chosen constants. The data trajectories generated for the period starting from $t_1 = 0$, $t_k = 100$ are presented in Fig. 2.2.

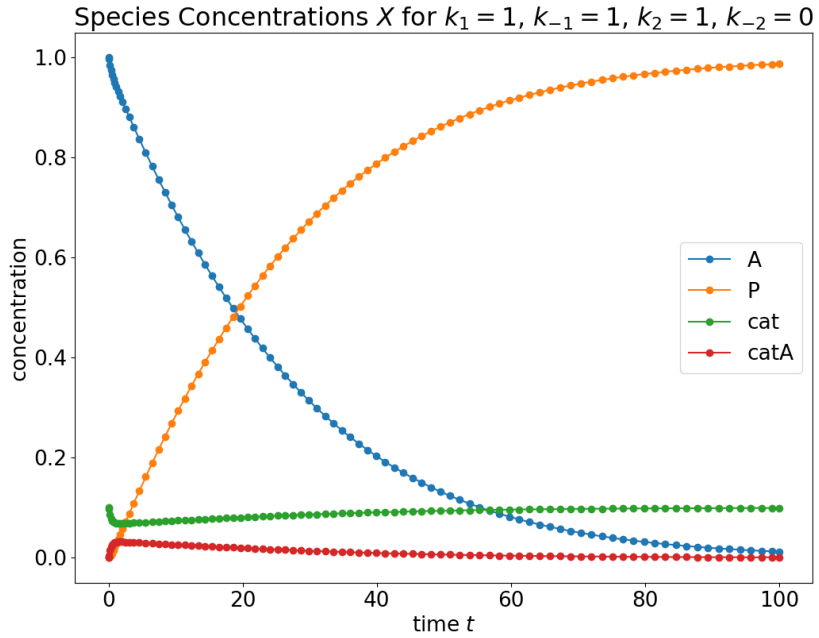

Figure 2.2. Data trajectories generated by RKF45

Research findings have revealed that employing multiple initial conditions proves to be more efficient in reconstructing mechanisms [18]. Consequently, we employ three sets of initial conditions with different starting concentration of intermediate $[\text{cat}]_0$. For multiple initial conditions experiments we use:

- $[\text{A}]_0 = 1$, $[\text{P}]_0 = 0$, $[\text{cat}]_0 = 0.1$, $[\text{catA}]_0 = 0$;
- $[\text{A}]_0 = 1$, $[\text{P}]_0 = 0$, $[\text{cat}]_0 = 0.15$, $[\text{catA}]_0 = 0$;
- $[\text{A}]_0 = 1$, $[\text{P}]_0 = 0$, $[\text{cat}]_0 = 0.05$, $[\text{catA}]_0 = 0$.

The obtained data matrices for ℓ sets of initial conditions are stacked on top of each other, resulting a matrix of size $[\ell k \times n]$

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_\ell \end{bmatrix}. \tag{2.19}$$

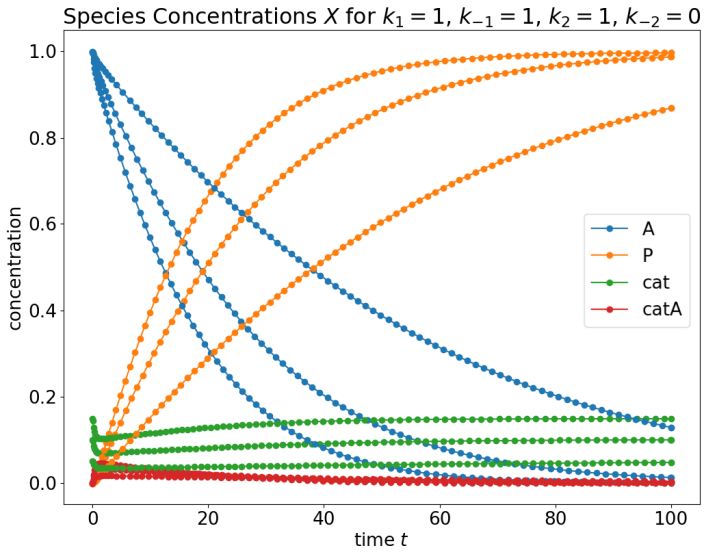

Figure 2.3. Data trajectories generated by RKF45 using three sets of initial conditions for $k_1 = 1$, $k_{-1} = 1$, $k_2 = 1$, $k_{-2} = 0$

The objective is to obtain the same set of ODEs or the set of ODEs with the coefficients close to those of (1.14, 1.15) from the generated time-series data using the methods described in the following chapters.

3 Numerical Differentiation

Various numerical differentiation methods can be applied to obtain $Y \approx \dot{X}$ from known values of X. One of the most straightforward approaches is finite difference approximation.

3.1 Polynomial Approximation

Derivatives of the concentrations can be approximated numerically. By knowing that the derivative $\frac{dx}{dt}$ corresponds to the rate of change of $x(t)$, let us consider the points $x(t_i)$ and $x(t_i + \Delta t)$, where $\Delta t = t_{i+1} - t_i$.

The gradient between these points can be approximated as:

$$\frac{dx}{dt} \approx \frac{x(t_i + \Delta t) - x(t_i)}{\Delta t}. \tag{3.1}$$

The above formula (or the forward difference approximation formula) can be derived by writing Taylor's series:

$$x(t + \Delta t) = x(t) + \Delta t \frac{dx}{dt}(t) + \frac{\Delta t^2}{2!}\frac{d^2x}{dt^2}(t) + \frac{\Delta t^3}{3!}\frac{d^3x}{dt^3}(t) + ... \tag{3.2}$$

By rearranging (3.2):

$$\frac{dx}{dt}(t) = \frac{x(t + \Delta t) - x(t)}{\Delta t} - \frac{\Delta t}{2!}\frac{d^2x}{dt^2}(t) + \frac{\Delta t^2}{3!}\frac{d^3x}{dt^3}(t) - \cdots \tag{3.3}$$

The error of this approximation results from truncating other terms of Taylor series.

Analogically we can write Taylor series for $x(t - \Delta t)$, $\Delta t = t_i - t_{i-1}$:

$$x(t - \Delta t) = x(t) - \Delta t \frac{dx}{dt}(t) + \frac{\Delta t^2}{2!}\frac{d^2x}{dt^2}(t) - \frac{\Delta t^3}{3!}\frac{d^3x}{dt^3}(t) + \cdots \tag{3.4}$$

The derivative $\frac{dx}{dt}(t)$ is found by rearranging the above series:

$$\frac{dx}{dt}(t) = \frac{x(t) - x(t - \Delta t)}{\Delta t} + \frac{\Delta t}{2!}\frac{d^2x}{dt^2}(t) - \frac{\Delta t^2}{3!}\frac{d^3x}{dt^3}(t) + ... \tag{3.5}$$

Truncating the above expression, we derive the backward difference approximation:

$$\frac{dx}{dt} \approx \frac{x(t_i) - x(t_i - \Delta t)}{\Delta t}. \tag{3.6}$$

Now for equally spaced data $\Delta t = t_{i+1} - t_i = t_i - t_{i-1}$ we can subtract (3.4) from (3.2), which leads to:

$$\frac{dx}{dt}(t) = \frac{x(t + \Delta t) - x(t - \Delta t)}{2\Delta t} - \frac{2\Delta t^2}{3!}\frac{d^3x}{dt^3}(t) + ... \tag{3.7}$$

The above expression can be truncated and used for a central difference approximation:

$$\frac{dx}{dt} \approx \frac{x(t_i + \Delta t) - x(t_i - \Delta t)}{2\Delta t}. \tag{3.8}$$

The central difference approximation can be used for finding the derivatives numerically for all data points except for the boundary ones. Derivatives at the first and last point cannot be approximated, but one may use the forward difference for the initial point and backward difference for the last point.

The problem is that backward and forward difference approximation is not as accurate as central difference. Central difference has an error of order $O(\Delta t^2)$, while the error of backward and forward difference is $O(\Delta t)$.

It is possible to consider the same problem from the polynomial point of view [19, Chapter 1]. The derivatives can be approximated numerically by fitting the Lagrange polynomial [20, Chapter 3] (using piecewise interpolation by Lagrange interpolating polynomials and differentiating them). The Lagrange basis polynomials are defined as:

$$L_i(t) = \prod_{j=1, j\neq i}^{k} \frac{t - t_j}{t_i - t_j}. \tag{3.9}$$

The Lagrange interpolation polynomial is then given as a sum of local polynomials:

$$p(t) = \sum_{i=1}^{k} x_i L_i(t). \tag{3.10}$$

When using linear interpolation, the polynomial is constructed using two neighbour points:

$$p(t) = x(t_{i-1})\frac{(t - t_i)}{(t_{i-1} - t_i)} + x(t_i)\frac{(t - t_{i-1})}{(t_i - t_{i-1})}. \tag{3.11}$$

Assuming that the evaluation point is between t_i and t_{i-1}. So now we can rewrite the

above formula as

$$p(t_i) = x(t_{i-1})\frac{(t_i - t_{i+1})}{(t_{i-1} - t_{i+1})} + x(t_{i+1})\frac{(t_i - t_{i-1})}{(t_{i+1} - t_{i-1})}. \tag{3.12}$$

The value of the polynomial $p(t_i)$ should follow the relation:

$$p(t_i) = x_i. \tag{3.13}$$

Thus, by using the known points x_{i-1} and x_{i+1} we construct a linear polynomial as:

$$p(t) = c_1 t + c_0, \tag{3.14}$$

such that

$$\begin{aligned} p(t_{i-1}) &= c_1 t_{i-1} + c_0 = x_{i-1} \\ p(t_{i+1}) &= c_1 t_{i+1} + c_0 = x_{i+1}. \end{aligned}$$

From the linear system above it is possible to find a slope of this polynomial:

$$p'(t_i) = \frac{x_{i+1} - x_{i-1}}{t_{i+1} - t_{i-1}}.$$

The slope can be found for any time-point by using the above expression. Thus, the approximated derivatives for the interior nodes can be expressed by the following matrix vector product:

$$\begin{bmatrix} ? \\ p'(t_2) \\ p'(t_3) \\ \vdots \\ p'(t_{k-1}) \\ ? \end{bmatrix} = \begin{bmatrix} ? & & & & & \\ \frac{-1}{t_3 - t_1} & 0 & \frac{+1}{t_3 - t_1} & & & \\ 0 & \frac{-1}{t_4 - t_2} & 0 & \frac{+1}{t_4 - t_2} & & \\ & & \ddots & \ddots & \ddots & \\ & & & \frac{-1}{t_k - t_{k-2}} & 0 & \frac{+1}{t_k - t_{k-2}} \\ & & & & & ? \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{k-1} \\ x_k. \end{bmatrix} \tag{3.15}$$

Linear interpolation uses two points for evaluating the approximation somewhere between those points. Thus, we can not evaluate the approximation at the initial point $t = t_1$ or final point $t = t_k$.

In order to find the derivative at the boundary points t_1 and t_k, it is possible to apply quadratic interpolation, in which the second-order polynomial goes through each of three points:

$$p(t) = x(t_{i-1})\frac{(t-t_i)(t-t_{i+1})}{(t_{i-1}-t_i)(t_{i-1}-ti+1)} + x(t_i)\frac{(t-t_{i-1})(t-t_{i+1})}{(t_i-t_{i-1})(t_i-t_{i+1})} - \\ -x(t_{i+1})\frac{(t-t_{i-1})(t-t_i)}{(t_{i+1}-t_{i-1})(t_{i+1}t_{i+1}-t_i)}. \tag{3.16}$$

Then the derivative $p'(x)$ can be expressed as:

$$p'(t) = x(t_{i-1})\frac{2t-t_i-t_{i+1}}{(t_{i-1}-t_i)(t_{i-1}-t_{i+1})} + x(t_i)\frac{2t-t_{i-1}-t_{i+1}}{(t_i-t_{i-1})(t_i-t_{i+1})} - \\ -x(t_{i+1})\frac{2t-t_{i-1}-t_i}{(t_{i+1}-t_{i-1})(t_{i+1}-t_i)}. \tag{3.17}$$

The usage of quadratic polynomial approximation provides the same order of accuracy for the derivative approximation at the boundary points as at the interior nodes.

3.2 Spline Interpolation

Another approach, which can be used to find the approximation of $\dot{X}$, is spline interpolation. A spline is defined as a numerical function, which is piece-wise-defined by a set of polynomials [21]. Spline interpolation method constructs a spline interpolant $s_i(t), i \in \{1, ..., k\}$, through the data points. This spline interpolant is usually a lower degree polynomial. So instead of fitting all of the points with one higher-degree polynomial, the method splits it to several intervals and fits lower polynomial spline on each interval. As an example one can refer to Fig. 3.1, which represents some curve with uniform knot spacing which is interpolated using three splines.

When using the cubic spline, for an interval $[t_{i-1}, t_i]$ we can write:

$$s_i(t) = a_i + b_i(t-t_i) + c_i(t-t_i)^2 + d_i(t-t_i)^3. \tag{3.18}$$

The key advantage of spline interpolation over the finite difference method is its ability to provide a smooth approximation. This is achieved because the spline approximation requires the following conditions:

$$s_i(t_{i-1}) = s_{i-1}(t_{i-1}), \tag{3.19}$$

$$s_i'(t_{i-1}) = s_{i-1}'(t_{i-1}), \tag{3.20}$$

$$s_i''(t_{i-1}) = s_{i-1}''(t_{i-1}). \tag{3.21}$$

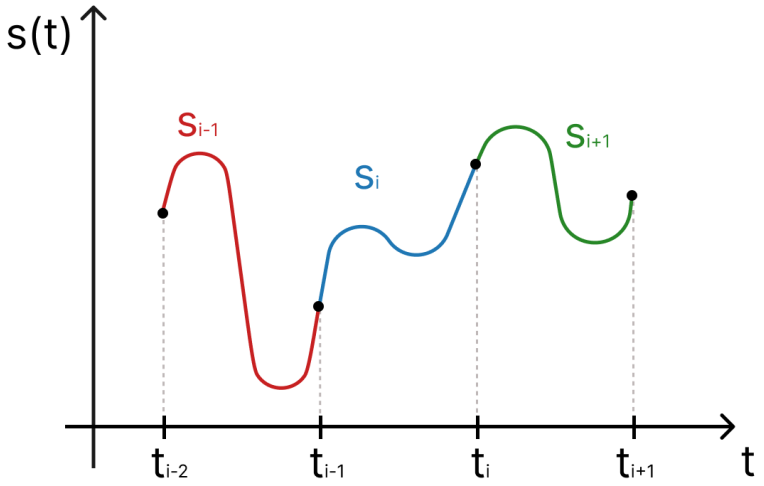

Figure 3.1. An interpolating spline

The above conditions mean that the approximated line will not have any breaks in the edges of the intervals, as it is twice continuously differentiable for each interval knots.

By using (3.18), we can find:

$$s_i(t_i) = a_i + b_i(t_i - t_i) + c_i(t_i - t_i)^2 + d_i(t_i - t_i)^3 = a_i. \tag{3.22}$$

The first derivative of the cubic polynomial at t_i is

$$s_i'(t_i) = b_i + 2c_i(t - t_i) + 3d_i(t - t_i)^2\big|_{t=t_i} = b_i. \tag{3.23}$$

Calculating the second and third derivatives, we can find that:

$$s_i''(t_i) = 2c_i \tag{3.24}$$

$$s_i''(t_i) = 6d_i. \tag{3.25}$$

We can find the derivatives of the neighbouring splines at t_{i-1}:

$$s_{i-1}''(t_{i-1}) = 2c_{i-1} \tag{3.26}$$

$$s_i''(t_{i-1}) = 2c_i + 6d_i(t_{i-1} - t_i). \tag{3.27}$$

By using the conditions for continuity, from (3.21):

$$2c_i - 6d_i(t_i - t_{i-1}) = 2c_{i-1}. \tag{3.28}$$

Then the coefficient d_i is found as:

$$d_i = \frac{c_i - c_{i-1}}{3(t_i - t_{i-1})}. \tag{3.29}$$

Defining the step $h_i = t_i - t_{i-1}$, we can rewrite is as:

$$d_i = \frac{c_i - c_{i-1}}{3h_i}. \tag{3.30}$$

By using the continuity conditions and knowing (3.30), it is possible to find another coefficient:

$$b_i = \frac{a_i - a_{i-1}}{h_i} + \frac{(2c_i + c_{i-1})h_i}{3}. \tag{3.31}$$

The last one for natural cubic spline is:

$$c_{i-1}h_i + 2c_i(h_i + h_{i+1}) + c_{i+1}h_{i+1} = 3(\frac{a_{i+1} - a_i}{h_{i+1}} - \frac{a_i - a_{i-1}}{h_i}), \tag{3.32}$$

where $c_k = s''(t_k) = 0$ and $s''(t_0) = c_1 - 3d_1h_1 = 0$.

If we use natural cubic spline with $c_0 = c_k = 0$, then the coefficients c_i can be found by solving a linear system with a tridiagonal matrix [22].

In the following simulations we will use cubic spline provided by UnivariateSpline class of scipy Python library.

3.3 Numerical Experiments

By knowing the system of ODEs and the set of coefficients, it is possible to calculate $\dot{X}$ directly, by substituting the vectors of X to (1.3). The obtained ground truth vectors of $\dot{X}$ for single set of initial conditions are plotted in Fig. 3.2.

The approximated values of $\dot{X}$ can be found without prior knowledge of the ODE system by either finite difference approximation or spline interpolation. Let Y_{FD} be the approximation found using finite difference method and Y_s be the approximation obtained using cubic univariate spline. The obtained approximations of derivatives are presented in Fig. 3.3.

To obtain the error between Y_{FD} and the ground truth $\dot{X}$ we can find the absolute and relative residual. The absolute residual is just the Frobenius norm of the difference between the approximation $Y$and ground truth $\dot{X}$:

$$\varepsilon_{abs} = \|Y - \dot{X}\|_F. \tag{3.33}$$

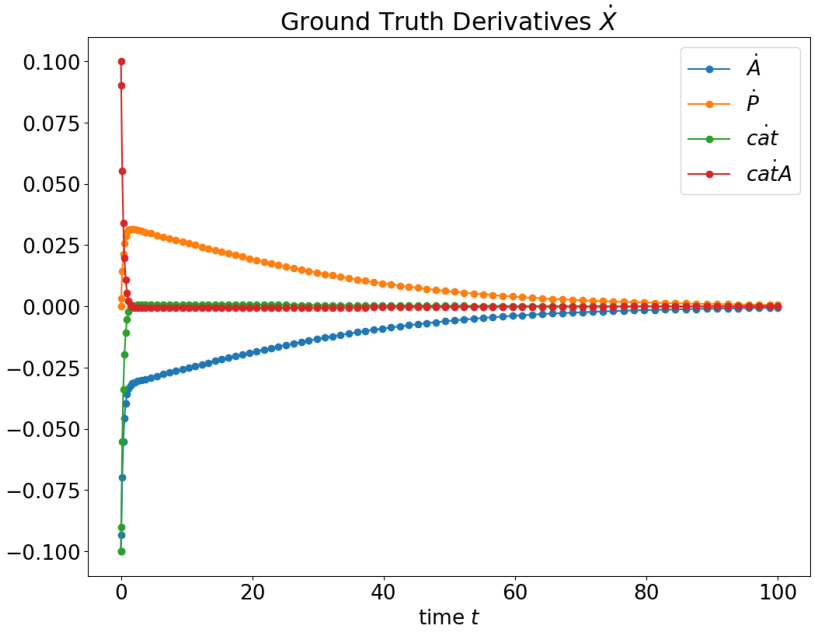

Figure 3.2. Ground Truth Derivatives $\dot{X}$ of Species Concentrations for mechanism A1r with $k_1 = 1$, $k_{-1} = 1$, $k_2 = 1.$ $k_{-2} = 0$, $[\text{cat}]_0 = 0.1$

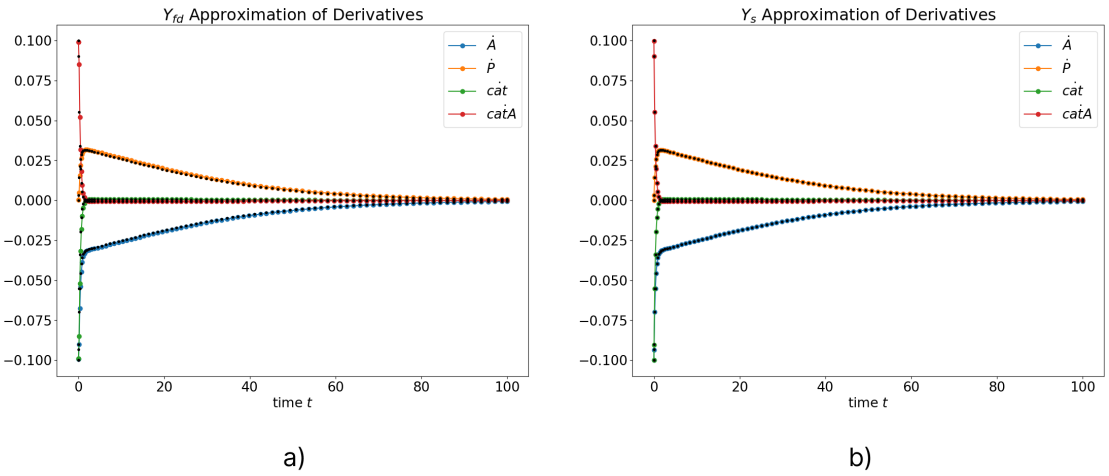

Figure 3.3. Approximation of derivatives Y of species concentrations for mechanism A1r with $k_1 = 1$, $k_{-1} = 1$, $k_2 = 1.$ $k_{-2} = 0$, $[\text{cat}]_0 = 0.1$: a) using finite difference; b) using cubic spline

The relative residual between the approximation and the ground truth vectors is found as :

$$\varepsilon_r = \frac{\|Y - \dot{X}\|_F}{\|\dot{X}\|_F}. \tag{3.34}$$

Optimizing the error in numerical differentiation can be achieved by carefully selecting appropriate time points. One straightforward approach is to use equispaced points, where the number of points matches those chosen by the RKF45 method during the data generation process. In this case, each trajectory is approximated using a total of 87 equispaced points. The results of this approximation can be seen in Table 3.1.

The table clearly demonstrates that spline approximation yields significantly more accurate results compared to finite difference approximation. The obtained results indicate that spline

Table 3.1. Derivatives approximation error at the equispaced points

Approximation	$[\text{cat}]_0 = 0.1$		$[\text{cat}]_0 = \{0.05, 0.1, 0.15\}$	
	ε_{abs}	ε_r	ε_{abs}	ε_r
Finite Difference	0.11356	0.53534	0.21324	0.56913
Cubic Spline	0.07880	0.38963	0.14838	0.41751

approximation provides greater stability due to the smooth and continuous nature of spline functions. In contrast, finite difference approximation is sensitive to the chosen step size and the location of the approximation points.

Another possible option is to use the same points, which were selected by RKF45. Inferring kinetics using Runge-Kutta methods for sparse identification of nonlinear dynamics (RK4-SINDy) [23] is claimed to be more efficient than SINDy [24] for noisy data. Thus, using the points selected by adaptive RKF45 algorithm is expected to be efficient. RKF45 selected $k = 87$ points to generate the data trajectories from $t_1 = 0$, $t_k = 100$. The results obtained by calculating the derivatives approximation Y using either polynomial or spline interpolation at the points, selected by RK45 are presented in Table 3.2.

Table 3.2. Derivatives approximation error at the points chosen by RKF45

Approximation	$[\text{cat}]_0 = 0.1$		$[\text{cat}]_0 = \{0.05, 0.1, 0.15\}$	
	ε_{abs}	ε_r	ε_{abs}	ε_r
Finite Difference	0.06277	0.16409	0.11647	0.16406
Cubic Spline	0.00027	0.00078	0.00047	0.00073

By comparing the results in Table 3.2 with those presented in Table 3.1, we can see the advantage of using the RKF45-chosen points. When using finite difference approximation the absolute error became roughly two times smaller and the relative error roughly three times smaller for RKF45-chosen points. The most drastic difference however can be noticed when using the univariate cubic spline approximation, since the errors became roughly a hundred times smaller.

