Exploring the HIsarna process using SINDy-CRN

Jasper Camman, S3789608

Abstract— **This research aims to expand the current understanding of the SINDy-CRN algorithm by incorporating input, output, and noisy data. The influence of these additional parameters on the algorithm's performance will be thoroughly examined, leading to insightful recommendations. To validate the effectiveness of these recommendations, they will be rigorously tested and verified using the chemical reaction network of the HIsarna process. The SINDy-CRN algorithm, which combines the Sparse Identification of Nonlinear Dynamics (SINDy) method with chemical reaction networks (CRNs), has shown promise in modeling and analyzing complex dynamical systems. However, the algorithm's existing knowledge base primarily focuses on clean and well-controlled data. By introducing input, output, and noisy data into the SINDy-CRN algorithm, this research will investigate how these factors impact its performance. The study will explore the extent to which input variables affect the accuracy and robustness of the algorithm's predictive capability. Additionally, the effect of noisy data on the reliability of the algorithm will be examined. Based on these finding some recommendations will be made to improve the algorithm performance. To verify the effectiveness of the recommendations, the chemical reaction network of the HIsarna process will serve as a testbed. Ultimately, this research will contribute to the advancement of the SINDy-CRN algorithm by expanding its applicability to scenarios involving input, output, and noisy data.**

I. Introduction

Chemical reaction networks are widely utilized in various fields because of their fundamental role in understanding complex systems. However, comprehending the intricate mechanisms of these networks can often pose significant challenges, primarily due to their high dimensionality and non-linear nature [3][4]. To address these difficulties, the Sparse Identification of Nonlinear Dynamics (SINDy) algorithm has emerged as a powerful approach for extracting governing equations from available data. The SINDy algorithm leverages the power of sparse regression techniques to identify the most relevant terms in a system's dynamics. By iteratively testing potential terms and comparing them with available data, SINDy efficiently determines the governing equations that best describe the observed behavior of a system. This approach has shown promising results in simple reaction networks without external disturbances [7]. The SINDy algorithm has been extended to improve its performance, these types of SINDy are called SINDy-PI. However these algorithms still lack the ability to be applied on chemical reaction networks, because of this a new type of SINDy, SINDy-CRN has been proposed in paper [2].

The SINDy-PI algorithm has been extended to include chemical reaction networks. The SINDy-CRN algorithm has been proposed as an extension of the original SINDy-PI framework. By incorporating additional capabilities, the SINDy-CRN algorithm aims to handle more intricate and realistic scenarios [2].

This research aims to extend the existing knowledge and capabilities of the SINDy-CRN algorithm by applying it to data obtained from the HIsarna process. The HIsarna process is an innovative technology for ironmaking that reduces the total amount of CO_2 that is emitted during the process.

In this context, this research project seeks to push the boundaries of the SINDy-CRN algorithm by applying it to data obtained from the HIsarna process. By analyzing the data generated from this process using the SINDy-CRN algorithm, the researcher aims to extract valuable insights into the underlying dynamics and kinetics of the chemical reactions taking place and make recommendations on the improvement of the SINDy-CRN algorithm

II. Materials & Methods

A. Chemical Reaction Networks

Chemical reaction networks (CRNs) form the backbone of our understanding of chemical reactions and their dynamics. They provide a powerful framework for modeling and analyzing complex systems, ranging from biochemical networks in living cells to industrial processes and environmental reactions.

At its core, a chemical reaction network consists of a set of chemical species and the reactions that inter convert them. The dynamics of the system are governed by the rates at which these reactions occur and the concentrations of the species involved.

The behavior of chemical reaction networks can be described using mathematical equations. These equations often involve systems of ordinary differential equations or stochastic models, which capture the temporal evolution and statistical fluctuations of the species' concentrations. Analyzing these mathematical models allows us to predict the behavior of the chemical system under different conditions and gain insights into its dynamics.

B. HIsarna process

The HIsarna process is an innovative ironmaking technology that aims at reducing the energy consumption and CO_2 emission compared to traditional production methods. It has been developed by combining two well-known technologies, the cyclone converter furnace, and the smelting reduction vessel [8][9]. Traditionally, iron is produced in blast furnaces where iron ore, coke, and limestone are added in a specific ratio and heated at high temperatures [10]. Using the HIsarna process, the need for coke is eliminated as well as removing the sintering, pelletizing, and blast furnace ironmaking steps, significantly reducing the amount of CO_2 emitted [8].

The process begins with the injection of fine iron ore, pulverized coal, and fluxes into a reactor vessel called the Cyclone Converter Furnace (CCF). Inside the CCF, the injected iron ore reacts with the burned CO-rich gas from the smelting reduction vessel (SRV). The iron ore melts and forms a liquid film along the cyclone walls, under the influence of gravity the liquid film falls down into the slag layer where afterwards it is exposed to oxygen to generate heat, which results in the reduction of iron oxide to metallic iron. The rate of the reaction is controlled by the amount of dissolved carbon in the hot metal droplets [8].