The density of the points selected by the RKF45 method is higher in regions of the data trajectory where the most significant changes occur. In contrast, when equispaced points are used, the density remains uniform throughout the entire trajectory, regardless of whether the concentration remains relatively stable or undergoes drastic changes. As a result, the approximation of the derivative $Y$is less accurate when equispaced points are employed.

Apart from the RKF45-chosen points, it is also possible to use Chebyshev points:

$$t_i = \cos(\frac{(2i-1)\pi}{2k}), \quad i = 1, 2, \dots, k. \tag{3.35}$$

Chebyshev points are selected in a way that ensures the polynomial oscillates between the maximum and minimum distances from the function at each point. This characteristic of Chebyshev points results in the error being evenly distributed across the interval, rather than being concentrated at specific points.

Chebyshev points are chosen based on the roots of certain orthogonal polynomials, which have desirable numerical properties [25, Chapter 12]. Polynomial approximation using Chebyshev points is more stable than other types of points, meaning that small changes in the input data do not result in large changes in the approximation. Moreover, the convergence rate of polynomial approximation using Chebyshev points is faster than the convergence rate of the equispaced points or RKF45-chosen points [26].

The results obtained using Chebyshev points are presented in Table 3.3. It is seen that the results are a bit more accurate than when using RKF45-chosen points (see Table 3.2).

Table 3.3. Derivatives approximation error at Chebyshev points

Approximation	$[\text{cat}]_0 = 0.1$		$[\text{cat}]_0 = \{0.05, 0.1, 0.15\}$	
	ε_{abs}	ε_r	ε_{abs}	ε_r
Finite Difference	0.05936	0.15738	0.11144	0.16060
Cubic Spline	0.000131	0.00038	0.00025	0.00039

The advantage of using either RKF45 or Chebyshev points rather than equispaced points can be presented graphically (see Figure 3.4). Comparing the plots of the ground truth $\dot{X}$ (plotted with the black line) and the approximated derivatives at the different points, it can be seen that the approximation at the equidistant points is much worse than the ones using Chebyshev and RKF45-chosen points.

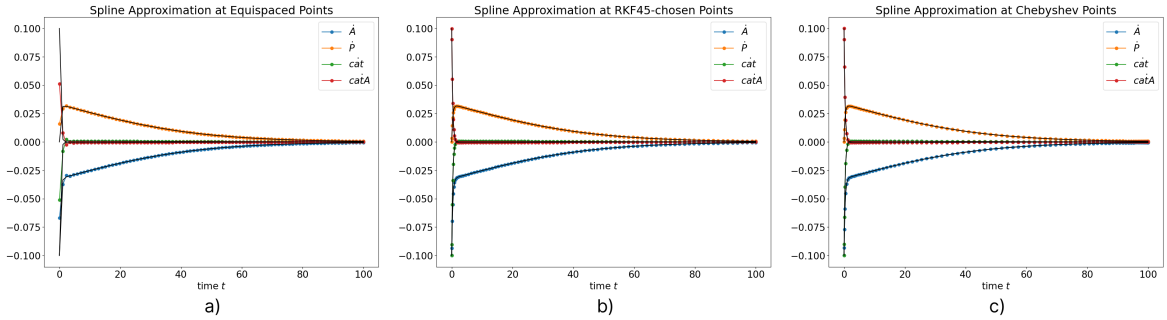

Figure 3.4. Spline approximation of the derivatives for mechanism A1r with $k_1 = 1$, $k_{-1} = 1$, $k_2 = 1$. $k_{-2} = 0$, $[\text{cat}]_0 = 0.1$ for: a) equispaced points; b) RKF45-chosen points; c) Chebyshev points

The numerical experiments clearly demonstrate that the most accurate results are obtained by employing spline approximation at Chebyshev points. Approximation at the points selected by adaptive RKF45 method also provides relatively good results, while using equidistant points is not a favorable choice for the specific case at hand.

4 Sparse Identification of Nonlinear Dynamics (SINDy)

The concept of extracting governing equations from measured data is not novel, as it plays a crucial role in modeling the nonlinear dynamics of a system. One of the existing methods, which is based on the least squares problem, is Sparse Identification of Nonlinear Dynamical Systems (SINDy) [24]. This method is capable of extracting models of dynamical systems from time series data.

4.1 SINDy Model and Regularization

The SINDy model assumes that the dynamics of a system can be expressed as a linear combination of basis functions, often chosen as nonlinear functions of the system variables. The resulting dynamical system model is typically represented as a system of ordinary differential equations (ODEs) or partial differential equations (PDEs) [27]. One of the main strengths of this method is that it not only produces a model that fits the time-series data but also returns interpretable equations that describe the system's behavior. These resulting ODEs usually contain only a small number of terms, making them more interpretable and easier to utilize for prediction and control purposes. In other words, SINDy provides a low-dimensional and sparse model that captures the underlying dynamics of the system. However, as we will observe, this does not necessarily imply that the system identified by SINDy is identical or closely related to the system that generated the data.

SINDy offers the capability to identify accurate and interpretable systems, which holds significant importance in various scientific and engineering domains. Its applicability extends to modeling biological systems [28], nonlinear optics systems [29], and many other domains. Furthermore, SINDy has also proven to be effective in modeling chemical reactions [30].

SINDy uses the mathematical formulation as described in Section 1.1 and is implemented in the pySINDy python library [31] [32]. In this study, the pySINDy package is employed to obtain the governing equations from the data of catalytic reactions.

The pySINDy library offers various optimizers, one of which is the sequentially thresholded least squares algorithm (STLSQ), accessible through the "pysindy.optimizers.stlsq" module. This optimizer finds the least-squares solution for coefficients W. It eliminates all the coefficients which are smaller than the chosen λ threshold value. The procedure is repeated recursively until there are no non-zero coefficients with an absolute value less than

the threshold value left. Further details on the STLSQ method can be found in Section 5.2

To promote sparsity in the SINDy model, a Ridge regularization term can be incorporated [33]. Ridge regression is claimed to be effective when there are near-linear relationships among the independent variables of the system. In the presence of such relationships, the data can be affected by multicollinearity, where variables are highly correlated with each other. Multicolliniarity in our case is caused by the model specification itself. Such source of multicollinearity was discussed in [34].

Ridge regression is a modified least squares regression with added ℓ_2-norm regularization added. In order to tune the regularization, λ parameter is introduced:

$$w_j = \underset{\hat{w}_j}{\operatorname{argmin}} \|y_j - \widetilde{X}\hat{w}_j\|_2^2 + \lambda\|\hat{w}_j\|_2^2, \tag{4.1}$$

where λ is the Ridge regularization parameter on the weight vector. By tuning this parameter it is possible to control the desired sparsity of the solution.

Ridge regression, in general, does not enforce coefficients in the matrix W to be exactly zero. Instead, it shrinks the coefficients towards zero, resulting in relatively small values. This characteristic can be leveraged to promote sparsity in the solution.

When using pySINDy model, the differentiation for approximating $\dot{X}$ is obtained using smoothed finite difference method, which is a built-in numerical differentiation method in pySINDy library [24].

4.2 SINDy Experimental Results

It has already been discovered, that the sparsity of the solution provided by SINDy can be achieved by using iterative thresholding method called STLSQ or by adding a ℓ_2-norm regularization.

We can evaluate the performance of the SINDy model by calculating the number of non-zero terms in the extracted system of ODEs (sparsity s) and the error between the obtained matrix of coefficients W and the exact one W_{ex}:

$$\varepsilon_r(W) = \frac{\|W - W_{\text{ex}}\|_F}{\|W_{\text{ex}}\|_F}. \tag{4.2}$$

By trying different values of the thresholding parameter α and Ridge regularization coefficient λ, it is seen how the sparsity of the obtained solution is changed (see Table 4.1).

Based on the results presented in Table 4.1, it is evident that adjusting the thresholding and regularization parameters has an impact on the level of sparsity, while the error remains constant for the obtained weight matrix. Furthermore, the error value is relatively high, reach-

Table 4.1. SINDy results for $k_1 = 1$, $k_{-1} = 1$, $k_2 = 1$, $k_{-2} = 0$, $[\text{cat}]_0 = [0.1]$

α	λ	$\varepsilon_r(W)$	s
0.01	0.005	0.6746	49
0.01	0.05	0.6746	24
0.01	0.1	0.6746	25
0.05	0.01	0.6746	11
0.1	0.01	0.6746	10
0.1	0.1	0.6746	0

ing 0.6746. This suggests that although the SINDy algorithm can promote sparsity, it may struggle to accurately extract the correct active components of the system of ODEs.

The regularization paths can represent graphically how tuning the thresholding parameter α or Ridge regularization parameter λ changes the obtained solution and which components are extracted for the right-hand side of ODEs by SINDy. Plotting the regularization path is an efficient method to estimate the efficiency of the linear models with convex penalties [35].

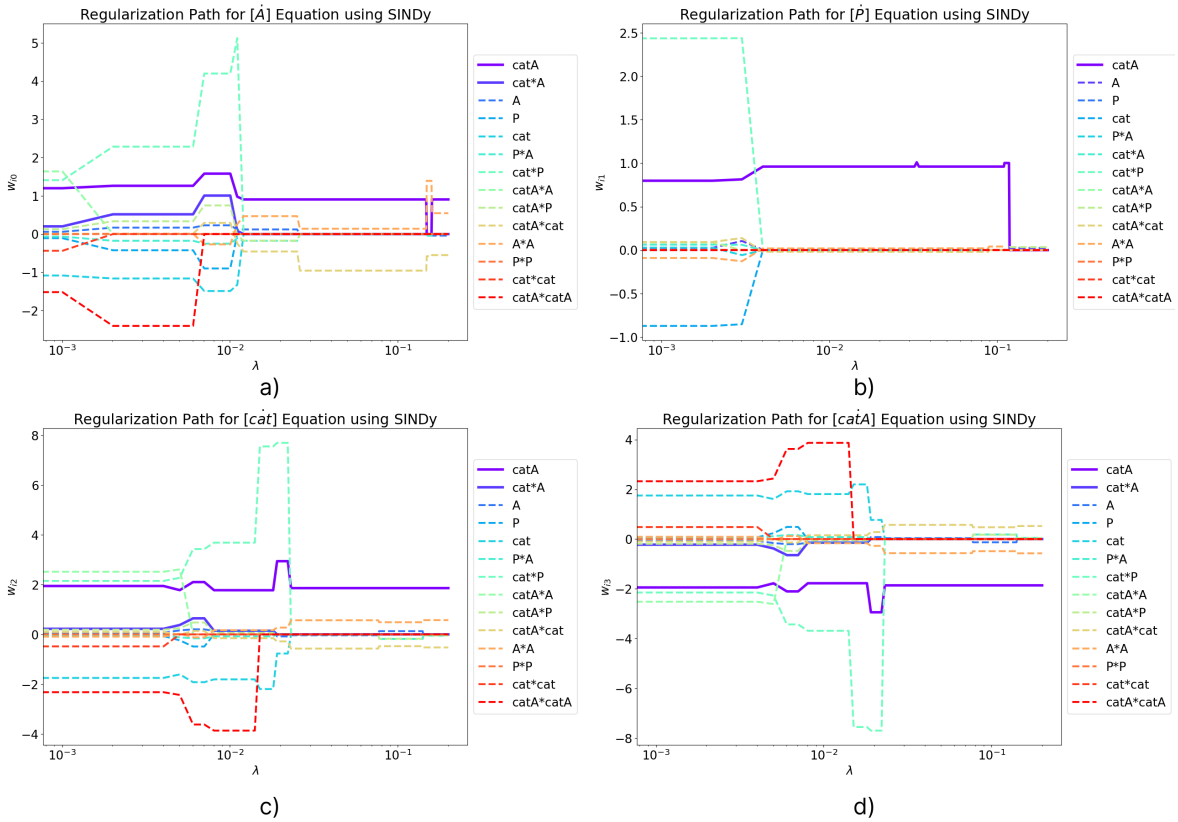

Figure 4.1. Regularization paths for regularization parameter λ in SINDy: a) $[\dot{A}]$; b) $[\dot{P}]$; c) $[\dot{cat}]$; d) $[\dot{catA}]$

By trying different values of λ with the fixed value $\alpha = 0.05$, regularization paths are obtained for each ODE of the system (see Fig. 4.1). Regularization paths show how the coefficients are changed with the change of the parameter. The dashed lines represent the components of the feature matrix $\widetilde{X}$, which should have zero coefficients according to the original set of equations (1.14) with the chosen coefficients. The solid lines represent the components of $\widetilde{X}$, which should have non-zero coefficients.

If the value of λ is fixed, we can vary α and see how it affects the results. The regularization paths for fixed $\lambda = 0.015$ and different α values are presented in Fig. 4.2.

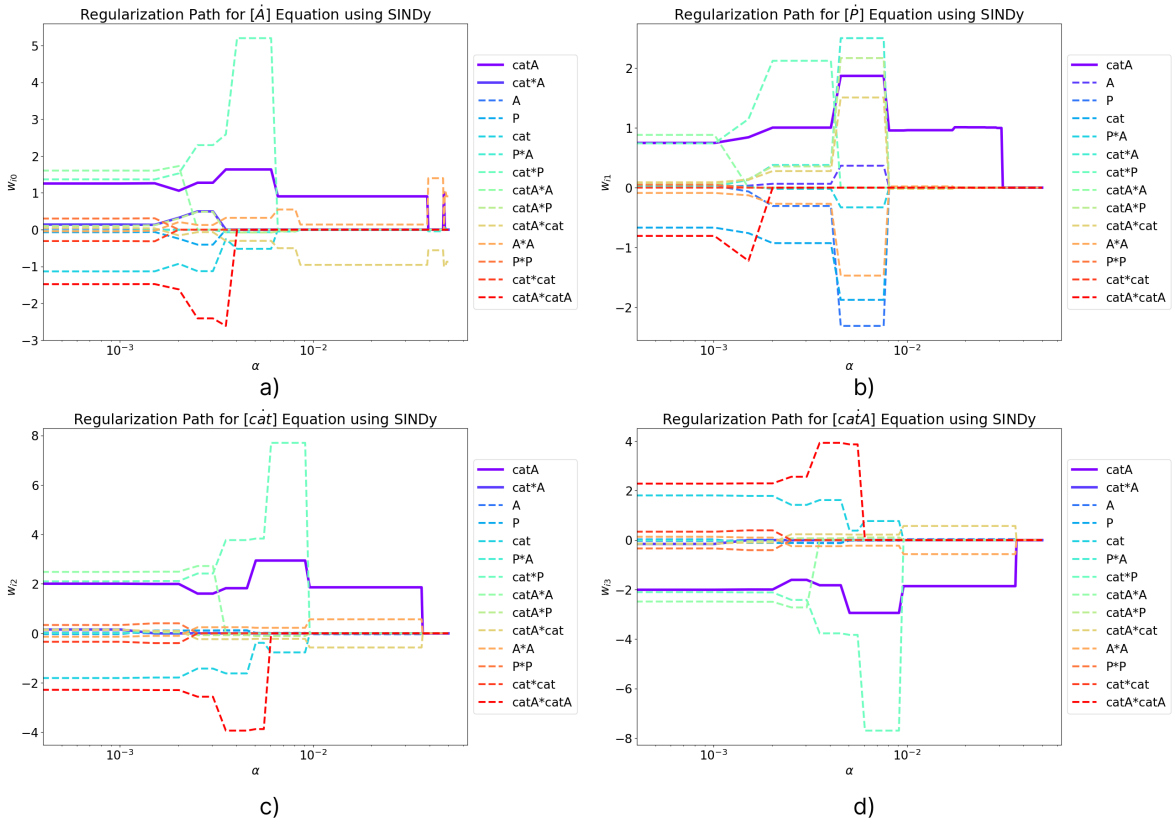

Figure 4.2. Regularization paths for thresholding parameter α in SINDy: a) $[\dot{A}]$; b) $[\dot{P}]$; c) $[\dot{cat}]$; d) $[\dot{catA}]$

It is seen from Fig. 4.1 that pySINDy with STLSQ and ℓ_2 regularizer definitely proposes sparsity of the solution, however it fails to identify the correct non-zero components. The method does not eliminate the coefficients of the components which should be zero properly for small values of λ.

It can be concluded, that the model fails to identify correct components with non-zero coefficients. From the regularization paths plotted it is hard to see how each of the parameters affects the results. The obtained system of ODEs can be written for each chosen pair of threshold and regularisation parameters. For instance, by choosing $\alpha = 0.01$ and $\lambda = 0.05$, the following system of ODEs is extracted from the data by SINDy:

$$\begin{aligned} [\dot{A}] &= 1.01[\text{catA}] - 1.00[\text{A}][\text{cat}] \\ [\dot{P}] &= 1.01[\text{catA}] - 0.01[\text{catA}][\text{cat}] - 0.01[\text{A}][\text{A}] \\ [\dot{cat}] &= 2.03[\text{catA}] - 1.01[\text{catA}][\text{cat}] + 0.02[\text{A}][\text{A}] \\ [\dot{catA}] &= -2.03[\text{catA}] + 1.01[\text{catA}][\text{cat}] + 0.02[\text{A}][\text{A}]. \end{aligned} \tag{4.3}$$

It is easy to compare the above system of ODEs with the original system of ODEs for the chosen kinetic constants ($k_1 = 1$, $k_{-1} = 1$, $k_2 = 1$. $k_{-2} = 0$), which is:

$$[\dot{A}] = [\text{catA}] - [\text{A}][\text{cat}] \tag{4.4}$$

$$
\begin{aligned}
[\dot{\mathrm{P}}] &= [\mathrm{catA}] \\
[\dot{\mathrm{cat}}] &= 2[\mathrm{catA}] - [\mathrm{A}][\mathrm{cat}] \\
[\dot{\mathrm{catA}}] &= -2[\mathrm{catA}] + [\mathrm{A}][\mathrm{cat}].
\end{aligned}
$$

Despite the fact that pySINDy provides sparse solutions, it is seen that it fails to extract the exact non-zero parameters. For instance, for the third equation of the system instead of having $[\mathrm{A}][\mathrm{cat}]$ the result of pySINDy has $[\mathrm{catA}][\mathrm{cat}]$. Due to that, it would be impossible to define the type of mechanism correctly by the system of ODEs (4.3) extracted by pySINDy.

In the next chapter thresholding and regularization techniques are implemented without the pySINDy library, to make their usage more understandable and flexible.

5 Regularized Least Squares Solution

5.1 Unregularized Least Squares

In our case the matrix $\widetilde{X}$ is a tall skinny matrix of size $[k \times m]$, where $k > m$. Since there are more data points than the number of parameters needed to be estimated, the system $\widetilde{X}w = y$ is overdetermined. Thus, it is not possible to find an exact unique solution w that satisfies all the equations. That is why methods like least squares approximation can be used to find the best-fitting solution that minimizes the overall error between the equations and the unknowns.

Solving a least squares problem is a standard approach for regression problems. The basic idea behind least squares approximation is to minimize the sum of the squared differences between the observed values and the values predicted by the system of equations. This is done by adjusting the values of the unknowns to find the best-fit solution. The least squares approach is widely used for various problems and the analysis of kinetic data is not the exception [36].

The least squares method minimizes the Frobenius norm between Y and the product $\widetilde{X}W$:

$$W = \underset{\hat{W}}{\operatorname{argmin}} \|Y - \widetilde{X}\hat{W}\|_F. \tag{5.1}$$

This is equivalent to minimizing the 2-norm of the difference between the column y_j and $\widetilde{X}w_j$ corresponding to each ODE:

$$w_j = \underset{\hat{w}}{\operatorname{argmin}} \|y_j - \widetilde{X}\hat{w}_j\|_2. \tag{5.2}$$

We can perform a singular value decomposition of the feature matrix $\widetilde{X} = U\Sigma V^T$, where $U \in \mathbb{R}^{k\times k}$, $V \in \mathbb{R}^{m\times m}$ are orthogonal and $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_{\min(m,k)}) \in \mathbb{R}^{k\times m}$. The minimum norm least squares solution is then found as:

$$w_{j,i} = \sum_{i=1}^{r} (\frac{u_i^T y_i}{\sigma_i}) v_i, \tag{5.3}$$

where $r = \text{rank}(\widetilde{X})$.

The scipy Python library [17] is applied to the data concentrations for solving the least squares problem. The least squares solver obtains the coefficients matrix W, which is used to

restore the equations. By solving the obtained equations using RKF45 it is possible to find the new concentrations matrix $\widehat{X}$. So, the new $\widehat{X}$ are the concentrations that fit the solution found by the solver. The relative error between $\widehat{X}$ and X is computed as:

$$\varepsilon_r(\widehat{X}) = \frac{\|\widehat{X} - X\|_F}{\|X\|_F}. \tag{5.4}$$

By forwarding $\widehat{X}$ to the system of the obtained equations, we can find $\hat{Y}$. The analogical metrics can be used for finding the error between $\hat{Y}$ and the approximated Y found either by spline approximation (Y_{S}) or by finite difference approximation (Y_{FD}):

$$\varepsilon'_r(\hat{Y}) = \frac{\|\hat{Y} - Y\|_F}{\|Y\|_F}. \tag{5.5}$$

Analogically, we can compare $\hat{Y}$, which fits the least squares solution, with the ground truth $\dot{X}$, which had been calculated previously numerically by forwarding X to the original system of ODES. The error is calculated as follows:

$$\varepsilon_r(\hat{Y}) = \frac{\|\hat{Y} - \dot{X}\|_F}{\|\dot{X}\|_F}. \tag{5.6}$$

Obviously the error $\varepsilon_r(\hat{Y})$ obtained for ground truth derivatives is expected to be greater than $\varepsilon'_r(\hat{Y})$ obtained for the approximated ones, because there is the additional error occuring during the numerical approximation (see Chapter 3.3).

The obtained matrix of coefficients W can be compared to the exact matrix of coefficients W_{ex} (see (1.13)). The error then is found as:

$$\varepsilon_r(W) = \frac{\|W - W_{\text{ex}}\|_F}{\|W_{\text{ex}}\|_F}. \tag{5.7}$$

Unregularized least squares solver does not provide sparse solution. In most cases for mechanism A1r unregularized least squares provide $[14 \times 4]$ matrix W with 56 non-zero elements (see Appendix). In some cases, however, the coefficients are small enough to obtain a more sparse matrix. For instance, when using kinetic coefficients $k_1 = 1$, $k_{-1} = 1$, $k_2 = 1$, $k_{-2} = 0$ for three initial conditions with $Y = Y_s$ obtained using spline approximation, unregularized least squares provides matrix W with sparsity $s = 48$ (see Table 5.1).

Overall, it can be observed that the unregularized least squares method provides a satisfactory fit to the data. When using equispaced points and setting $Y = Y_{\text{S}}$, it is observed that the method achieves the best fit with X, resulting in a relative error of $\varepsilon_r(\widehat{X}) = 0.0001$. However, there is a notable discrepancy between the obtained coefficients matrix W and the exact coefficients matrix W_{ex}, indicating a lack of sparsity in the solution. As a result, the

Table 5.1. Unregularized least squares results for $k_1 = 1$, $k_{-1} = 1$, $k_2 = 1$, $k_{-2} = 0$, $[\text{cat}]_0 = 0.1$

Sampling	Approximation	$\varepsilon_r'(\widehat{Y})$	$\varepsilon_r(\widehat{Y})$	$\varepsilon_r(\widehat{X})$	$\varepsilon_r(W)$	s
RKF45	finite difference	0.2774	0.1025	0.0128	0.9933	56
RKF45	spline	0.0008	0.0001	0.000002	1.0297	56
Equispaced	finite difference	0.4745	0.0229	0.0107	0.9995	56
Equispaced	spline	0.3216	0.0171	0.0043	0.9978	56
Chebyshev	finite difference	0.2641	0.0926	0.0140	0.9932	56
Chebyshev	spline	0.0004	0.0001	0.000003	1.0241	56

obtained system of ODEs is not interpretable and cannot be used for kinetic analysis. In order to solve this issue several methods for promoting sparsity are discussed in the next sections.

5.2 Sequentially Thresholded Least Squares

Iterative thresholding methods are claimed to be quite efficient for finding an optimal sparse approximation [37]. A simple iterative thresholding algorithm named sequentially thresholded least squares (STLSQ) is used in [24] for efficient extraction of governing equations for Lorenz system.

The algorithm iteratively computes the least squares solution while comparing the entries of the weight matrix W column w_j with the threshold value α. If a coefficient $w_{j,i}$ is found to be smaller than α, it is set to zero, and only the indices of the non-zero coefficients are stored. The least squares solution is then recalculated using the remaining indices. This iterative process continues until convergence is achieved for the non-zero coefficients. The convergence properties of this algorithm have been studied by Zhang and Shaeffer [38].

Algorithm 1 Sequentially Thresholded Least Squares

$n \leftarrow$ number of species
$m \leftarrow 2n + C_n^2$
$[k \times n]$ $Y \leftarrow$ derivatives approximation
$[k \times m]$ $\widetilde{X} \leftarrow$ feature matrix
$\alpha \leftarrow$ threshold parameter
$[m \times n]$ W

Input: $\widetilde{X}$, Y, α
Output: W

for i **from** 1 **to** n **do**
  integer $[1 \times m]$ $idx \leftarrow [0, 1, 2, ..., m-1]$
  $w := \text{argmin}_{\hat{w}} \|Y[:, i] - \widetilde{X}\hat{w}\|_2$
  while len(idx) $\neq 0$ **do**
    for j **from** 1 **to** m **do**:
      if $|w[j]| < \alpha$ **then**
        $idx := idx$.remove(j)
        $w := \text{argmin}_{\hat{w}} \|Y[:, i] - \widetilde{X}[:, idx]\hat{w}\|_2$
      end if
    end for
  end while
  $W[i, idx] = w$
end for

By manipulating the parameter α, the sparsity of the weights matrix W can be adjusted.

Increasing the value of α leads to a sparser solution. To determine the optimal value of the thresholding parameter α, regularization paths can be plotted and analyzed.

Tuning the value of α, however, may not always affect the solution. For instance, by observing the regularization paths for equispaced sampled solution with finite difference approximation (see Fig. 5.1), it is seen that the sparsity of the solution does not change and the algorithm provides the same coefficients W no matter which value of α is chosen. There is a slight change noticed in the regularization path for the second equation of the system (see Fig. 5.1(b)), however it is still not significant.

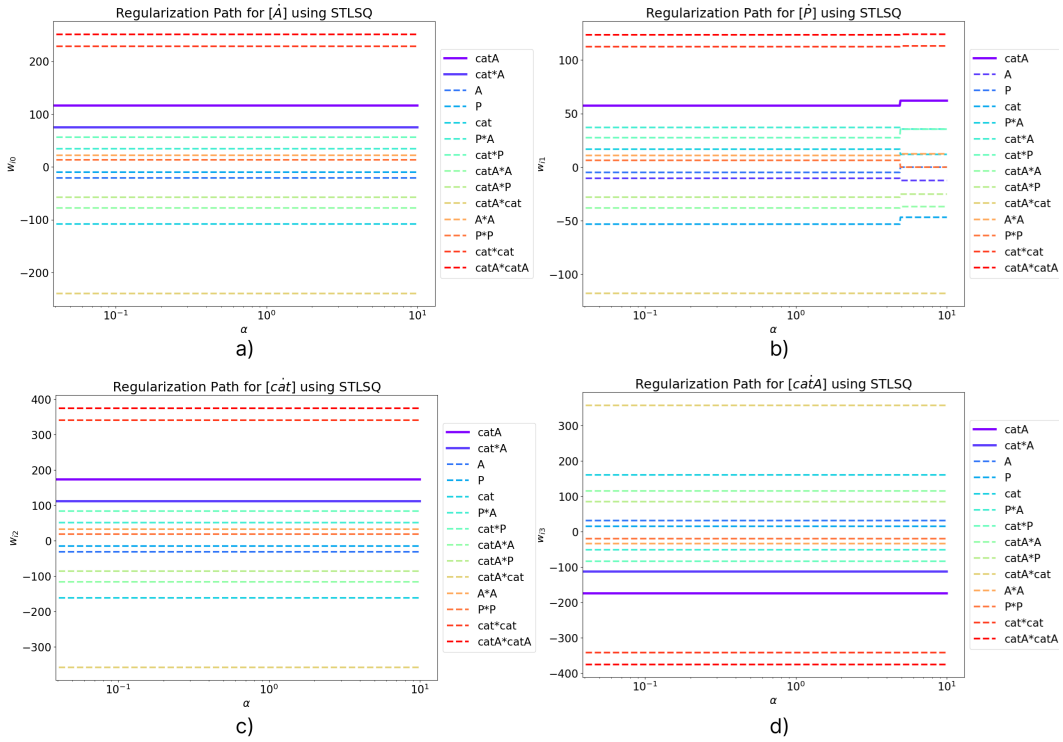

Figure 5.1. Regularization paths for STLSQ (equispaced points, finite difference approximation, $k_{-2} = 0$): a) [Ȧ] equation; b) [Ṗ] equation; c) [ċat] equation; d) [ċatȦ] equation

Changing the approximation method to that of spline approximation, does not provide sufficiently good results either, however it is seen from the regularization paths presented by Fig. 5.2 that changing α affects the final result.

A good example of the regularization paths can be observed when the equispaced sampling is replaced with points adaptively selected by RKF45. The regularization paths plotted for RKF45 sampling with spline approximation are depicted in Fig. 5.3.

Fig. 5.3 represents regularization paths for four ODEs of the system, showing the active components which should have non-zero coefficients with solid lines and others with dashed lines. Whenever there is a region, where active component's coefficients are non-zero and the dashed lines are zero, we mark it with green window, saying it is a "good" region,

It is seen from the regularization paths that the best result is achieved for $\alpha \approx 0.4$. The results obtained using this threshold with different sampling and approximation methods are presented in Table 5.2.

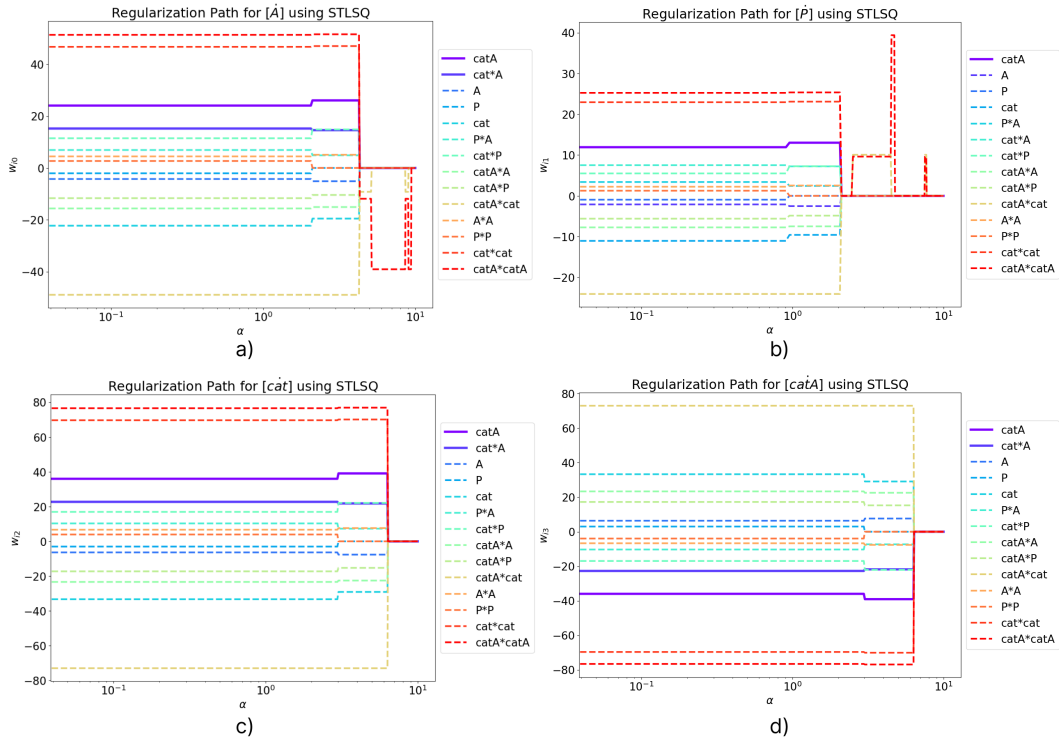

Figure 5.2. Regularization paths for STLSQ (equispaced points, spline approximation, $k_{-2} = 0$): a) [Ȧ] equation; b) [Ṗ] equation; c) [ċat] equation; d) [ċatA] equation

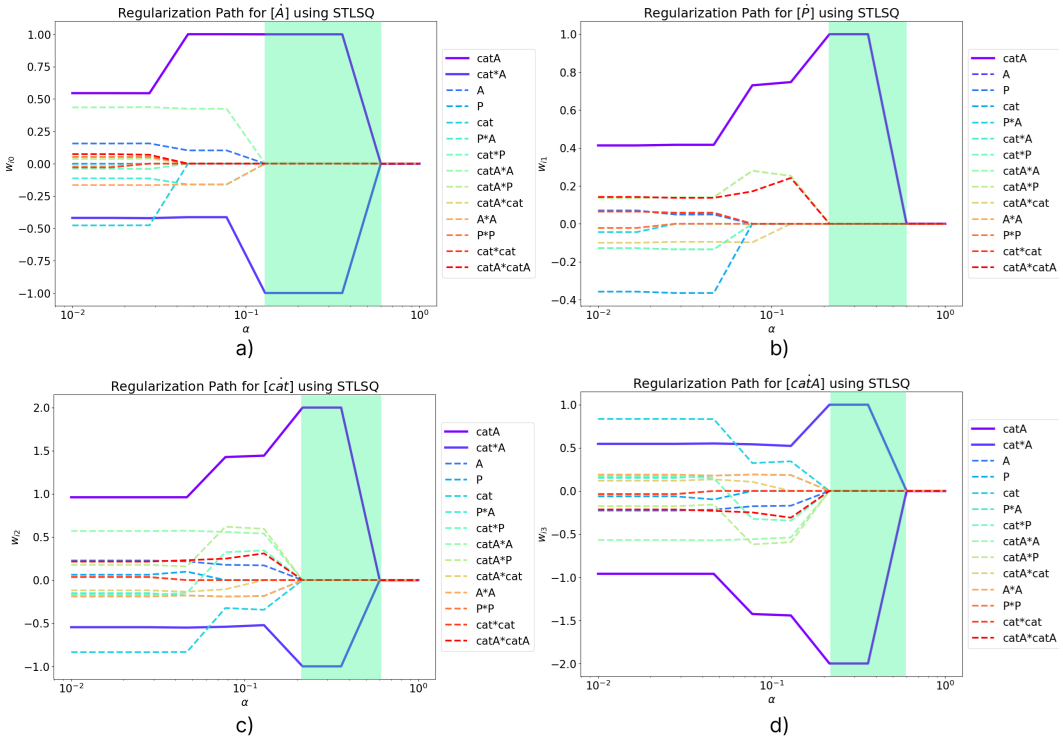

Figure 5.3. Regularization paths for STLSQ (RKF45 points, spline approximation, $k_{-2} = 0$): a) [Ȧ] equation; b) [Ṗ] equation; c) [ċat] equation; d) [ċatA] equation

By observing Table 5.2, it can be concluded that STLSQ yields distinct outcomes for different sampling and approximation methods when a fixed thresholding parameter is employed. When using equispaced sampling STLSQ provides completely non-sparse solution with $s = 56$.

The sampling chosen by RKF45 achieves $s = 54$ when using the finite difference approxi-

Table 5.2. STLSQ results for $k_1 = 1$, $k_{-1} = 1$, $k_2 = 1$, $k_{-2} = 0$, $[\text{cat}]_0 = 0.1$

Sampling	Approximation	α	$\varepsilon'_r(\hat{Y})$	$\varepsilon_r(\hat{Y})$	$\varepsilon_r(\widehat{X})$	$\varepsilon_r(W)$	s
Equispaced	finite difference	α = 0.4	0.4762	0.0267	0.0149	0.9995	56
Equispaced	spline	α = 0.4	0.3224	0.0163	0.0002	0.9978	56
RKF45	finite difference	α = 0.4	0.2774	0.1025	0.0128	0.9932	54
RKF45	spline	α = 0.5	0.0368	0.0368	1.4108	0.7562	5
RKF45	spline	α = 0.4	0.0008	0.0001	**0.000008**	**0.0003**	**7**
Chebyshev	finite difference	α = 0.4	1.0269	0.8246	0.1891	0.9931	54
Chebyshev	spline	α = 0.4	0.7986	0.7984	0.1814	**0.00004**	**7**

mation of the derivatives. The errors $\varepsilon_r(\widehat{X})$ and $\varepsilon_r(\widehat{W})$ do not significantly change compared to the equispaced sampling with finite difference approximation. However, when the derivatives are calculated using spline approximation, RKF45 sampling achieves almost zero $\varepsilon_r(\widehat{X})$ and the obtained weights matrix W is very close to the exact one, since $\varepsilon_r(\widehat{W}) = 0.0003$. The obtained system of ODEs for this case is:

$$\begin{aligned} [\dot{\text{A}}] &= 0.999568[\text{catA}] - 0.999759[\text{cat}][\text{A}] \\ [\dot{\text{P}}] &= 1.00004[\text{catA}] \\ [\dot{\text{cat}}] &= 1.999384[\text{catA}] - 0.999652[\text{cat}][\text{A}] \\ [\dot{\text{catA}}] &= -1.999384[\text{catA}] + 0.999652[\text{cat}][\text{A}]. \end{aligned} \tag{5.8}$$

It is evident that the method exhibits high sensitivity to errors. It demonstrates excellent accuracy when utilizing spline approximation, however, when employing finite difference approximation, the method struggles to identify the correct active components of the right-hand side of the system of ODEs. This observation is further supported by the analysis of the regularization paths shown in Fig. 5.4.

It is seen from Fig. 5.4 that when using finite difference approximation of the derivatives, which is less accurate than spline approximation, STLSQ fails to identify correct components with nonzero coefficients, no matter which value of α is chosen.

The same trend holds for Chebyshev sampling. The regularization paths for spline approximation case have good regions for all four equations (see. Fig. 5.5), meaning that the method correctly identifies the active components.

When using Chebyshev points and spline approximation of derivatives, STLSQ extracts the following system of the governing equations:

$$\begin{aligned} [\dot{\text{A}}] &= 1.000032[\text{catA}] - 1.000002[\text{cat}][\text{A}] \\ [\dot{\text{P}}] &= 1.000013[\text{catA}] \end{aligned} \tag{5.9}$$

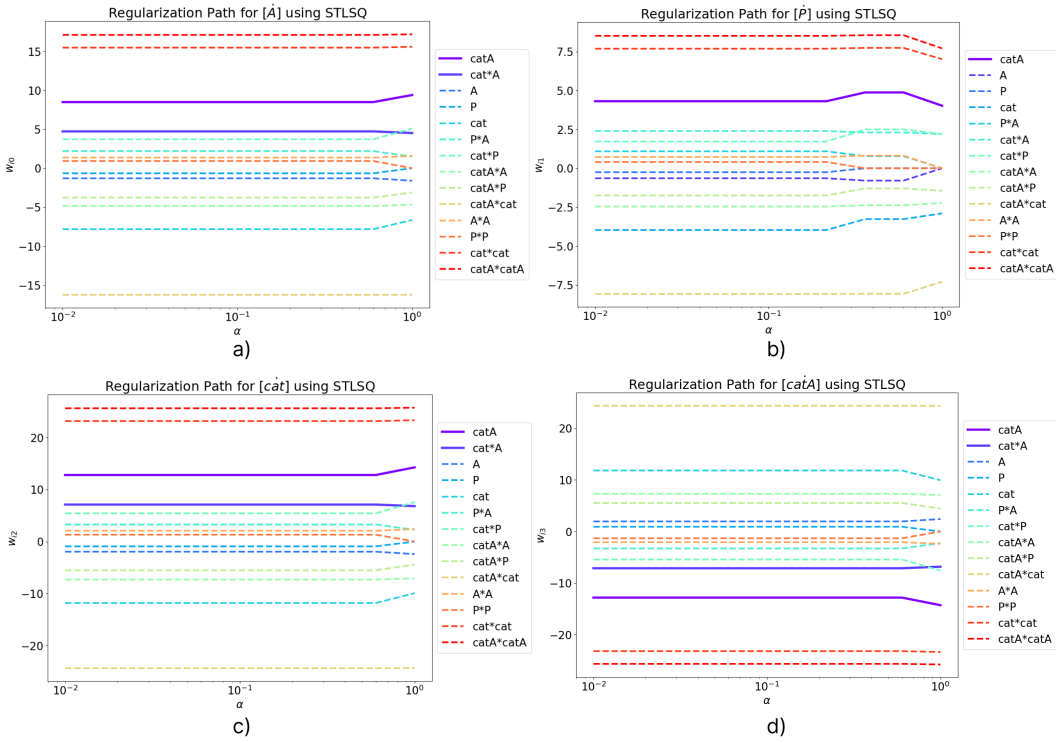

Figure 5.4. Regularization paths for STLSQ (RKF45 points, finite difference approximation, $k_{-2} = 0$): a) $[\dot{A}]$ equation; b) $[\dot{P}]$ equation; c) $[\dot{cat}]$ equation; d) $[\dot{catA}]$ equation

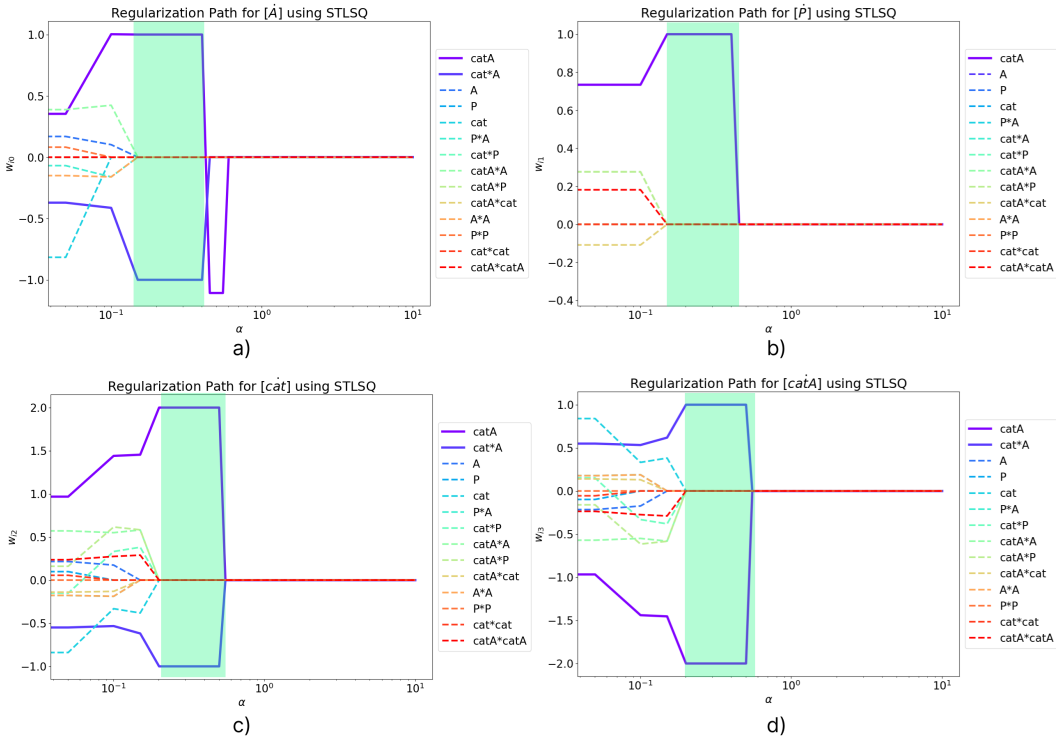

Figure 5.5. Regularization paths for STLSQ (Chebyshev points, spline approximation, $k_{-2} = 0$): a) $[\dot{A}]$ equation; b) $[\dot{P}]$ equation; c) $[\dot{cat}]$ equation; d) $[\dot{catA}]$ equation

$$
\begin{aligned}
[\dot{cat}] &= 2.000051[catA] - 1.000005[cat][A] \\
[\dot{catA}] &= -2.0000515[catA] + 1.000005[cat][A].
\end{aligned}
$$

It is seen that the obtained ODEs are very close to the exact ones for $k_1 = 1$, $k_{-1} = 1$,

$k_2 = 1, k_{-2} = 0$. The error between the obtained weights and the exact ones is now $\varepsilon_r(\widehat{W}) = 0.00004$, which is even smaller than it was for RKF45 sampling.

Based on this specific case, it can be anticipated that STLSQ is a promising method for inferring kinetics, offering a sparse and accurate solution. However, to attain such favorable outcomes, it is crucial to identify the optimal thresholding parameter and select appropriate data sampling and numerical approximation methods. When employing Chebyshev sampling with the same thresholding parameter α, but utilizing a different approximation method, the resulting solution is non-sparse ($s = 54$), and the relative error is $\varepsilon_r(\widehat{W}) = 0.9931$. Upon examining the regularization paths (refer to Fig. 5.6), it becomes evident that the method is highly sensitive to noise. In the case of finite difference approximation, the method failed to accurately identify the correct active components in all four equations.

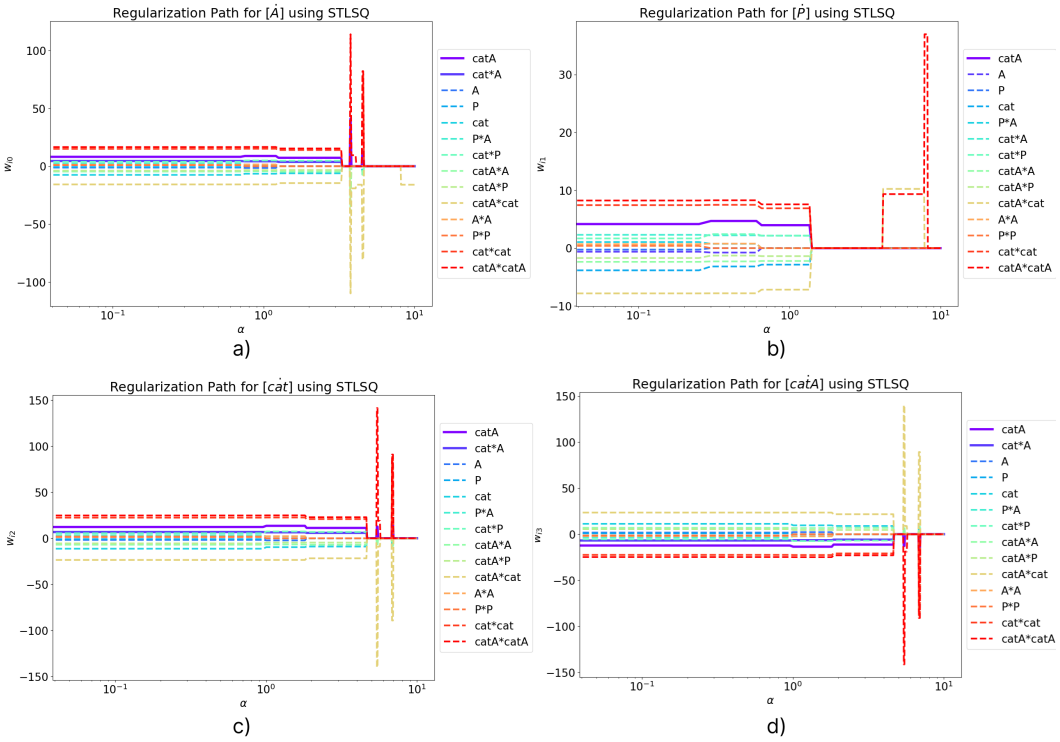

Figure 5.6. Regularization paths for STLSQ (Chebyshev points, finite difference approximation, $k_{-2} = 0$): a) [Ȧ] equation; b) [Ṗ] equation; c) [ċat] equation; d) [ċatA] equation

The results for STLSQ using three sets of initial conditions are presented in Appendix A, and it can be observed that they are quite similar to those obtained using a single set of initial conditions. In the case where the kinetic constant k_{-2} is set to 1 (see Appendix B), STLSQ once again demonstrates better results when using either Chebyshev or RKF45-chosen points. However, the accuracy of the results for $k_{-2} = 1$ is slightly diminished due to the increased complexity of the system.