During this research, a simplified chemical reaction network is used. The simplified stoichiometric equations for the reactions occurring in the blast furnace are defined as

$$2\,\mathrm{C} + \mathrm{O2} \xrightarrow{\mathrm{k1}} 2\,\mathrm{CO} \tag{1}$$

$$2\,\mathrm{Fe2O3} + 3\,\mathrm{C} \xrightarrow{\mathrm{k2}} 4\,\mathrm{Fe} + 3\,\mathrm{CO2} \tag{2}$$

$$\mathrm{C} + \mathrm{O2} \xrightarrow{\mathrm{k3}} \mathrm{CO2} \tag{3}$$

The reaction rates of each reaction are respectively,

$$k_1 = 0.00833,\ k_2 = 0.1000,\ k_3 = 0.2250 \tag{4}$$

C. SINDy-CRN

In order to model the chemical reactions taking place inside the blast furnace we make use of an approach called Sparse Identification of Chemical Reaction Networks, or SINDy-CRN in short. This method differs from normal SINDy, and variants like implicit SINDy, and parallel and implicit SINDy (SINDy-PI) as it is based on prior knowledge of the CRN. Firstly, the minimal number of independent state variables is linked to the number of independent kinetics taking place inside the CRN. Secondly, the information about the singular value decomposition of the concentration data matrix is used to construct a library of possible functions for every independent kinetic which is straight away used in the SINDy-PI algorithm to get a sparse representation.

In the paper of [2] a nine step approach is presented for the identification of the chemical reaction network dynamics, this approach will be used in this paper for the identification of the chemical reactions. The approach calls for the following steps:

Firstly, identifying invariant relationships and concentration variables. Considering the concentration matrix C, where

$$C = [c_1(t)\ c_2(t)\ \dots\ c_i(t)] \tag{5}$$

We need to find the reaction variant form of concentration matrix D which can be obtained by performing the following calculation

$$D = C - 1_m c_0^T \tag{6}$$

Where 1_m is the m-dimensional matrix containing ones as its only element, and c_0^T being the initial concentration.

Secondly, Singular Value Decomposition (SVD) is performed on matrix D^T. The SVD of D^T is than given by

$$SVD(D^T) = USV^T = U_1S_1V_1^T + U_2S_2V_2^T \tag{7}$$

Where S = $\begin{bmatrix} S_1 & 0 \\ 0 & S_2 \end{bmatrix}$, is the SxS diagonal singular value matrix containing the first R-non zero values corresponding to S_1, and the $(S-R)$ which are the zero singular values corresponding to S_2.

By performing SVD, we can decompose a given matrix into the product of three matrices: U, S, and V^T, where U and V are orthogonal matrices, and S is a diagonal matrix containing singular values [5]. The SVD provides valuable insights into the properties and characteristics of the original matrix, allowing for efficient data compression, dimensionality reduction, and other analytical techniques [6].

Thirdly, in the singular value decomposition process, examining the singular values allows us to determine the number of non-zero and zero singular values. Let's denote the number of non-zero singular values as R, and the number of zero singular values as $(S-R)$.

Fourthly, we need to obtain matrix Q, which is U_2^T in the SVD. Additionally, we require U_2, which specifically refers to the left singular vectors corresponding to the number of zero singular values. These left singular vectors are instrumental in capturing the linear relationships and structure within the data [6].

Fifthly, the independent and dependent concentration values must be chosen such that the rank of the matrix with the dependent values, Q_d, is the same as the amount of zero singular values. Thus rank$(Q_d) = S - R$.

Sixthly, the invariant relationship $Q(c - c_0)$ must be determined.

Seventhly, the columns of the concentration matrix C must be chosen so that it represents the independent concentration values C_i.

Eighthly, the library matrix $\Phi(C_i, \dot{C}_i)$ must be constructed for every independent variable. The variables inside the matrix are chosen by looking at the reactions taking place and the aforementioned invariant relationship.

Lastly, the ninth step is to perform the identification of the sparse matrix Σ by performing the sequentially least squares method.

D. SINDy-PI

The Sparse Identification of Nonlinear Dynamics, Parallel and Implicit (SINDy-PI) algorithm is employed as the final step in the method described. Its purpose is to uncover the underlying dynamics present in a given dataset [1].