In summary, the experimental results indicate that STLSQ is capable of providing accurate and sparse solutions, achieving coefficients of the matrix with an error of only $\varepsilon_r(\widehat{W}) = 0.00004$ (when using Chebyshev sampling with spline approximation). However, the main limitation of the method is its sensitivity to noise. When less accurate approximations of the derivatives are employed, it fails to identify interpretable solutions.

5.3 Lasso Regularization

Lasso, short for "least absolute shrinkage and selection operator," was introduced by R. Tibshirani in the middle of 1990s [39]. The Lasso regularization technique incorporates a penalty term into the loss function of a model, which discourages large coefficient values and encourages them to approach zero. This regularization approach promotes the selection of the most significant features while disregarding irrelevant ones.

The ℓ_1 penalty used in Lasso regularization has the effect of forcing some of the model coefficients to become exactly zero, thereby performing feature selection as well as regularization. This makes Lasso regularization particularly useful in situations where there are many features, some of which may be irrelevant or redundant. Lasso regularization can be tuned by adjusting the strength of the penalty term, which controls the amount of regularization applied. Lasso regularization term can be added to the least squares for proposing sparsity.

Lasso regularization minimizes the residual sum of squares subject to the sum of the absolute value of the coefficients being less than some parameter λ:

$$w = \underset{\hat{w}}{\operatorname{argmin}} \|y - \widetilde{X}\hat{w}\|_2^2 + \lambda\|\hat{w}\|_1 \tag{5.10}$$

When the parameter λ is set to zero, the algorithm behaves the same as an unregularized least squares solver, described earlier (see Section 5.1). By choosing a small value for λ, it becomes possible to achieve a sparse yet accurate solution. For our implementation of Lasso, we utilized the CVXPY Python library [40], [41], which proved to be more efficient for convex optimization problems compared to the scipy library [17].

Again, the regularization paths can be plotted in order to investigate the appropriate values of λ. For now we just observe the values manually and choose the best one, however it is also possible to implement a predictor-corrector method for adaptive λ selection in Lasso [42].

Using Lasso regularization with equispaced sampling does not yield efficient results. When employing finite difference approximation of $\dot{X}$ to evaluate the solution, the model fails to correctly identify the active components of the reaction, as can be seen from Fig. 5.7. The model extracts incorrect components from the right-hand side of the ODEs. For instance, when using $\lambda = 3 \cdot 10^{-3}$, we can receive quite sparse system of ODEs, which is:

$$\begin{aligned}
[\dot{\mathrm{A}}] &= -0.03[\mathrm{A}] - 0.01[\mathrm{A}][\mathrm{A}] \\
[\dot{\mathrm{P}}] &= 0.03[\mathrm{A}] + 0.03[\mathrm{P}][\mathrm{A}] \\
[\dot{\mathrm{cat}}] &= 0.02[\mathrm{P}][\mathrm{A}] - 0.02[\mathrm{A}][\mathrm{A}] \\
[\dot{\mathrm{catA}}] &= -0.02[\mathrm{P}][\mathrm{A}] - 0.02[\mathrm{A}][\mathrm{A}].
\end{aligned} \tag{5.11}$$

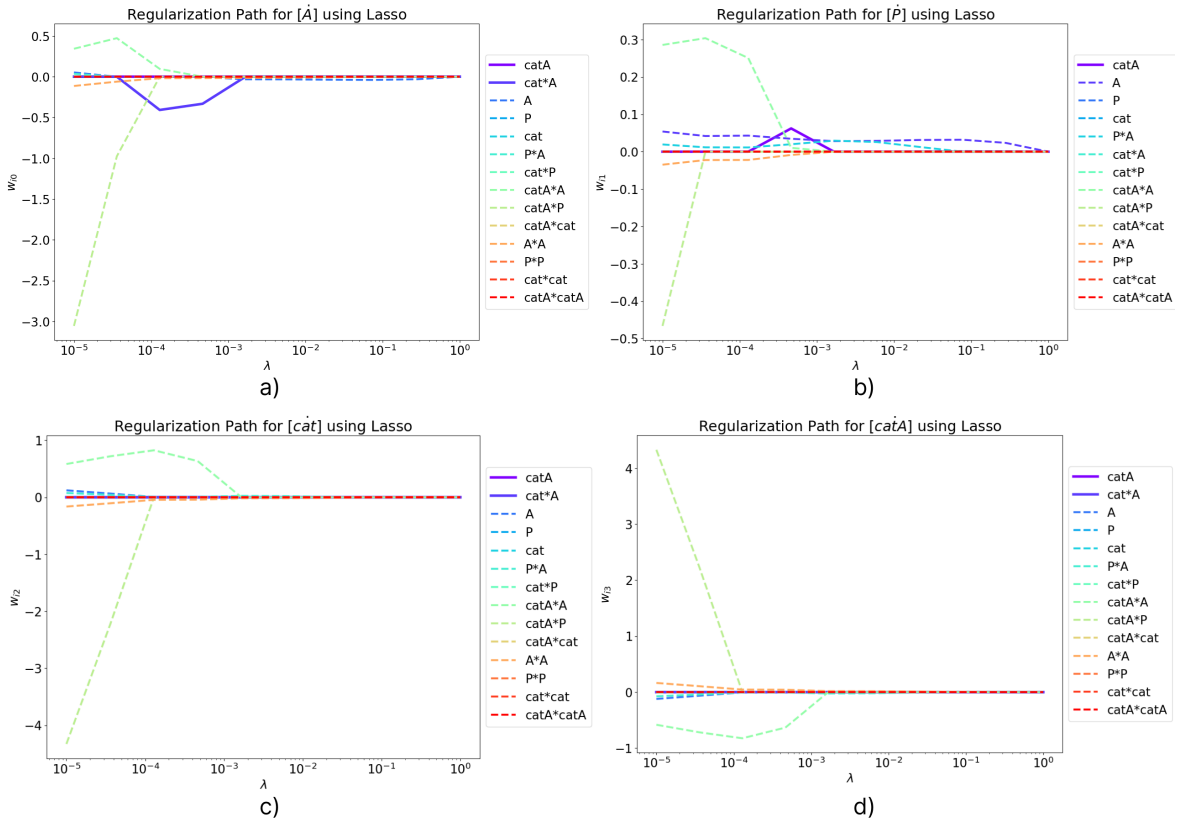

Figure 5.7. Regularization paths for Lasso regression (equispaced points, FD approximation, $k_{-2} = 0$): a) $[\dot{A}]$; b) $[\dot{P}]$; c) $[c\dot{a}t]$; d) $[ca\dot{t}A]$

Despite the fact that the extracted system of ODEs is sparse with $s = 8$, it is completely different from the exact one (1.14), since the active components of right-hand side of the equations are wrong.

The same trend persists when using spline approximation for equispaced sampling (see Fig. 5.8). Despite the fact that using spline approximation significantly improves the appearance of the regularization paths, it still mistakenly chooses certain components to be active. For instance, in the first equation $[\text{catA}][\text{A}]$ is chosen to have non-zero coefficients, while it should be $[\text{catA}]$ (see Fig. 5.8(a)).

The solution is noticeably improved by using the points adaprively selected by the RKF45 instead of equispaced points. This improvement is evident from the regularization paths shown in Fig. 5.9. Even when using finite different approximation, which introduces a relatively large error when approximating $\dot{X}$, the method still provides quite good results. All of the equations exhibit a range of λ values that correctly identify the active components. However, it is observed that the range of λ values is relatively small in the case of finite difference approximation.

It is possible to get a wider range of appropriate λ values when using spline approximation for RKF45 selected points. Spline approximation usage prevents high error at the derivatives approximation step and provides better results. A perfect example of the regularization paths is presented by Fig. 5.10.

It is evident from the plots that the model accurately identifies the active components of

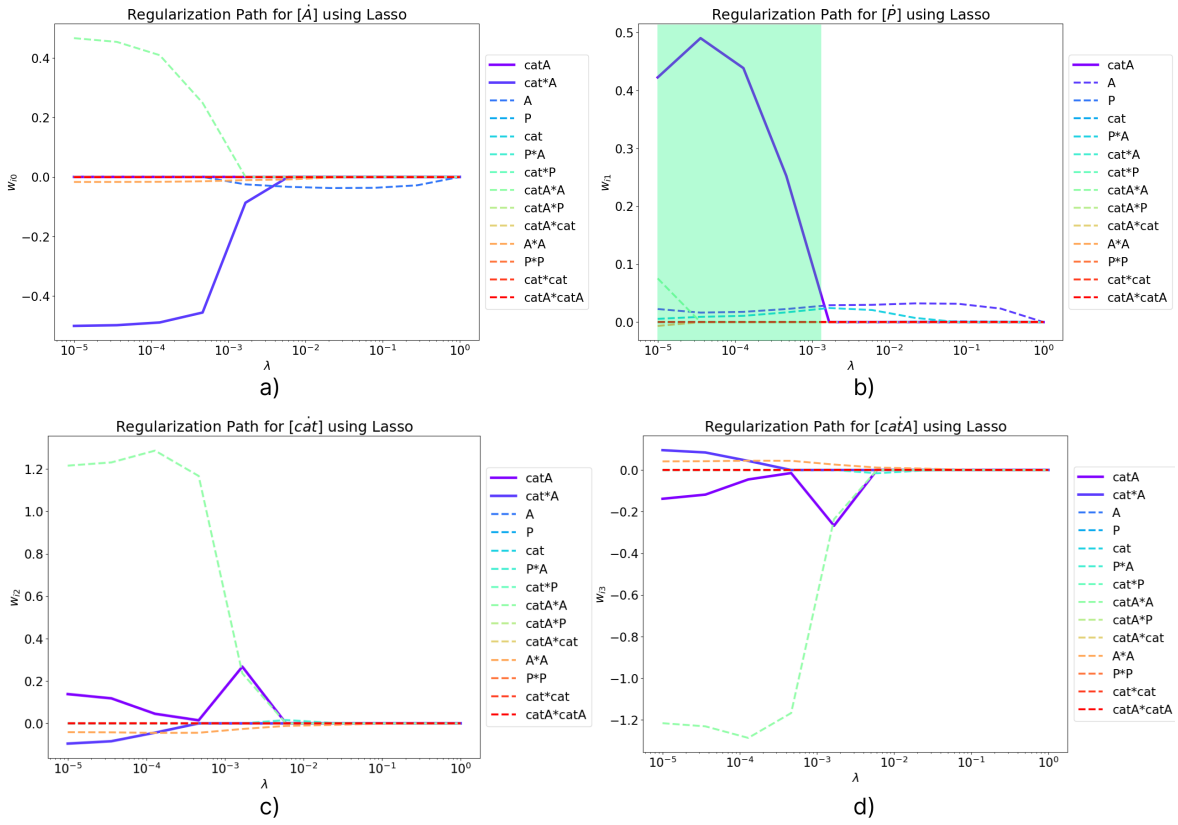

Figure 5.8. Regularization paths for Lasso regression (equispaced points, FD approximation, $k_{-2} = 0$): a) [$\dot{A}$]; b) [$\dot{P}$]; c) [$\dot{cat}$]; d) [$\dot{catA}$]

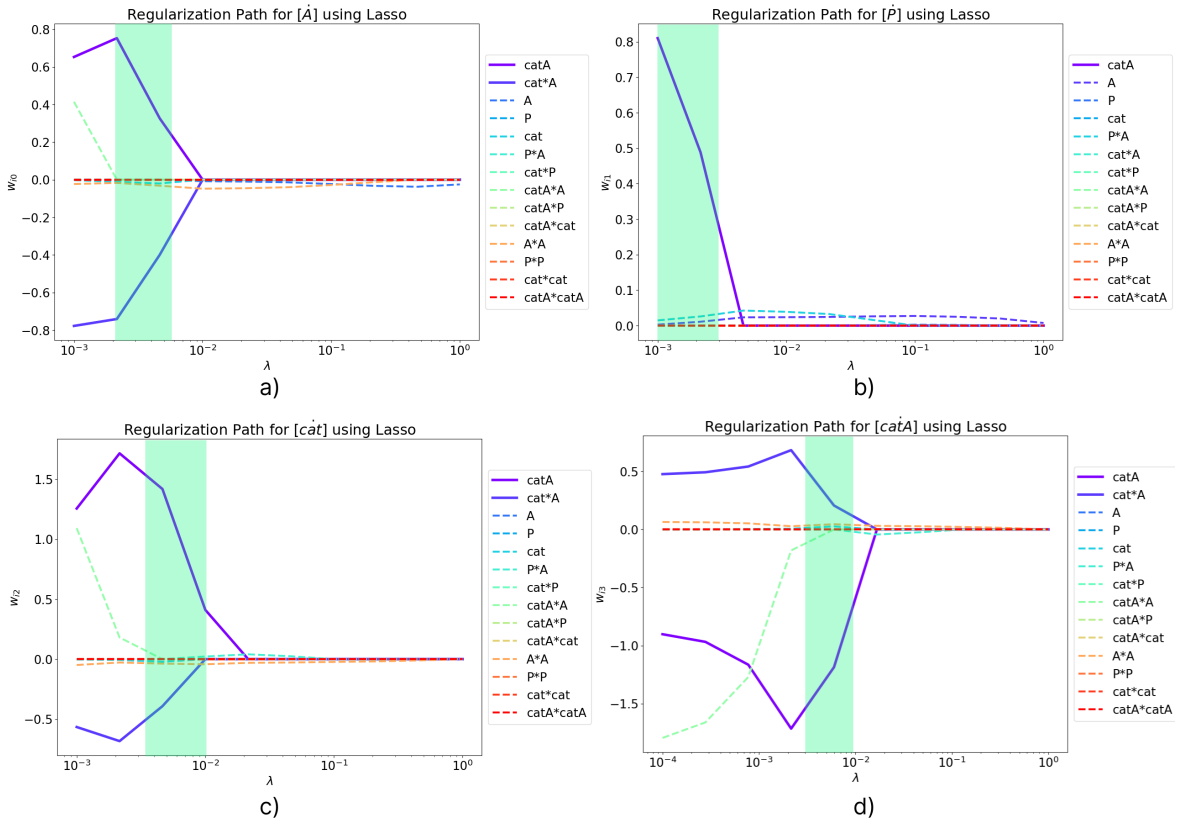

Figure 5.9. Regularization paths for Lasso regression (RKF45 points, FD approximation, $k_{-2} = 0$): a) [$\dot{A}$]; b) [$\dot{P}$]; c) [$\dot{cat}$]; d) [$\dot{catA}$]

the right-hand side of the equations starting from a value of $\lambda \approx 3 \cdot 10^{-2}$. Additionally, for $\lambda = 10^{-4}$, the coefficients closely resemble those of the exact system of ODEs (1.14). By

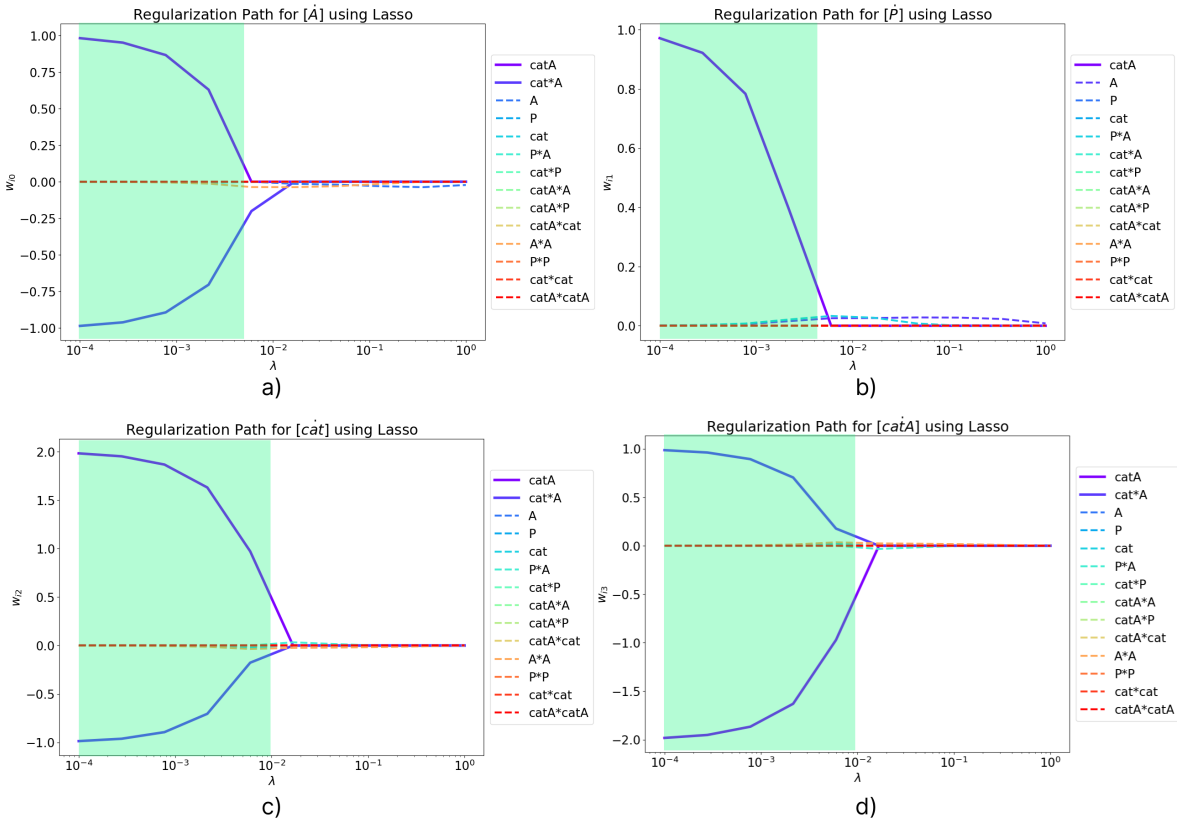

Figure 5.10. Regularization paths for Lasso regression (RKF45 points, spline approximation, $k_{-2} = 0$): a) [Ȧ]; b) [Ṗ]; c) [ċat]; d) [ċatA]

using $\lambda = 10^{-4}$ Lasso regression obtains the following governing equations:

$$\begin{aligned} [\dot{\text{A}}] &= 0.98[\text{catA}] - 0.99[\text{cat}][\text{A}] \\ [\dot{\text{P}}] &= 0.97[\text{catA}] \\ [\dot{\text{cat}}] &= 1.98[\text{catA}] - 0.99[\text{cat}][\text{A}] \\ [\dot{\text{catA}}] &= -1.98[\text{catA}] + 0.99[\text{cat}][\text{A}]. \end{aligned} \tag{5.12}$$

If the λ value is decreased to $\lambda = 10^{-5}$, the system of the equations is:

$$\begin{aligned} [\dot{\text{A}}] &= 1.00[\text{catA}] - 1.00[\text{cat}][\text{A}] \\ [\dot{\text{P}}] &= 1.00[\text{catA}] \\ [\dot{\text{cat}}] &= 2.00[\text{catA}] - 1.00[\text{cat}][\text{A}] \\ [\dot{\text{catA}}] &= -2.00[\text{catA}] + 1.00[\text{cat}][\text{A}]. \end{aligned} \tag{5.13}$$

In this case the error between for concentrations fit is $\varepsilon_r(\widehat{X}) = 0.00005$, and for the matrix of coefficients $\varepsilon_r(W) = 0.00164$.

Apart from using equispaced sampling and the points adaptively chosen by RKF45, we can also test the Lasso solver using Chebyshev points. However, when Chebyshev sampling is combined with finite difference approximation, it does not yield efficient results, as evidenced by the obtained regularization paths shown in Fig. 5.11.

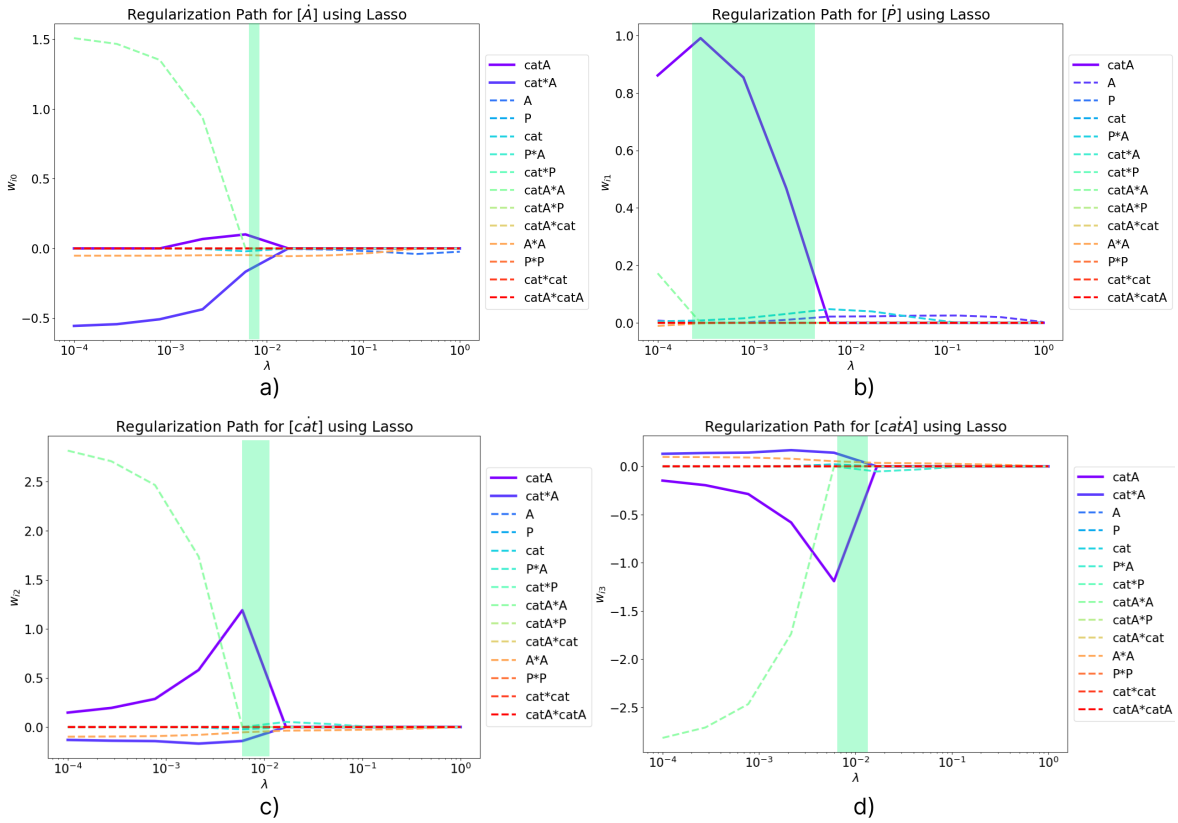

Figure 5.11. Regularization paths for Lasso regression (Chebyshev points, FD approximation, $k_{-2} = 0$): a) [$\dot{A}$]; b) [$\dot{P}$]; c) [$\dot{cat}$]; d) [$\dot{catA}$]

By comparing the obtained regularization paths with those plotted for RKF45-chosen points with finite difference approximation (see Fig. 5.9), it is evident that using Chebyshev points with finite difference approximation is not a good option, as the results worsen compared to the RKF45 case.

However when changing the approximation type to spline method, the results, again, are significantly improved. Comparing the regularization paths for Chebyshev points with spline approximation in Fig. 5.12 with those presented in Fig. 5.10 for RKF45 points, it can be said that the results are almost the same.

The extracted ODEs for $\lambda = 10^{-5}$ are the same for Chebyshev points and for RKF45-chosen points (5.14). The error for the concentrations fit however is slightly higher for Chebyshev points, with this error equal to $\varepsilon_r(\widehat{X}) = 0.00006$. The error for the coefficients is a bit smaller, reaching $\varepsilon_r(W) = 0.00132$. However the difference between the results obtained for Chebyshev points and for RKF45 points is negligible.

The results obtained during the experiments for equispaced sampling, RKF45-chosen points and Chebyshev points are presented in Table 5.3. The table represent the results for different approximation types (finite difference approximation and spline approximation) and different λ values, in order to see how the change affects the final result.

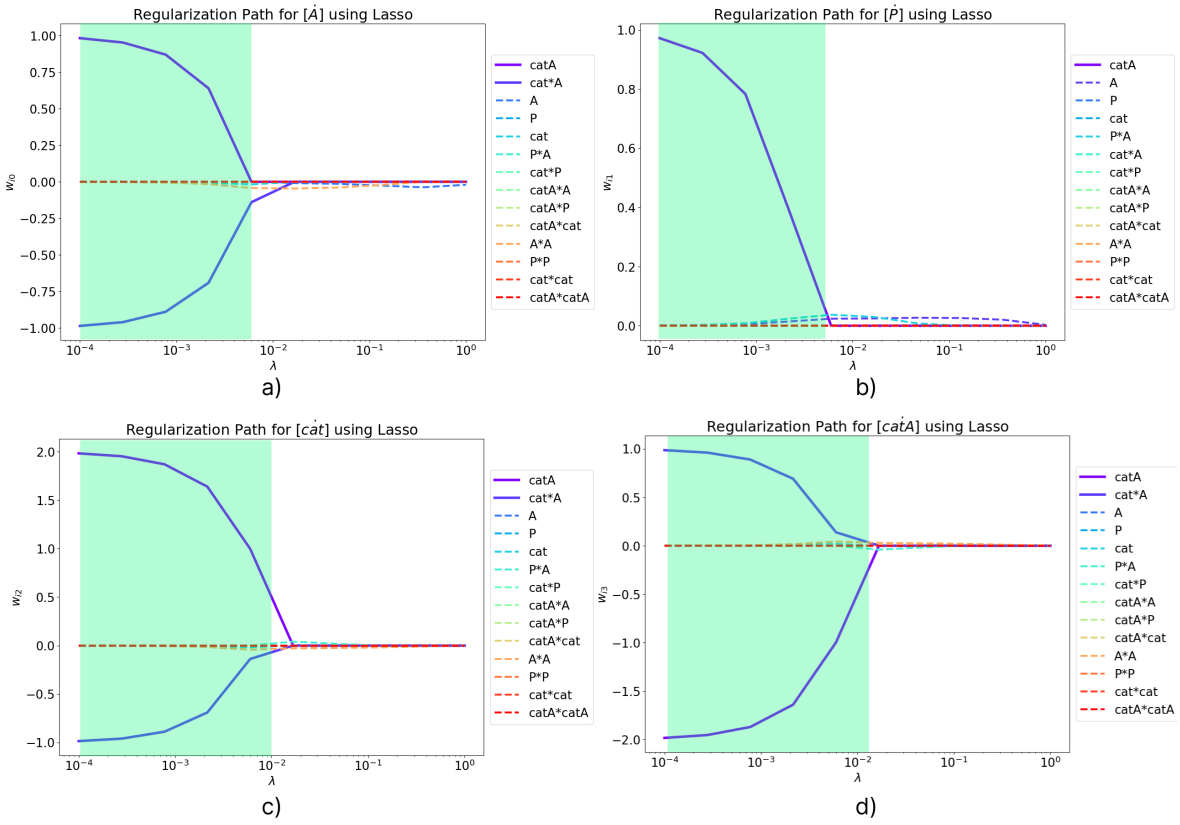

Figure 5.12. Regularization paths for Lasso regression (Chebyshev points, spline approximation, $k_{-2} = 0$): a) [$\dot{A}$]; b) [$\dot{P}$]; c) [$\dot{cat}$]; d) [$\dot{catA}$]

Table 5.3. Lasso regression results for $k_1 = 1$, $k_{-1} = 1$, $k_2 = 1$, $k_{-2} = 0$, $[\text{cat}]_0 = 0.1$

Sampling	Approximation	λ	$\varepsilon'_r(\widehat{Y})$	$\varepsilon_r(\widehat{Y})$	$\varepsilon_r(\widehat{X})$	$\varepsilon_r(W)$	s
Equispaced	finite difference	λ = 0.001	0.4307	0.2757	0.0229	3.064	10
Equispaced	finite difference	λ = 1e-05	0.9997	0.9997	1.2282	1.1272	20
Equispaced	spline	λ = 0.001	0.9105	0.8904	0.0508	7.5710	11
Equispaced	spline	λ = 1e-05	0.3219	0.0896	0.0025	1.9061	14
RKF45	finite difference	λ = 0.001	1.0732	0.8596	0.2589	0.7275	14
RKF45	finite difference	λ = 1e-05	1.0275	0.8181	0.2618	0.9990	16
RKF45	spline	λ = 0.001	0.8014	0.8011	0.2502	0.1486	7
RKF45	spline	λ = 1e-05	0.7800	0.7798	**0.00005**	**0.0016**	**7**
Chebyshev	finite difference	λ = 0.001	0.2469	0.1211	0.0104	1.2316	13
Chebyshev	finite difference	λ = 1e-5	0.2469	0.1211	0.0104	1.2316	13
Chebyshev	finite difference	λ = 0.01	0.0366	0.0367	0.0076	0.1489	8
Chebyshev	spline	λ = 1e-5	0.0365	0.0367	**0.00006**	**0.0013**	**7**

Concluding the results obtained for Lasso regularization, it can be said that Lasso provides a very accurate results for non-equispaced data distributions, which include Chebyshev points and the points adaptively selected by RKF45. The equispaced data distribution provides very low accuracy, selecting the wrong active components. However, it is seen from the first four rows of Table 5.3 that even though the equispaced data usage provides a very erroneous solution, Lasso still proposes sparsity quite efficiently. So for the equispaced data the obtained system of ODEs is wrong, but interpretable.

The same trend holds when using several initial conditions (see Appendix A) and for another set of kinetic constants (see Appendix B). According to the obtained results, it can be concluded that using Lasso regularization with spline approximation for adaptively se-

lected points or for Chebyshev points provide the best result. RKF45-chosen points provide relatively good results even when using a finite difference approximation, which leads to additional numerical error.

Comparing the results with those obtained by STLSQ, it can be concluded that the usage of Lasso regularization is much more preferable due to its lower sensitivity to noise. Furthermore, Lasso consistently provides interpretable and sparse solutions, whereas STLSQ only achieves sparsity when spline approximation is used.

6 Concluding Remarks

The extraction of sparse coefficients for a system of ODEs from chemical time-series data has been implemented using the least squares approach, which is a commonly applied method for such problems [24]. Several experiments have been conducted by using pySINDy library for sparse identification of nonlinear dynamics [31]. However, the application of SINDy with the built-in STLSQ and Ridge regression did not yield relevant results, despite Ridge regression being claimed to be effective for systems with linear relationships [43].

In order to solve the given problem, we employed a simple least squares approach with various regularization techniques to promote sparsity in the solution. Additionally, we examined whether the sampling of the generated data influenced the final outcome. The results obtained for inferring the ODEs of the A1r mechanism, with $k_1 = 1$, $k_{-1} = 1$, $k_2 = 1$ $k_{-2} = 0$, using a single set of initial conditions, are summarized in Table 6.1.

Table 6.1. Simulation results

Solver	Sampling	Approximation	$\varepsilon_r(\widehat{X})$	$\varepsilon_r(W)$	s
Unregularized	Equispaced	finite difference	0.0107	0.9995	56
Unregularized	Equispaced	spline	0.0043	0.9978	56
Unregularized	RKF45	finite difference	0.0128	0.9933	56
Unregularized	RKF45	spline	0.000002	1.0297	56
Unregularized	Chebyshev	finite difference	0.0140	0.9932	56
Unregularized	Chebyshev	spline	0.000003	1.0241	56
STLSQ, α = 0.4	Equispaced	finite difference	0.0149	0.9995	56
STLSQ, α = 0.4	Equispaced	spline	0.0002	0.9978	56
STLSQ, α = 0.4	RKF45	finite difference	0.0128	0.9932	54
STLSQ, α = 0.4	RKF45	spline	**0.000008**	**0.0003**	**7**
STLSQ, α = 0.4	Chebyshev	finite difference	0.1891	0.9931	54
STLSQ, α = 0.4	Chebyshev	spline	0.1814	**0.00004**	**7**
Lasso, λ = 1e-5	Equispaced	finite difference	1.2282	1.1272	20
Lasso, λ = 1e-5	Equispaced	spline	0.0025	1.9061	14
Lasso, λ = 1e-5	RKF45	finite difference	0.2618	0.9990	16
Lasso, λ = 1e-5	RKF45	spline	**0.00005**	**0.0016**	**7**
Lasso, λ = 1e-5	Chebyshev	finite difference	0.0076	0.1489	8
Lasso, λ = 1e-5	Chebyshev	spline	**0.00006**	**0.0013**	**7**

Research has discovered that employing Lasso regularization or STLSQ thresholding for least squares solvers can enhance the sparsity of the solution obtained. Compared to the unregularized least squares solver, which yields a dense coefficient matrix W with $s = 56$, both Lasso and STLSQ contribute to formulating more interpretable governing equations. Nevertheless, STLSQ is highly sensitive to numerical errors, and fails to extract an interpretable system when employing finite difference approximation, leading to a non-sparse W matrix with $s = 54$. Lasso is more stable and proposes sparsity even when using a more erroneous approximation method.

It should be emphasized that achieving sparsity is not the ultimate objective; we must also identify the correct components of the features matrix $\widetilde{X}$ with non-zero coefficients. This fact is corroborated by the $\varepsilon_r(W)$ error, which measures the discrepancy between the acquired matrix W and the correct W_{ex}. The $\varepsilon_r(\widehat{X})$ error provides an indication of how closely the derived solution aligns with the concentrations X.

The sampling of the generated X concentrations significantly affects the accuracy and sparsity of the final result. All solvers yield inaccurate results characterized by high $\varepsilon_r(W)$ errors when employing equispaced points. However, adaptively chosen by RKF45 points and Chebyshev points yield considerably superior outcomes, as evidenced by Table 6.1.

It has been discovered that using several sets of initial conditions can be a bit more efficient (see Appendix A). The results derived from three sets of initial conditions with varying $[\text{cat}]_0$ are marginally more accurate than those obtained from a single set of initial conditions.

When applying more intricate systems of ODEs, such as mechanism A1r with kinetic constant k_{-2} set to 1 rather than 0, the proposed methods frequently fail to extract the accurate ODEs. The findings outlined in Appendix B prove that the complexity of the mechanism directly increases the difficulty in extracting the correct governing equations.

In summary, it can be concluded that it is crucial to minimize the error in the numerical approximation step by employing more accurate techniques (such as using cubic spline instead of finite difference approximation) and by augmenting the density of points at the onset of concentration trajectories. The impact of numerical error can be lessened through the use of a Chebyshev distribution or adaptive RKF45. However, while these options are viable when simulated data is employed, actual data may present more difficulties due to noise and differing sampling techniques.

In general, numerical differentiation is not a favorable option when dealing with real, noisy data. In future works, noise reduction may need to be performed prior to finding the approximation of derivatives, or the integral form of the system could be used to mitigate errors.

An alternative solution could involve focusing on the system itself and eliminating linear relations within it. The Variance Inflation Factor could be measured to detect highly correlated components. The removal of certain highly correlated independent variables could help avoid multicollinearity. Principal Component Analysis (PCA) may yield beneficial results in addressing multicollinearity [44].

The multicollinearity problem discussed in Section 4.1 may also be solved in the future by improving the thresholding and regularization method. For instance elastic net [45] might provide better results than Lasso, combining both Lasso and Ridge regularization. Though elastic net is more efficient for usage when the number of samples is smaller than the number of unknowns, which is not our case, it still might be worth trying, since elastic net is claimed to be efficient for highly correlated independent variables.

Yet another possible approach to solving the least squares problem involves the use of Basis Pursuit Denoising (BDPN) [46], which bears resemblance to Lasso regression, as it

also minimizes the ℓ_1-norm such that $|\widetilde{X}W|_2$ is less than a predefined error value. BDPN proves to be more efficient than Lasso when applied to real noisy data.

In conclusion, the extraction of kinetic constants from time-series data of kinetic mechanisms can be efficiently accomplished using Lasso regularization and adaptive RKF45 sampling. Nevertheless, it cannot be confidently stated that the proposed method will provide satisfactory results for more complex kinetic mechanisms.

The implementation of the described methods can be found at `https://github.com/AyanaMussabayeva/Kinetic-Constants-Extraction`.

References

[1] A. P. Arkin and J. Ross, "Statistical construction of chemical reaction mechanisms from measured time-series," *The Journal of Physical Chemistry*, vol. 99, pp. 970–979, 1995. DOI: 10.1021/J100003A020.

[2] D. G. Blackmond, "Reaction progress kinetic analysis: A powerful methodology for mechanistic studies of complex catalytic reactions," *Angewandte Chemie International Edition*, vol. 44, no. 28, pp. 4302–4320, DOI: 10.1002/anie.200462544.

[3] J. Burés, "Variable time normalization analysis: General graphical elucidation of reaction orders from concentration profiles," *Angewandte Chemie International Edition*, vol. 55, no. 52, pp. 16 084–16 087, 2016. DOI: 10.1002/anie.201609757.

[4] C. Nielsen and J. Burés, "Visual kinetic analysis," *Chemical Science*, vol. 10, pp. 348–353, Jan. 2019. DOI: 10.1039/C8SC04698K.

[5] E. Crampin, P. Mcsharry, and S. Schnell, "Extracting biochemical reaction kinetics from time series data," Sep. 2004, pp. 329–336, ISBN: 978-3-540-23206-3. DOI: 10.1007/978-3-540-30133-2_42.

[6] J. Srividhya, E. J. Crampin, P. E. McSharry, and S. Schnell, "Reconstructing biochemical pathways from time course data," *Proteomics*, vol. 7, pp. 828–838, 2007. DOI: 10.1002/pmic.200600428.

[7] K.-H. Cho, S.-Y. Shin, K. Hyun Woo, O. Wolkenhauer, B. McFerran, and W. Kolch, "Mathematical modeling of the influence of RKIP on the ERK signaling pathway," *Lecture Notes in Computer Science*, vol. 2602, pp. 127–141, Jan. 2003. DOI: 10.1007/3-540-36481-1_11.

[8] C. Moles, P. Mendes, and J. Banga, "Parameter estimation in biochemical pathways: A comparison of global optimization methods," *Genome Research*, vol. 13, pp. 2467–2474, Dec. 2003. DOI: 10.1101/gr.1262503.

[9] N. Galagali and Y. M. Marzouk, "Bayesian inference of chemical kinetic models from proposed reactions," *Chemical Engineering Science*, vol. 123, pp. 170–190, 2015, ISSN: 0009-2509. DOI: 10.1016/j.ces.2014.10.030.

[10] I.-C. Chou and E. O. Voit, "Recent developments in parameter estimation and structure identification of biochemical and genomic systems," *Mathematical Biosciences*, vol. 219, no. 2, pp. 57–83, 2009, ISSN: 0025-5564. DOI: 10.1016/j.mbs.2009.03.002.

[11] M. A. Mourão, J. Srividhya, P. E. McSharry, E. J. Crampin, and S. Schnell, "A graphical user interface for a method to infer kinetics and network architecture (MIKANA)," *PLoS ONE*, vol. 6, 2011. DOI: 10.1371/journal.pone.0027534.

[12] W. Ji and S. Deng, "Autonomous discovery of unknown reaction pathways from data by chemical reaction neural network," *The Journal of Physical Chemistry*, pp. 1082–1092, Jan. 2021. DOI: 10.1021/acs.jpca.0c09316.

[13] A. C. Hindmarsh and L. R. Petzold, "LSODA, Ordinary Differential Equation Solver for Stiff or Non-Stiff System," 2005. [Online]. Available: http://inis.iaea.org/search/search.aspx?orig_q=RN:41086668.

[14] A. Curtis, "The FACSIMILE numerical integrator for stiff initial value problems," in *Proceedings of the Conference on Computational Techniques for Ordinary Differential Equations*, 1980, pp. 47–82.

[15] J. Dormand and P. Prince, "A family of embedded Runge-Kutta formulae," *Journal of Computational and Applied Mathematics*, vol. 6, no. 1, pp. 19–26, 1980, ISSN: 0377-0427. DOI: 10.1016/0771-050X(80)90013-3.

[16] E. Fehlberg, "Low-order classical runge-kutta formulas with stepsize control and their application to some heat transfer problems," NASA Technical Report R-315, 1969.

[17] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020. DOI: 10.1038/s41592-019-0686-2.

[18] J. Srividhya, M. A. Mourão, E. J. Crampin, and S. Schnell, "Enzyme catalyzed reactions: From experiment to computational mechanism reconstruction," *Computational Biology and Chemistry*, vol. 34, no. 1, pp. 11–18, 2010, ISSN: 1476-9271. DOI: 10.1016/j.compbiolchem.2009.10.007.

[19] R. J. LeVeque, *Finite Difference Methods for Ordinary and Partial Differential Equations*. Society for Industrial and Applied Mathematics, 2007. DOI: 10.1137/1.9780898717839.

[20] H. W. Press, P. Flannery, A. S. Teukolsky, and W. T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, Second. Cambridge, USA: Cambridge University Press, 1992.

[21] G. Birkhoff and C. de Boor, "Piecewise polynomial interpolation and approximation," *Approximation of Functions*, pp. 164–190, 1964.

[22] C. de Boor, *A Practical Guide to Spline*. Jan. 1978, vol. Volume 27. DOI: 10.2307/2006241.

[23] P. Goyal and P. Benner, "Discovery of nonlinear dynamical systems using a Runge-Kutta inspired dictionary-based sparse regression approach," vol. 478, no. 2262, p. 20 210 883, 2022. DOI: 10.1098/rspa.2021.0883.

[24] 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, ISSN: 0027-8424. DOI: 10.1073/pnas.1517384113.

[25] L. N. Trefethen, *Approximation Theory and Approximation Practice, Extended Edition*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2019. DOI: 10.1137/1.9781611975949.

[26] J. P. Boyd, "The Rate of Convergence of Chebyshev Polynomials for Functions Which Have Asymptotic Power Series About One Endpoint," *Mathematics of Computation*, vol. 37, no. 155, pp. 189–195, Mar. 1981, ISSN: 00255718, 10886842. DOI: 10.2307/2007511.

[27] 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. DOI: 10.1126/sciadv.1602614.

[28] N. M. Mangan, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Inferring biological networks by sparse identification of nonlinear dynamics," *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, vol. 2, no. 1, pp. 52–63, 2016. DOI: 10.1109/TMBMC.2016.2633265.

[29] M. Sorokina, S. Sygletos, and S. Turitsyn, "Sparse identification for nonlinear optical communication systems: Sino method," *Opt. Express*, vol. 24, no. 26, pp. 30 433–30 443, 2016. DOI: 10.1364/OE.24.030433.

[30] M. Hoffmann, C. Fröhner, and F. Noé, "Reactive SINDy: Discovering governing reactions from concentration data," *The Journal of Chemical Physics*, vol. 150, Jan. 2019. DOI: 10.1063/1.5066099.

[31] B. de Silva, K. Champion, M. Quade, J.-C. Loiseau, J. Kutz, and S. Brunton, "Pysindy: A python package for the sparse identification of nonlinear dynamical systems from data," *Journal of Open Source Software*, vol. 5, no. 49, p. 2104, 2020. DOI: 10.21105/joss.02104.

[32] A. A. Kaptanoglu, B. M. de Silva, U. Fasel, K. Kaheman, A. J. Goldschmidt, J. Callaham, C. B. Delahunt, Z. G. Nicolaou, K. Champion, J.-C. Loiseau, J. N. Kutz, and S. L. Brunton, "Pysindy: A comprehensive python package for robust sparse system identification," *Journal of Open Source Software*, vol. 7, no. 69, p. 3994, 2022. DOI: 10.21105/joss.03994.

[33] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 42, no. 1, pp. 80–86, 2000, ISSN: 00401706. [Online]. Available: http://www.jstor.org/stable/1271436 (visited on 06/13/2022).

[34] D. Montgomery, E. Peck, and G. Vining, *Introduction to linear regression analysis*, 3. ed, ser. Wiley series in probability and statistics. New York, NY [u.a.]: Wiley, 2001, XVI, 641, ISBN: 0471315656. [Online]. Available: http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+321916239&sourceid=fbw_bibsonomy.

[35] J. H. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *Journal of Statistical Software*, vol. 33, no. 1, pp. 1–22, 2010. DOI: 10.18637/jss.v033.i01.

[36] C. L. Perrin, "Linear or nonlinear least-squares analysis of kinetic data?" *Journal of Chemical Education*, vol. 94, no. 6, pp. 669–672, 2017. DOI: 10.1021/acs.jchemed.6b00629.

[37] T. Blumensath and M. E. Davies, "Iterative Thresholding for Sparse Approximations," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5, pp. 629–654, 2008, ISSN: 1531-5851. DOI: 10.1007/s00041-008-9035-z.

[38] L. Zhang and H. Schaeffer, "On the convergence of the sindy algorithm," *Multiscale Modeling & Simulation*, vol. 17, no. 3, pp. 948–972, 2019. DOI: 10.1137/18M1189828.

[39] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996. DOI: doi.org/10.1111/j.2517-6161.1996.tb02080.x.

[40] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[41] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd, "A rewriting system for convex optimization problems," *Journal of Control and Decision*, vol. 5, no. 1, pp. 42–60, 2018.

[42] M. Y. Park and T. Hastie, "L1-regularization path algorithm for generalized linear models," *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, vol. 69, no. 4, pp. 659–677, 2007, ISSN: 13697412, 14679868. [Online]. Available: http://www.jstor.org/stable/4623289 (visited on 04/18/2023).

[43] D. Schreiber-Gregory, "Ridge regression and multicollinearity: An in-depth review," *Model Assisted Statistics and Applications*, vol. 13, pp. 359–365, Sep. 2018. DOI: 10.3233/MAS-180446.

[44] A. Gwelo, "Principal components to overcome multicollinearity problem," *Oradea Journal of Business and Economics*, vol. 4, pp. 79–91, Mar. 2019. DOI: 10.47535/1991ojbe062.

[45] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005. DOI: 10.1111/j.1467-9868.2005.00503.x.

[46] S. Chen and D. Donoho, "Basis pursuit," in *Proceedings of 1994 28th Asilomar Conference on Signals, Systems and Computers*, vol. 1, 1994, pp. 41–44. DOI: 10.1109/ACSSC.1994.471413.

Appendices

A Mechanism A1r, $k_{-2} = 0$

Initial conditions: $[\text{cat}]_0 = [0.05; 0.1; 0.15]$

Table A.1. Simulation results for $[\text{cat}]_0 = [0.05; 0.1; 0.15]$, $k_{-2} = 0$

Solver	Sampling	Approximation	$\varepsilon_r(\widehat{X})$	$\varepsilon_r(W)$	s
Unregularized	Equispaced	finite difference	0.0109	0.9927	56
Unregularized	Equispaced	spline	0.0263	0.9598	56
Unregularized	RKF45	finite difference	0.0128	0.9933	56
Unregularized	RKF45	spline	0.000002	1.0297	56
Unregularized	Chebyshev	finite difference	0.0140	0.9932	56
Unregularized	Chebyshev	spline	0.000003	1.0241	56
STLSQ, α = 0.4	Equispaced	finite difference	0.0231	0.9952	56
STLSQ, α = 0.4	Equispaced	spline	0.0136	0.9423	23
STLSQ, α = 0.4	RKF45	finite difference	0.0129	0.9921	54
STLSQ, α = 0.4	RKF45	spline	**0.00006**	**0.0011**	**7**
STLSQ, α = 0.4	Chebyshev	finite difference	0.1887	0.9931	52
STLSQ, α = 0.4	Chebyshev	spline	0.1812	**0.00003**	**7**
Lasso, λ = 1e-5	Equispaced	finite difference	0.0127	1.0545	30
Lasso, λ = 1e-5	Equispaced	spline	0.0047	0.9693	12
Lasso, λ = 1e-5	RKF45	finite difference	0.2618	0.9990	16
Lasso, λ = 1e-5	RKF45	spline	**0.00004**	**0.0013**	**7**
Lasso, λ = 1e-5	Chebyshev	finite difference	0.0079	0.1395	8
Lasso, λ = 1e-5	Chebyshev	spline	**0.00005**	**0.0012**	**7**

A.1 Regularization Paths for STLSQ

Equispaced points:

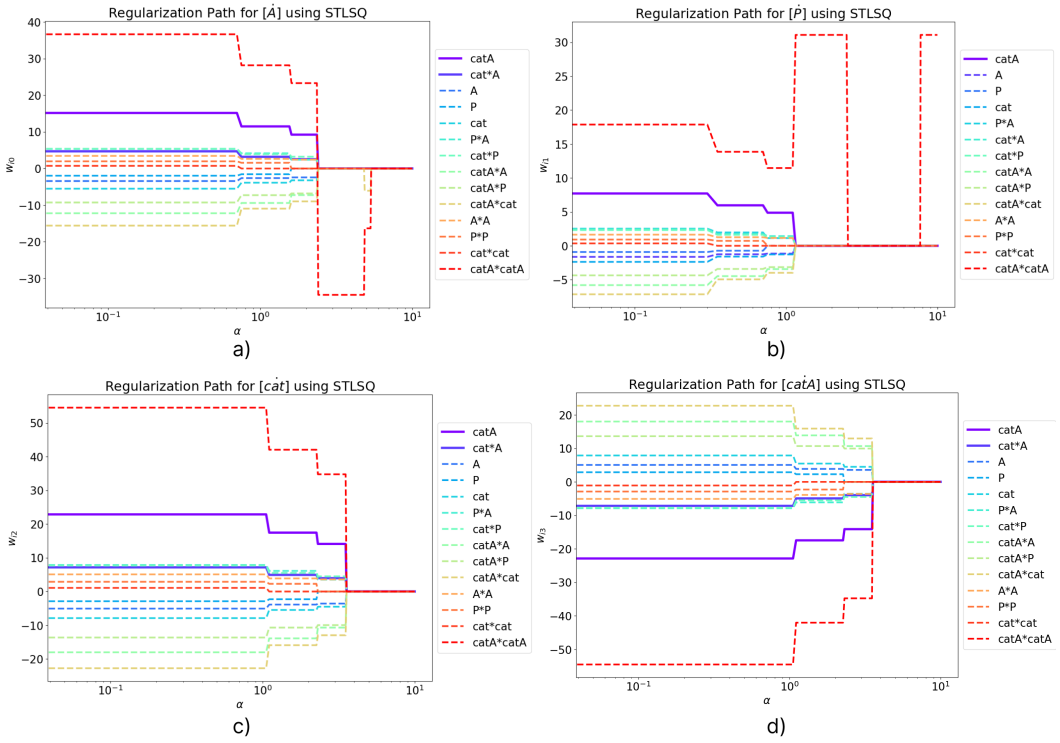

Figure A.1. Regularization paths for STLSQ (equispaced points, finite difference approximation, $k_{-2} = 0$, $[\text{cat}]_0 = [0.05; 0.1; 0.15]$): a) [$\dot{\text{A}}$] equation; b) [$\dot{\text{P}}$] equation; c) [$\dot{\text{cat}}$] equation; d) [$\dot{\text{catA}}$] equation

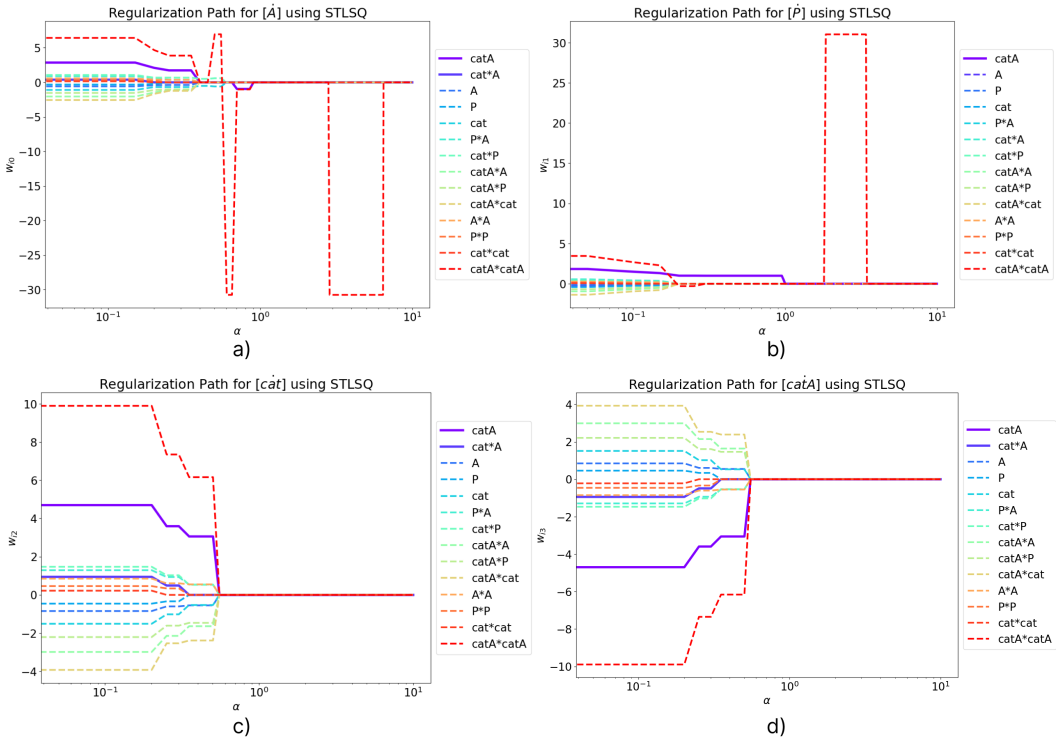

Figure A.2. Regularization paths for STLSQ (equispaced points, spline approximation, $k_{-2} = 0$, $[\text{cat}]_0 = [0.05; 0.1; 0.15]$): a) [$\dot{\text{A}}$] equation; b) [$\dot{\text{P}}$] equation; c) [$\dot{\text{cat}}$] equation; d) [$\dot{\text{catA}}$] equation

RKF45-chosen points:

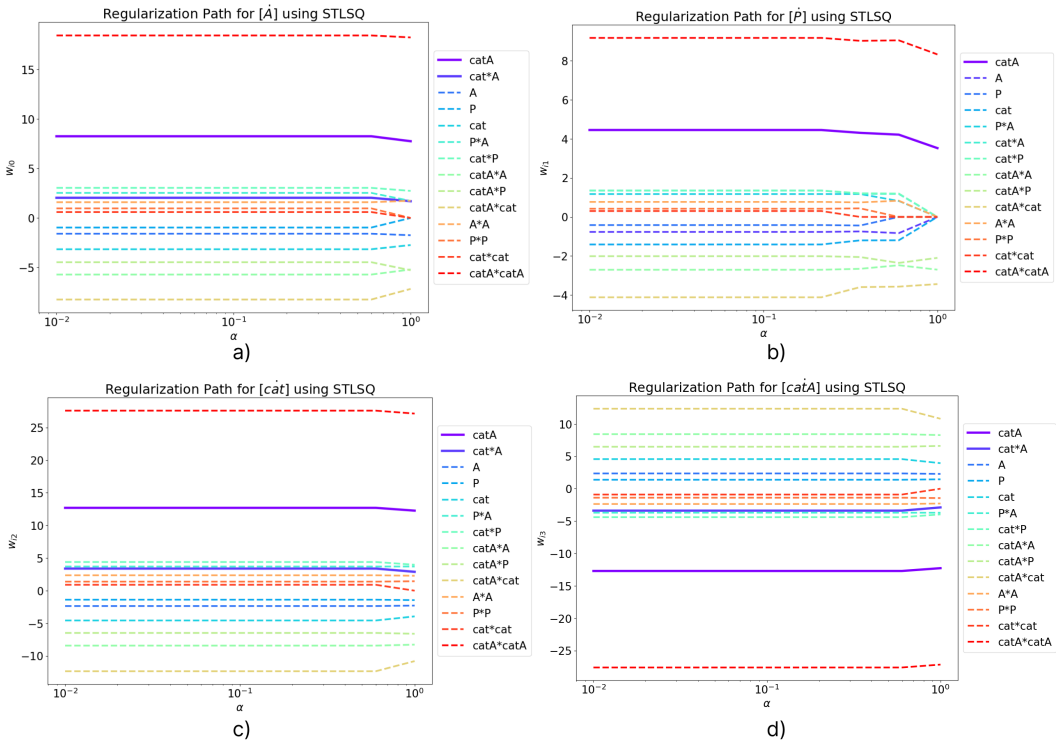

Figure A.3. Regularization paths for STLSQ (RKF45 points, finite difference approximation, $k_{-2} = 0$, $[\text{cat}]_0 = [0.05; 0.1; 0.15]$): a) $[\dot{\text{A}}]$ equation; b) $[\dot{\text{P}}]$ equation; c) $[\dot{\text{cat}}]$ equation; d) $[\dot{\text{catA}}]$ equation

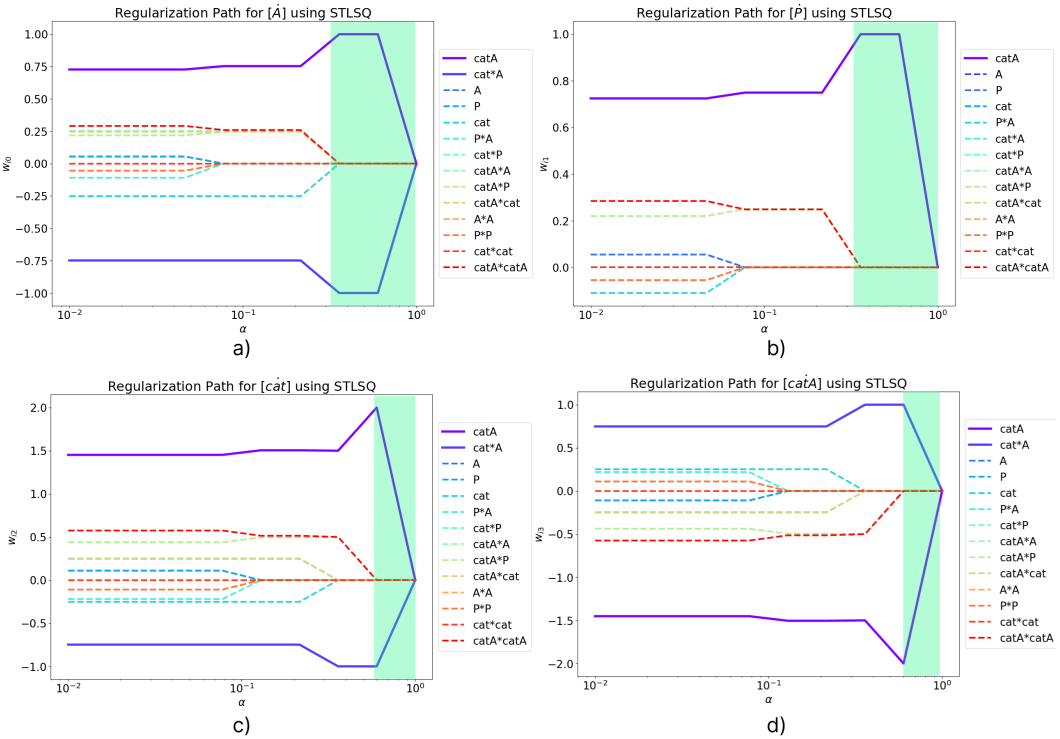

Figure A.4. Regularization paths for STLSQ (RKF45 points, spline approximation, $k_{-2} = 0$, $[\text{cat}]_0 = [0.05; 0.1; 0.15]$): a) $[\dot{\text{A}}]$ equation; b) $[\dot{\text{P}}]$ equation; c) $[\dot{\text{cat}}]$ equation; d) $[\dot{\text{catA}}]$ equation

Chebyshev points:

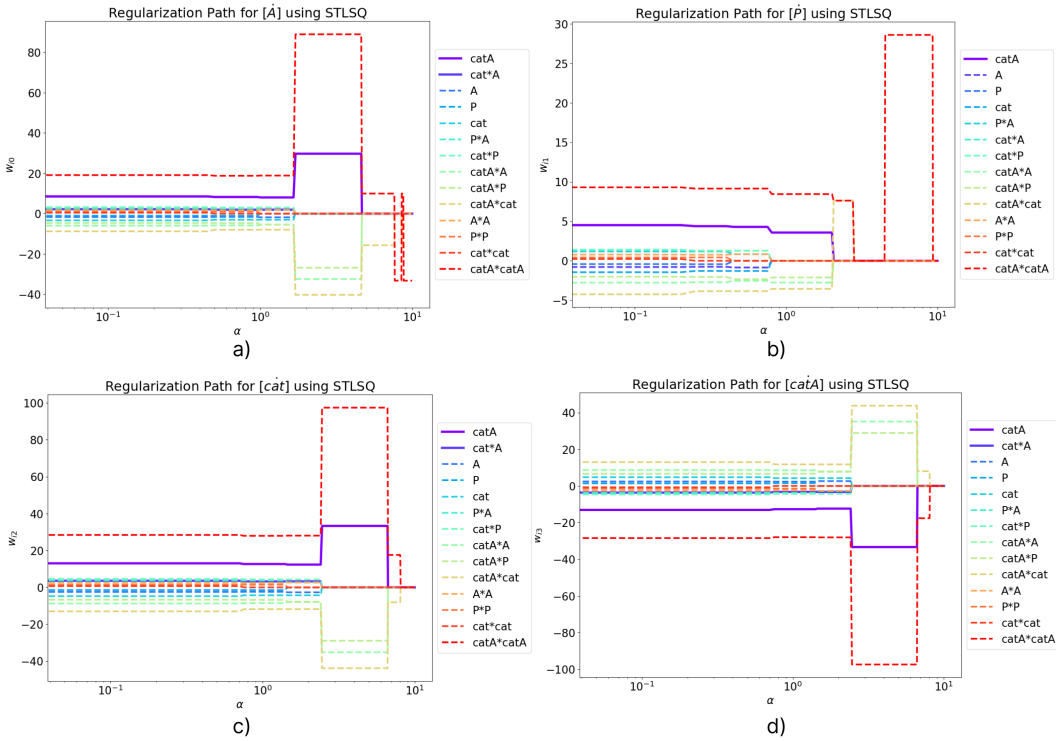

Figure A.5. Regularization paths for STLSQ (Chebyshev points, finite difference approximation, $k_{-2} = 0$, $[\text{cat}]_0 = [0.05; 0.1; 0.15]$): a) $[\dot{\text{A}}]$ equation; b) $[\dot{\text{P}}]$ equation; c) $[\dot{\text{cat}}]$ equation; d) $[\dot{\text{catA}}]$ equation

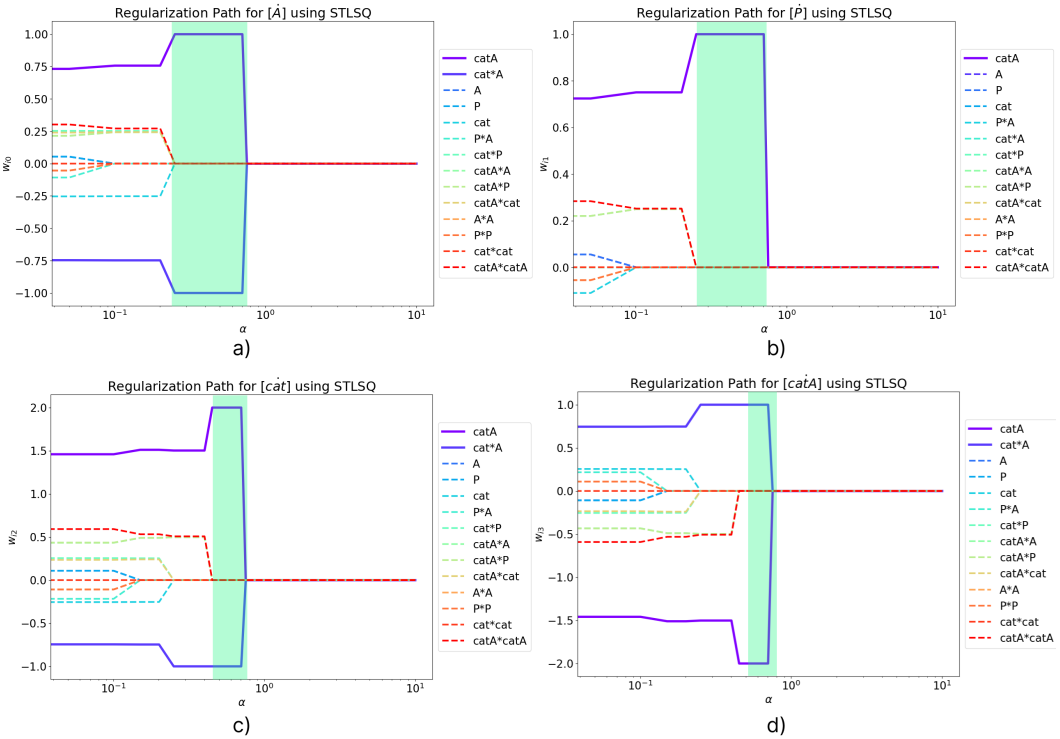

Figure A.6. Regularization paths for STLSQ (Chebyshev points, spline approximation, $k_{-2} = 0$, $[\text{cat}]_0 = [0.05; 0.1; 0.15]$): a) $[\dot{\text{A}}]$ equation; b) $[\dot{\text{P}}]$ equation; c) $[\dot{\text{cat}}]$ equation; d) $[\dot{\text{catA}}]$ equation

A.2 Regularization Paths for Lasso

Equispaced points:

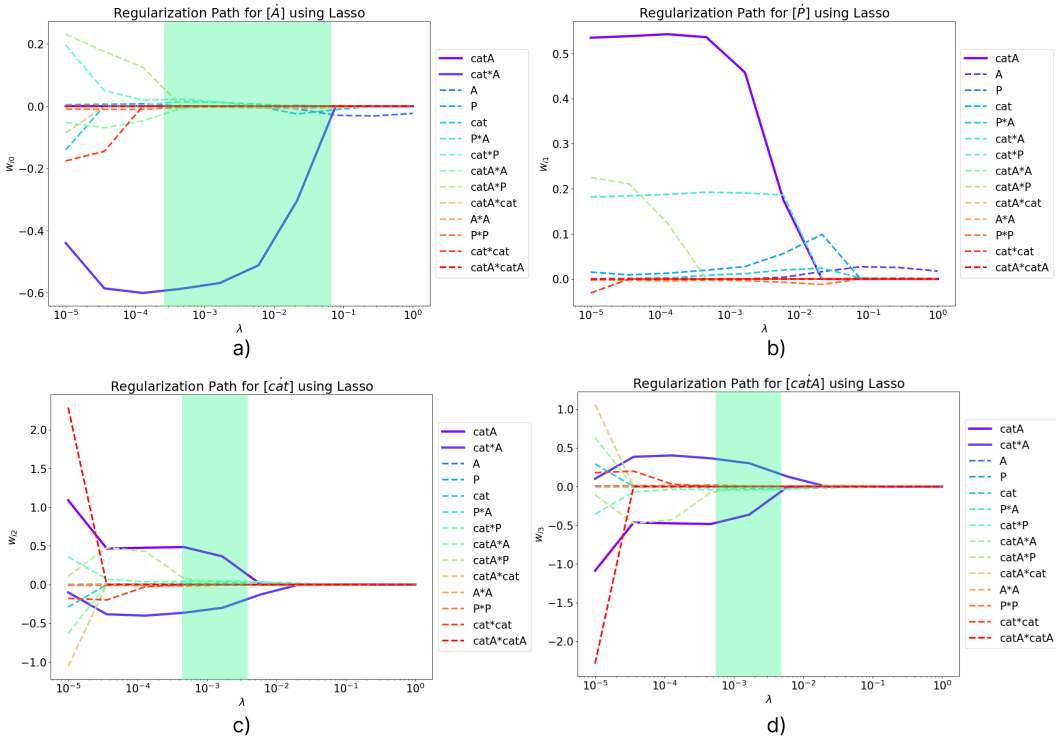

Figure A.7. Regularization paths for Lasso (equispaced points, finite difference approximation, $k_{-2} = 0$, $[\text{cat}]_0 = [0.05; 0.1; 0.15]$): a) [$\dot{\text{A}}$] equation; b) [$\dot{\text{P}}$] equation; c) [$\dot{\text{cat}}$] equation; d) [$\dot{\text{catA}}$] equation

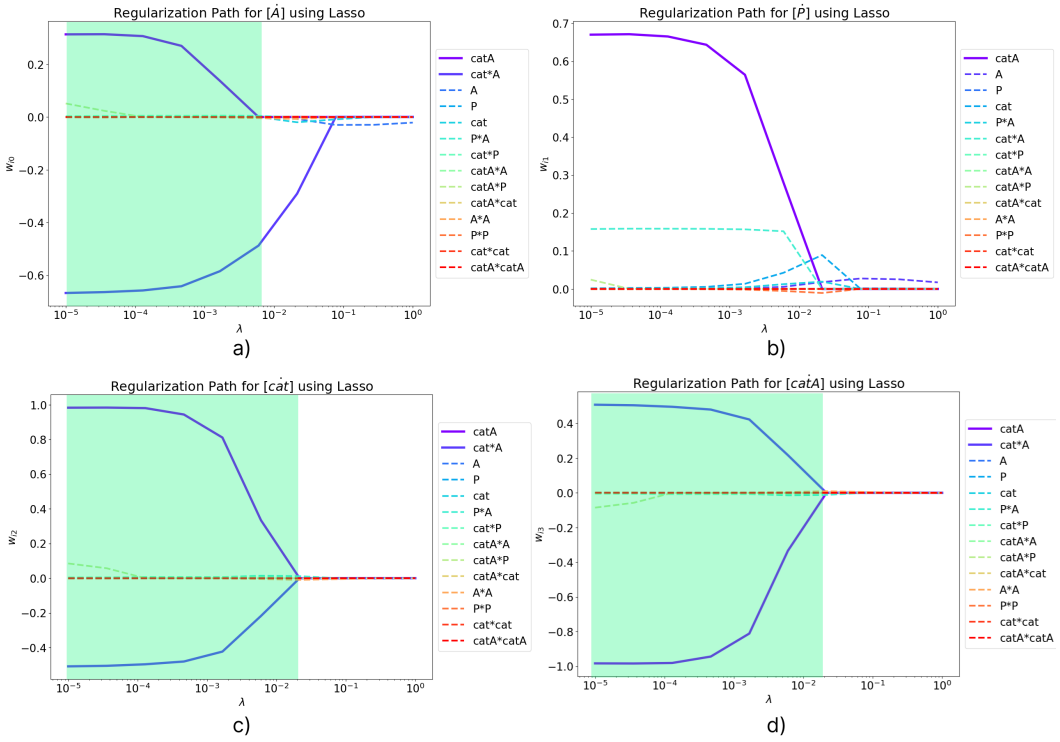

Figure A.8. Regularization paths for STLSQ (equispaced points, spline approximation, $k_{-2} = 0$, $[\text{cat}]_0 = [0.05; 0.1; 0.15]$): a) [$\dot{\text{A}}$] equation; b) [$\dot{\text{P}}$] equation; c) [$\dot{\text{cat}}$] equation; d) [$\dot{\text{catA}}$] equation

RKF45-chosen points:

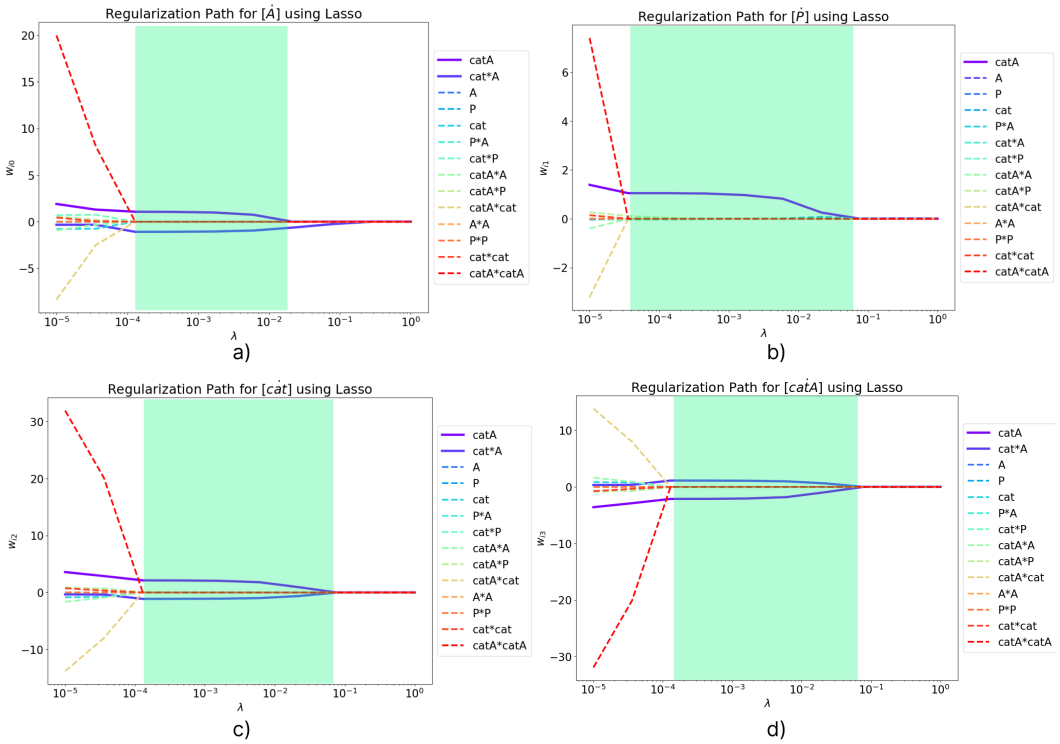

Figure A.9. Regularization paths for Lasso (RKF45 points, finite difference approximation, $k_{-2} = 0$, $[\text{cat}]_0 = [0.05; 0.1; 0.15]$): a) $[\dot{\text{A}}]$ equation; b) $[\dot{\text{P}}]$ equation; c) $[\dot{\text{cat}}]$ equation; d) $[\dot{\text{catA}}]$ equation

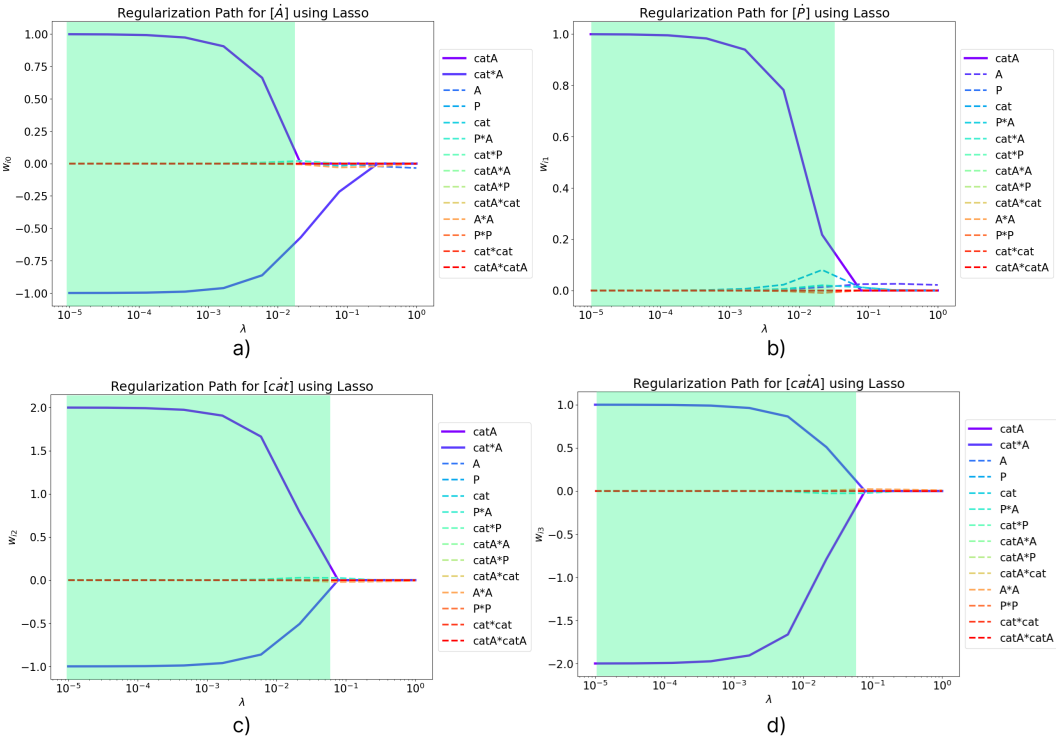

Figure A.10. Regularization paths for Lasso (RKF45 points, spline approximation, $k_{-2} = 0$, $[\text{cat}]_0 = [0.05; 0.1; 0.15]$): a) $[\dot{\text{A}}]$ equation; b) $[\dot{\text{P}}]$ equation; c) $[\dot{\text{cat}}]$ equation; d) $[\dot{\text{catA}}]$ equation

Chebyshev points:

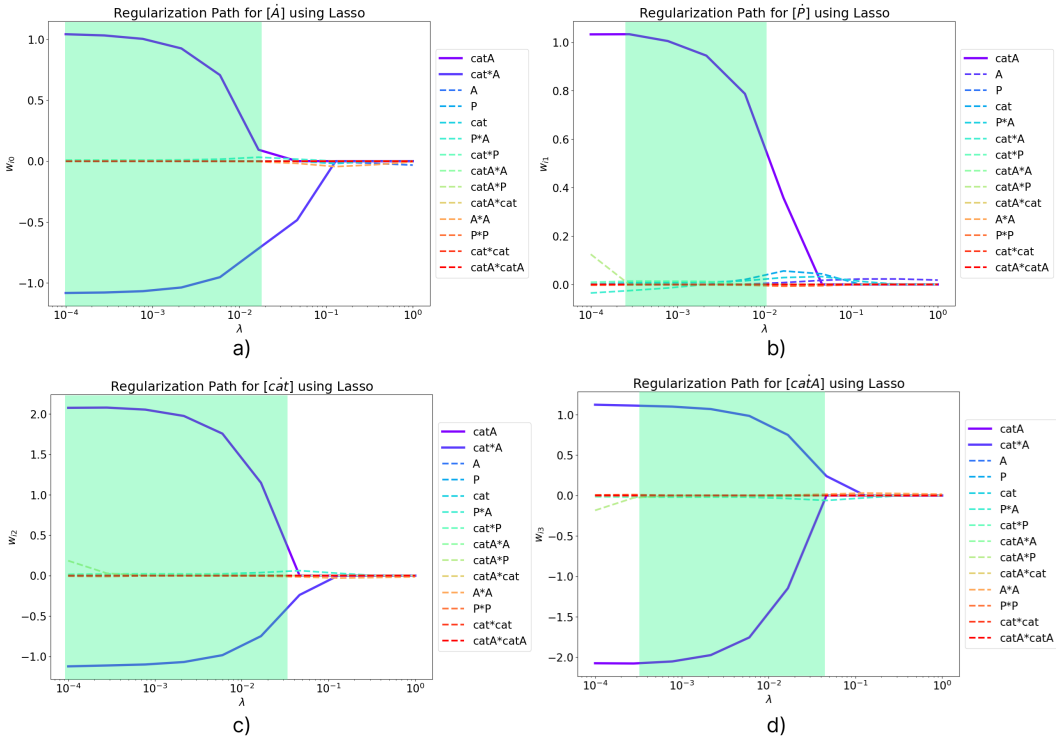

Figure A.11. Regularization paths for Lasso (Chebyshev points, finite difference approximation, $k_{-2} = 0$, $[\text{cat}]_0 = [0.05; 0.1; 0.15]$): a) $[\dot{\text{A}}]$ equation; b) $[\dot{\text{P}}]$ equation; c) $[\dot{\text{cat}}]$ equation; d) $[\dot{\text{catA}}]$ equation

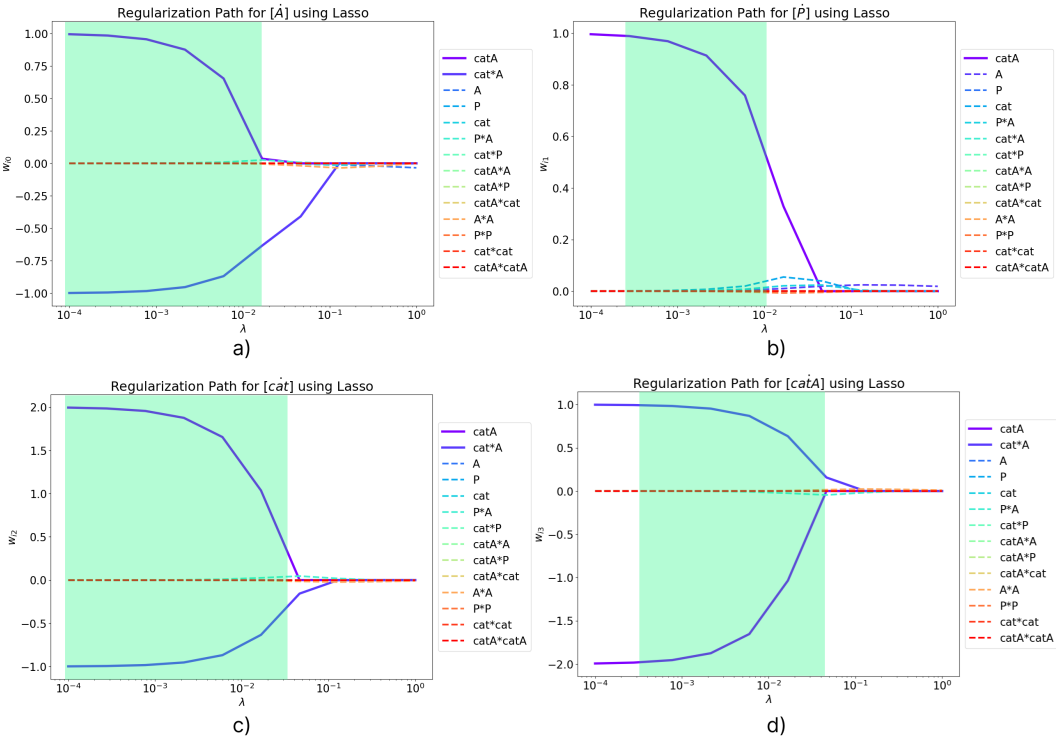

Figure A.12. Regularization paths for Lasso (Chebyshev points, spline approximation, $k_{-2} = 0$, $[\text{cat}]_0 = [0.05; 0.1; 0.15]$): a) $[\dot{\text{A}}]$ equation; b) $[\dot{\text{P}}]$ equation; c) $[\dot{\text{cat}}]$ equation; d) $[\dot{\text{catA}}]$ equation

B Mechanism A1r, $k_{-2} = 1$

Initial conditions: $[\text{cat}]_0 = 0.1$

Table B.1. Simulation results for $[\text{cat}]_0 = 0.1$, $k_{-2} = 1$

Solver	Sampling	Approximation	$\varepsilon_r(\widehat{X})$	$\varepsilon_r(W)$	s
Unregularized	Equispaced	finite difference	0.0243	1.0023	56
Unregularized	Equispaced	spline	0.0141	0.9998	56
Unregularized	RKF45	finite difference	0.0278	1.2235	56
Unregularized	RKF45	spline	0.000002	1.0815	56
Unregularized	Chebyshev	finite difference	0.1823	1.2186	56
Unregularized	Chebyshev	spline	0.1393	1.1571	56
STLSQ, α = 0.4	Equispaced	finite difference	0.0211	1.0293	56
STLSQ, α = 0.4	Equispaced	spline	0.0142	0.9998	56
STLSQ, α = 0.4	RKF45	finite difference	0.5329	0.9821	52
STLSQ, α = 0.4	RKF45	spline	0.6528	0.7997	6
STLSQ, α = 0.4	Chebyshev	finite difference	0.7883	0.9991	48
STLSQ, α = 0.4	Chebyshev	spline	0.7108	0.7996	6
Lasso, λ = 1e-5	Equispaced	finite difference	0.0173	1.3245	26
Lasso, λ = 1e-5	Equispaced	spline	0.0148	3.3832	16
Lasso, λ = 1e-5	RKF45	finite difference	0.2913	0.8371	16
Lasso, λ = 1e-5	RKF45	spline	0.00006	0.6986	14
Lasso, λ = 1e-5	Chebyshev	finite difference	0.0179	0.9395	16
Lasso, λ = 1e-5	Chebyshev	spline	0.00006	0.6627	14

B.0.1 Regularization Paths for STLSQ

Equispaced points:

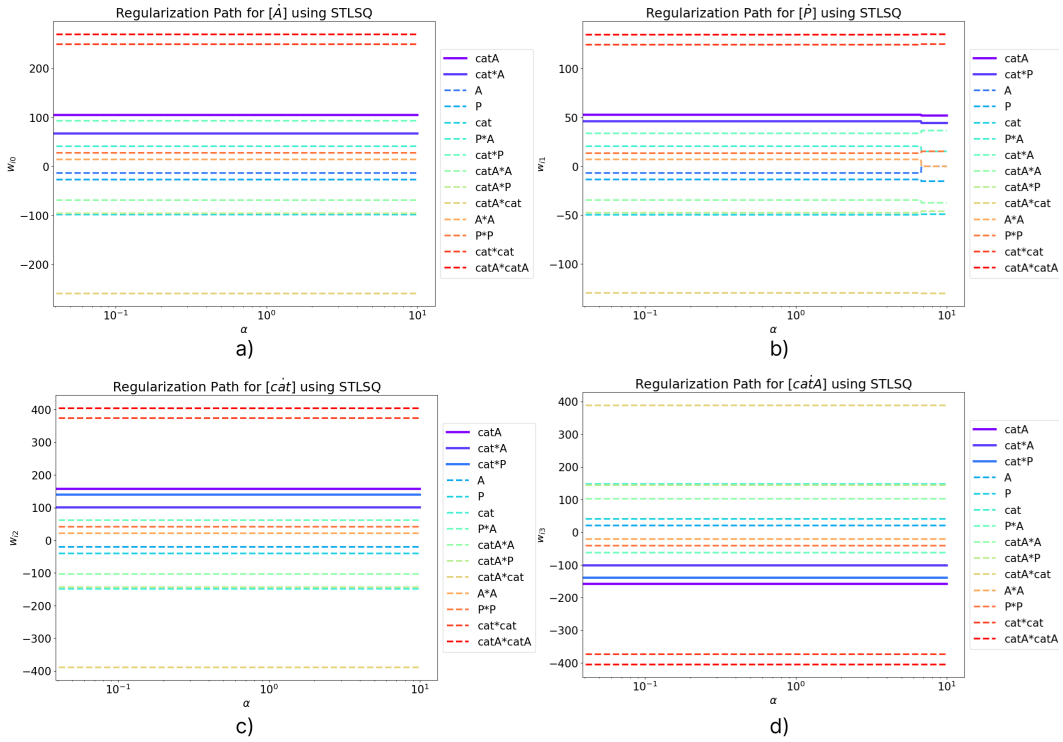

Figure B.1. Regularization paths for STLSQ (equispaced points, finite difference approximation, $k_{-2} = 1$, $[\text{cat}]_0 = [0.05; 0.1; 0.15]$): a) $[\dot{\text{A}}]$ equation; b) $[\dot{\text{P}}]$ equation; c) $[\dot{\text{cat}}]$ equation; d) $[\text{cat}\dot{\text{A}}]$ equation

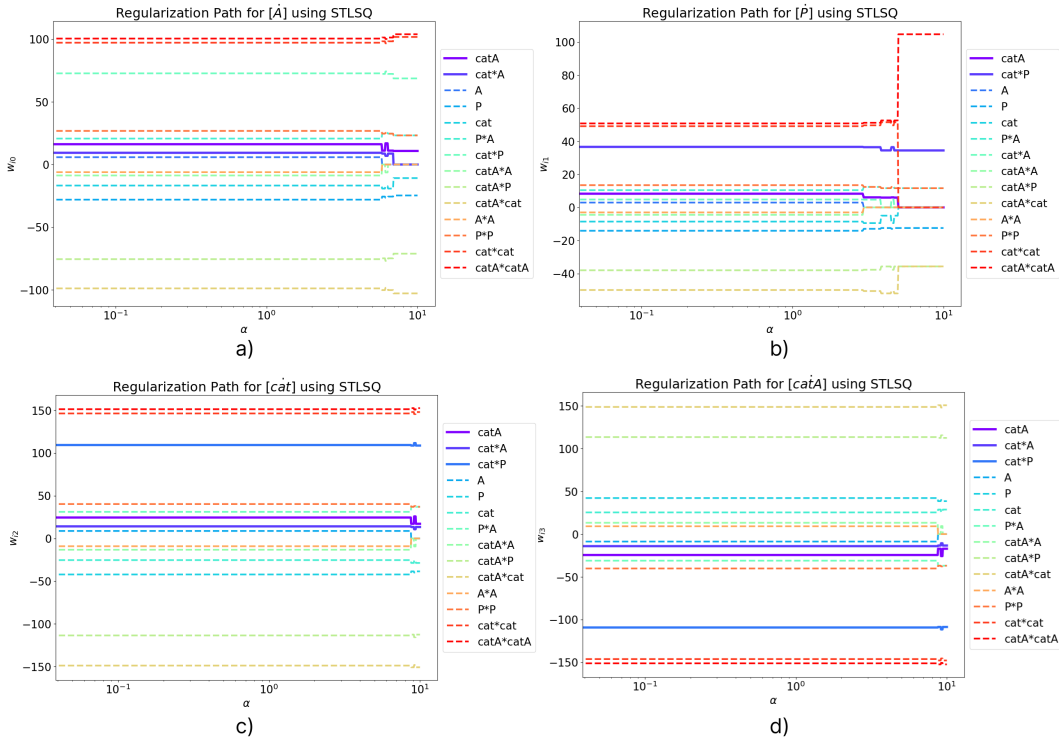

Figure B.2. Regularization paths for STLSQ (equispaced points, spline approximation, $k_{-2} = 1$, $[\text{cat}]_0 = 0.1$): a) $[\dot{\text{A}}]$ equation; b) $[\dot{\text{P}}]$ equation; c) $[\dot{\text{cat}}]$ equation; d) $[\text{cat}\dot{\text{A}}]$ equation

RKF45-chosen points:

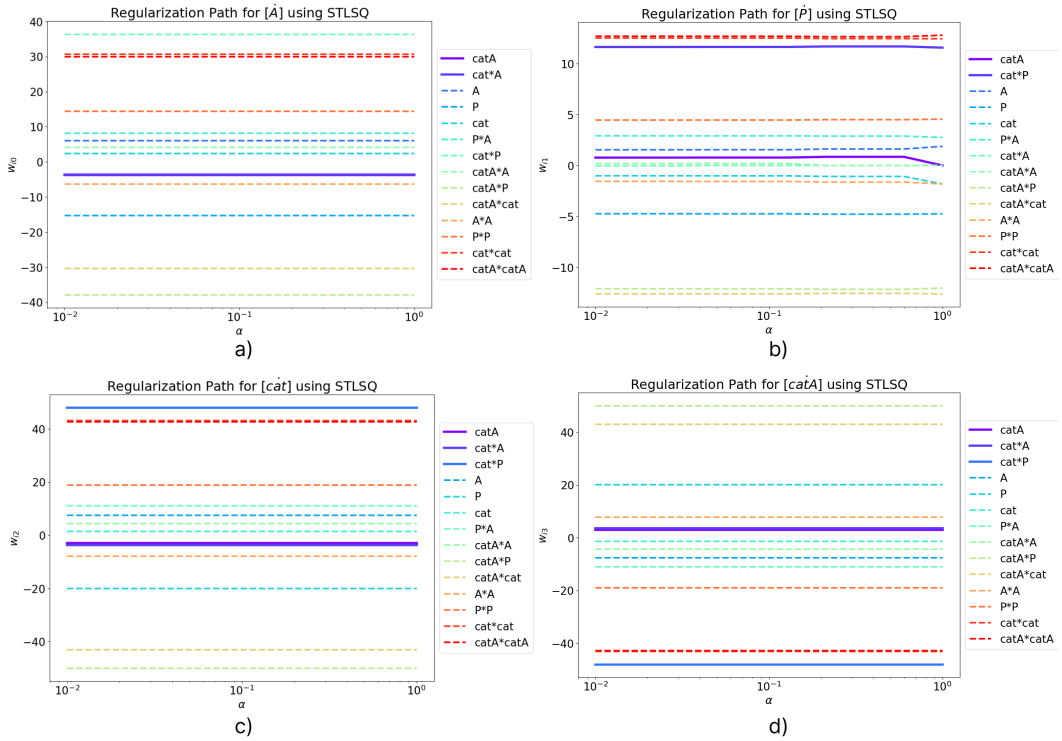

Figure B.3. Regularization paths for STLSQ (RKF45 points, finite difference approximation, $k_{-2} = 1$, $[\text{cat}]_0 = [0.05; 0.1; 0.15]$): a) $[\dot{\text{A}}]$ equation; b) $[\dot{\text{P}}]$ equation; c) $[\dot{\text{cat}}]$ equation; d) $[\text{cat}\dot{\text{A}}]$ equation

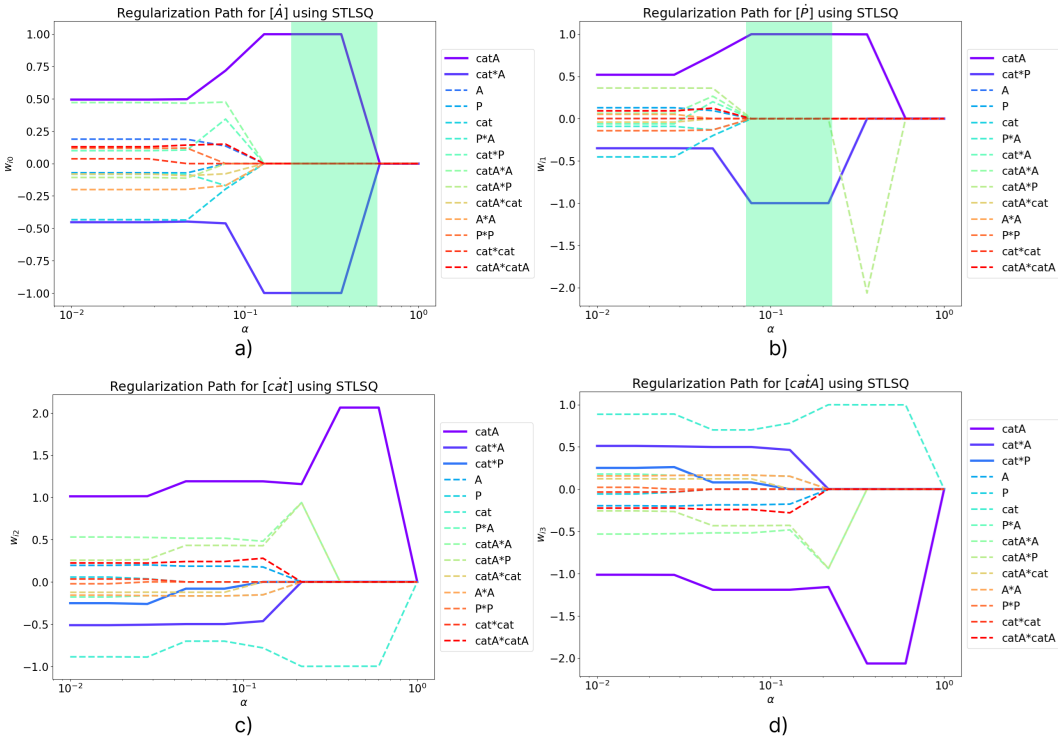

Figure B.4. Regularization paths for STLSQ (RKF45 points, spline approximation, $k_{-2} = 1$, $[\text{cat}]_0 = 0.1$): a) $[\dot{\text{A}}]$ equation; b) $[\dot{\text{P}}]$ equation; c) $[\dot{\text{cat}}]$ equation; d) $[\text{cat}\dot{\text{A}}]$ equation

Chebyshev points:

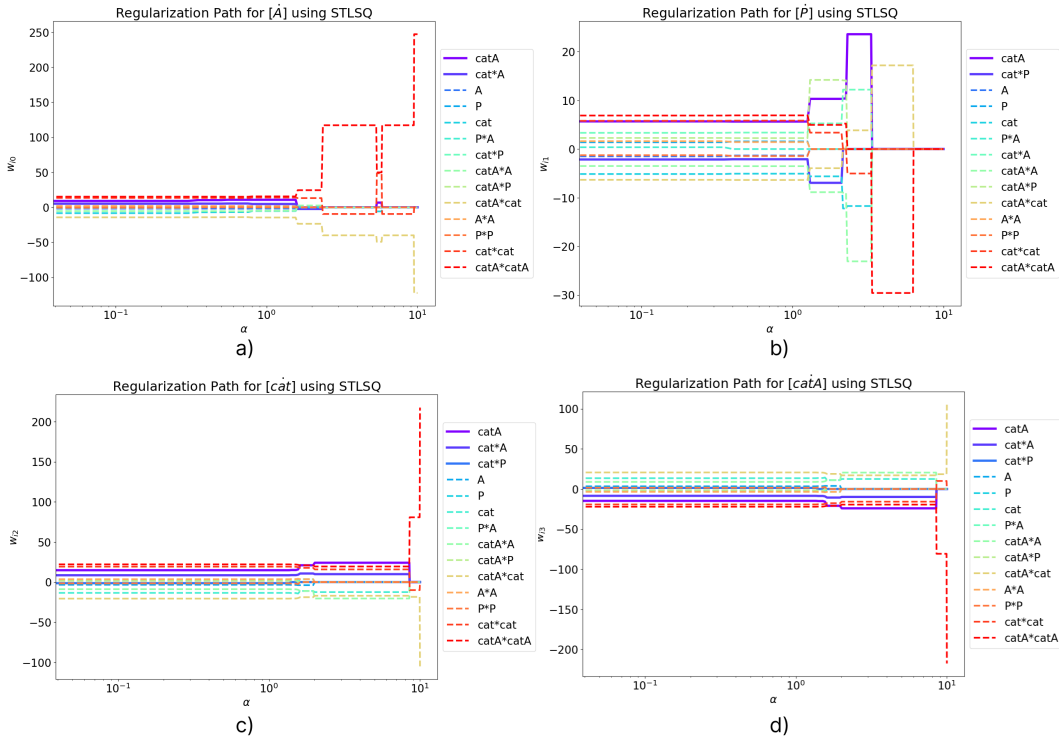

Figure B.5. Regularization paths for STLSQ (Chebyshev points, finite difference approximation, $k_{-2} = 1$, $[\text{cat}]_0 = [0.05; 0.1; 0.15]$): a) $[\dot{\text{A}}]$ equation; b) $[\dot{\text{P}}]$ equation; c) $[\dot{\text{cat}}]$ equation; d) $[\dot{\text{catA}}]$ equation

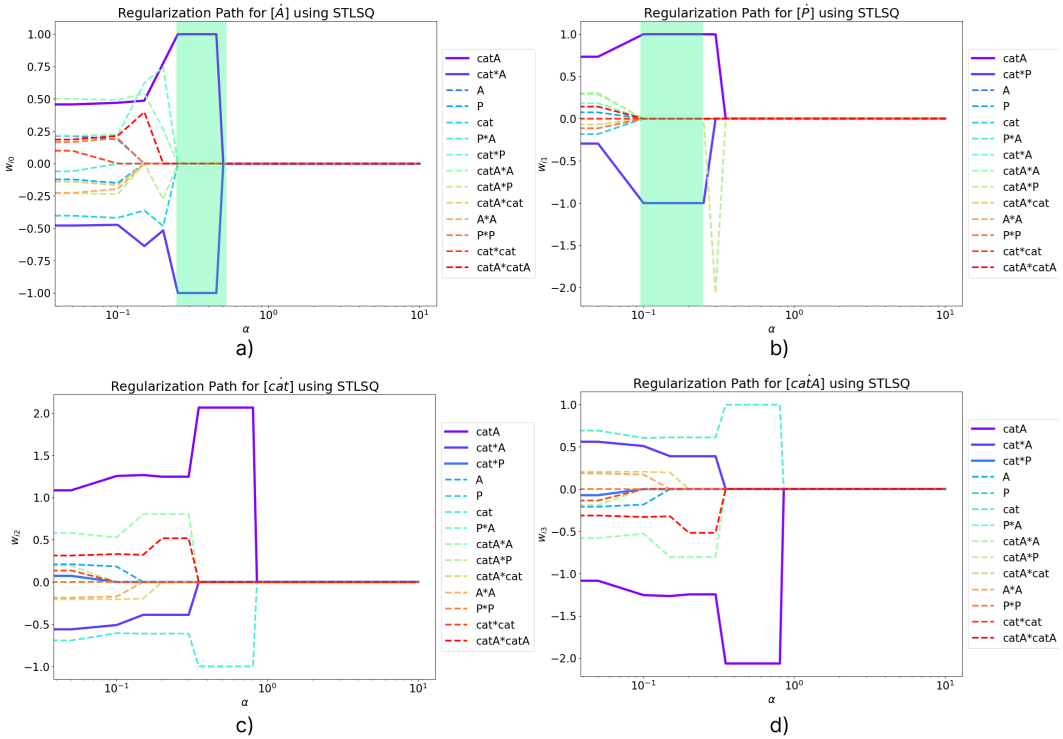

Figure B.6. Regularization paths for STLSQ (Chebyshev points, spline approximation, $k_{-2} = 1$, $[\text{cat}]_0 = 0.1$): a) $[\dot{\text{A}}]$ equation; b) $[\dot{\text{P}}]$ equation; c) $[\dot{\text{cat}}]$ equation; d) $[\dot{\text{catA}}]$ equation

B.0.2 Regularization Paths for Lasso

Equispaced points:

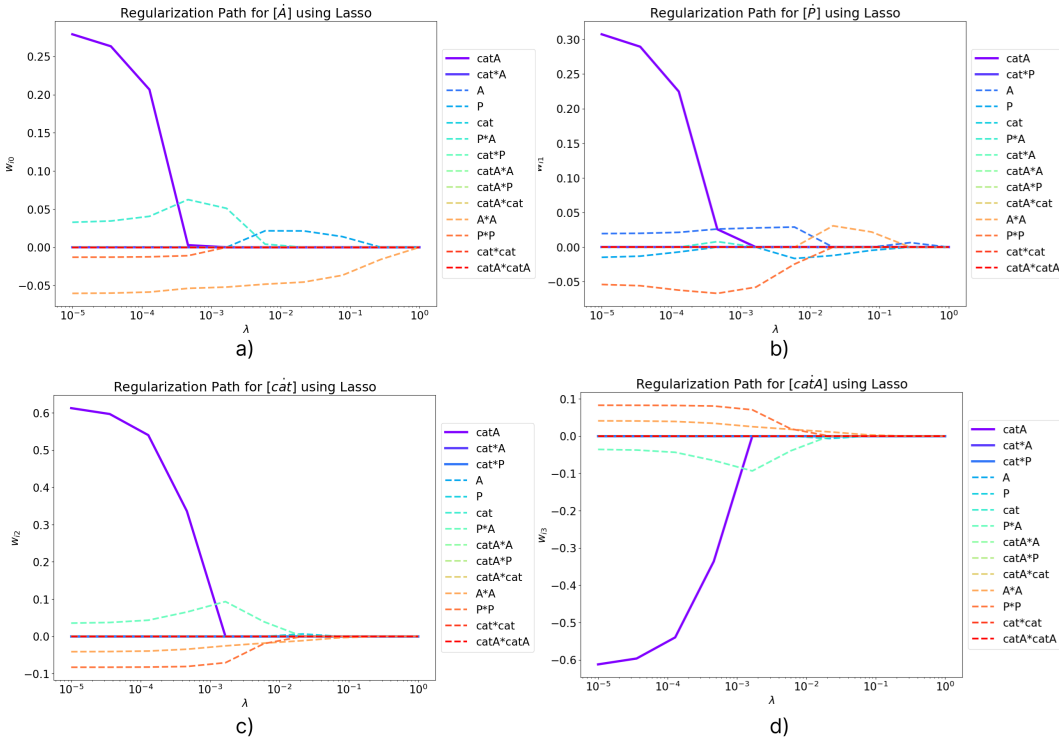

Figure B.7. Regularization paths for Lasso (equispaced points, finite difference approximation, $k_{-2} = 1$, $[\text{cat}]_0 = [0.05; 0.1; 0.15]$): a) $[\dot{\text{A}}]$ equation; b) $[\dot{\text{P}}]$ equation; c) $[\dot{\text{cat}}]$ equation; d) $[\dot{\text{catA}}]$ equation

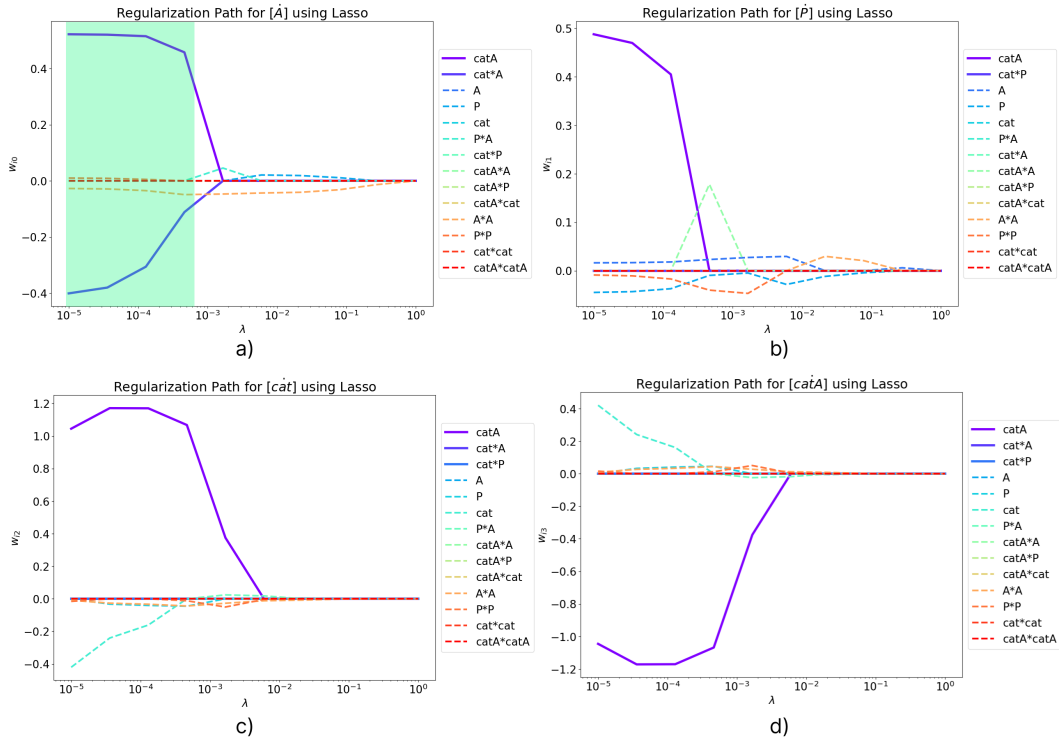

Figure B.8. Regularization paths for STLSQ (equispaced points, spline approximation, $k_{-2} = 1$, $[\text{cat}]_0 = 0.1$): a) $[\dot{\text{A}}]$ equation; b) $[\dot{\text{P}}]$ equation; c) $[\dot{\text{cat}}]$ equation; d) $[\dot{\text{catA}}]$ equation

RKF45-chosen points:

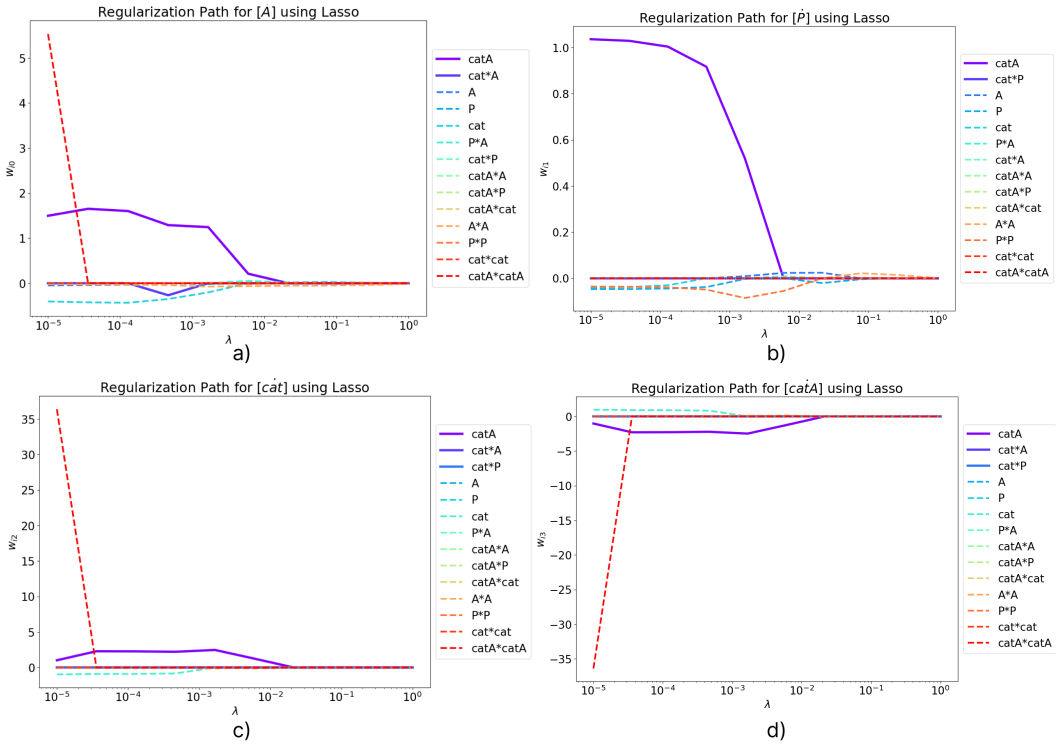

Figure B.9. Regularization paths for Lasso (RKF45 points, finite difference approximation, $k_{-2} = 1$, $[\text{cat}]_0 = [0.05; 0.1; 0.15]$): a) $[\dot{\text{A}}]$ equation; b) $[\dot{\text{P}}]$ equation; c) $[\dot{\text{cat}}]$ equation; d) $[\dot{\text{catA}}]$ equation

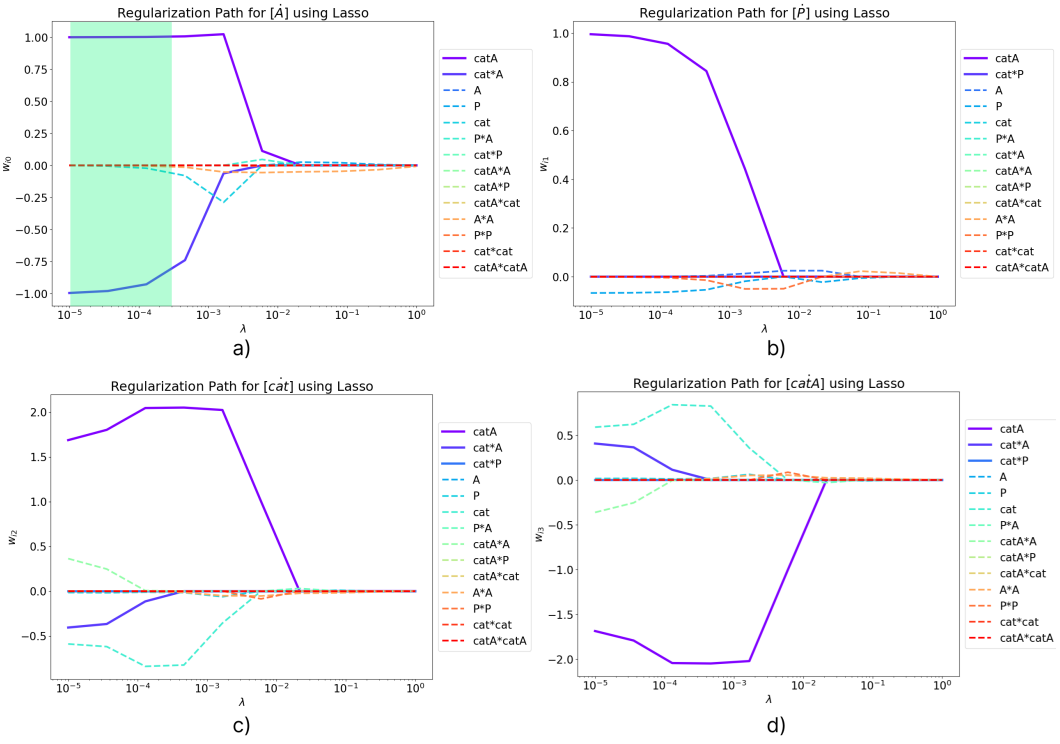

Figure B.10. Regularization paths for Lasso (RKF45 points, spline approximation, $k_{-2} = 1$, $[\text{cat}]_0 = 0.1$): a) $[\dot{\text{A}}]$ equation; b) $[\dot{\text{P}}]$ equation; c) $[\dot{\text{cat}}]$ equation; d) $[\dot{\text{catA}}]$ equation

Chebyshev points:

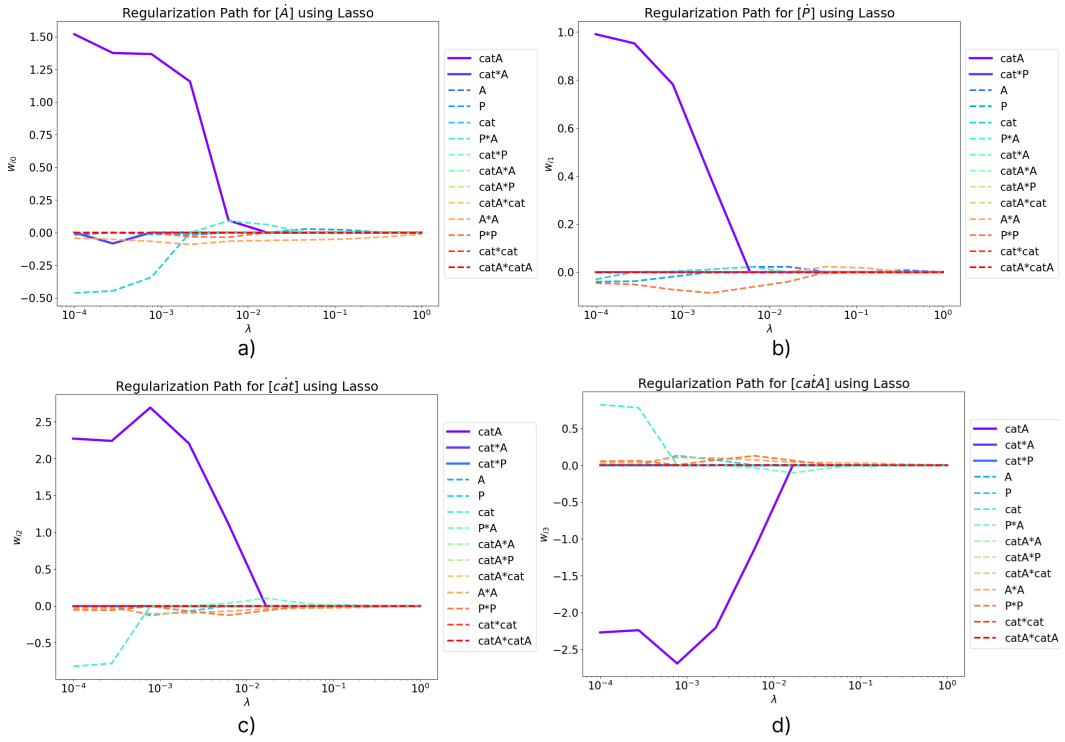

Figure B.11. Regularization paths for Lasso (Chebyshev points, finite difference approximation, $k_{-2} = 1$, $[\text{cat}]_0 = [0.05; 0.1; 0.15]$): a) $[\dot{\text{A}}]$ equation; b) $[\dot{\text{P}}]$ equation; c) $[\dot{\text{cat}}]$ equation; d) $[\dot{\text{catA}}]$ equation

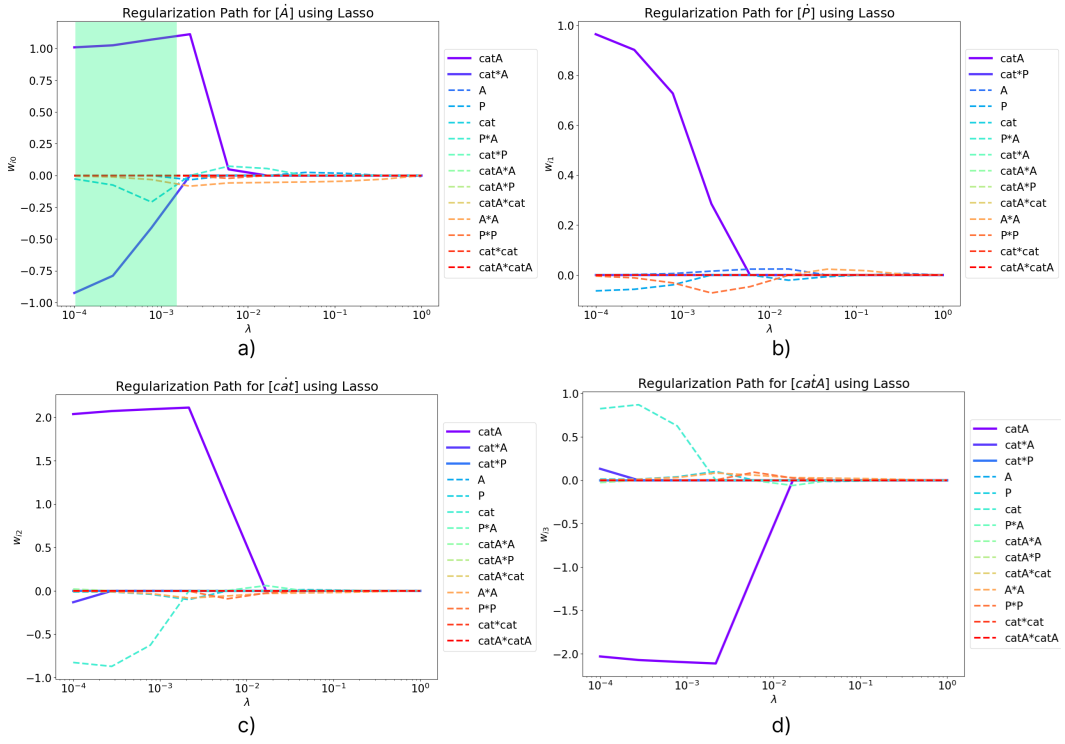

Figure B.12. Regularization paths for Lasso (Chebyshev points, spline approximation, $k_{-2} = 1$, $[\text{cat}]_0 = 0.1$): a) $[\dot{\text{A}}]$ equation; b) $[\dot{\text{P}}]$ equation; c) $[\dot{\text{cat}}]$ equation; d) $[\dot{\text{catA}}]$ equation