1) Working principle: To start, a matrix C is created which includes the values of the concentrations $[c_1\ c_2\ \cdots\ c_i]$ over time t, the following matrix C is constructed for this purpose

$$C = \begin{bmatrix} C^T(t_1) \\ C^T(t_2) \\ \vdots \\ C^T(t_n) \end{bmatrix} = \begin{bmatrix} c_1(t_1) & c_2(t_1) & \cdots & c_i(t_1) \\ c_1(t_2) & c_2(t_2) & \cdots & c_i(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ c_1(t_n) & c_2(t_n) & \cdots & c_i(t_n) \end{bmatrix} \tag{8}$$

Additionally, a matrix $\dot{C}$ containing the derivatives of the concentrations $[\dot{c_1}\ \dot{c_2} \cdots\ \dot{c_i}]$ over time is constructed

$$\dot{C} = \begin{bmatrix} \dot{C}^T(t_1) \\ \dot{C}^T(t_2) \\ \vdots \\ \dot{C}^T(t_n) \end{bmatrix} = \begin{bmatrix} \dot{c}_1(t_1) & \dot{c}_2(t_1) & \cdots & \dot{c}_i(t_1) \\ \dot{c}_1(t_2) & \dot{c}_2(t_2) & \cdots & \dot{c}_i(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \dot{c}_1(t_n) & \dot{c}_2(t_n) & \cdots & \dot{c}_i(t_n) \end{bmatrix} \tag{9}$$

Next, the library matrix $\Phi(C_i, \dot{C}_i)$ is constructed using the results obtained from the first eight steps of the method mentioned above. This matrix contains all the candidate functions that were discovered during the process. Furthermore, an empty matrix Ξ is created which will contain the sparse coefficients $\xi_1\ \xi_2\ \cdots\ \xi_i$.

With matrices C, $\dot{C}$, and Φ constructed, we can start with discovering the dynamics of the chemical reaction network by formulating a constrained optimization problem. The goal is to find the set of equations that best explain the observed dynamics. This optimization problem is formalized as follows

$$\min_{\Xi} = \|\Phi(C_i, \dot{C}_i - \Phi(C_i, \dot{C}_i)\Xi\|_2 + \beta\|\Xi\|_0, \tag{10}$$

$$\text{s.t. } \operatorname{diag}(\Xi) = 0$$

The objective of the optimization is to minimize the squared difference between the derivatives of the concentrations and the product of the library matrix and the sparse coefficient matrix, while simultaneously promoting sparsity in the solution [1].

By solving this constrained optimization problem, the SINDy-PI algorithm identifies the most appropriate set of equations that describe the dynamics of the chemical reaction network, effectively revealing the underlying behavior of the system [1].

An overview of the algorithm is shown in figure 1.

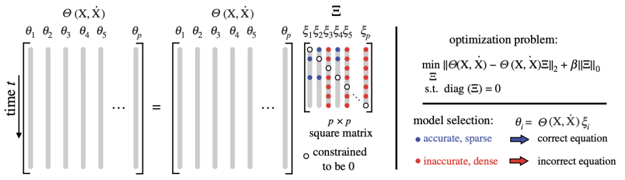

Θ (X, Ẋ)
θ_1 θ_2 θ_3 θ_4 θ_5 θ_p
time t
Ξ
$\xi_1 \xi_2 \xi_3 \xi_4 \xi_5$ ξ_p
$p \times p$ square matrix
constrained to be 0
optimization problem:
$\min_{\Xi} \|\Theta(X, \dot{X}) - \Theta(X, \dot{X})\Xi\|_2 + \beta\|\Xi\|_0$
s.t. diag (Ξ) = 0
model selection:
$\theta_i = \Theta(X, \dot{X})\xi_i$
accurate, sparse
correct equation
inaccurate, dense
incorrect equation

Fig. 1: Schematic illustration of SINDy-PI [1]

E. Tools & Methods

To analyze the discovered dynamics, we will use a method that involves calculating the relative error over time. This approach allows us to assess the accuracy of our findings. The relative error is determined by employing the following formula, which quantifies the deviation between the observed values and the expected values:

$$Error_{C_i} = \left| \frac{C_i(t) - \hat{C}_i(t)}{C_i(t)} \right| \times 100 \tag{11}$$

Research Method, To properly understand the dynamics of the HIsarna process using SINDy-CRN, the research will start with expanding existing knowledge about the SINDy-CRN because previously it has only been used on data without input, output, and no noise. Afterwards, the algorithm will be applied to the CRN of the HIsarna process and the dynamics will be evaluated and discussed. If this approach works, the dynamics will be made more realistic and complicated using inputs and outputs, otherwise the root cause of the problem will be discovered and elaborated on.

III. Results

A. Simulation 0

For the first simulation, we will expand the existing knowledge of the SINDy-CRN approach by expanding the CRN with inputs, outputs, and noisy data. The CRN that is evaluated will be the same as used in the paper [2], which exists of the following chemical reactions, and variables:

$$\mathrm{A} + 2\mathrm{B} \xrightarrow{\mathrm{r1}} \mathrm{C} \xleftrightarrow{\mathrm{r2}} \mathrm{D} \tag{12}$$

Where

$$r_1 = \frac{k_1 c_a c_b}{k_2 + k_3 c_a c_B} \tag{13}$$

And

$$r_2 = \frac{k_4 c_4}{k5 + c_c} - \frac{k_6 c_d}{k_7 c_d} \tag{14}$$

are the reaction rates for each reaction. Where k_i is respectively: 0.5, 3.5, 1.5, 2, 5, 1.5, and 6.

The concentrations of the different species is noted as

$$c = \begin{bmatrix} [A] \\ [B] \\ [C] \\ [D] \end{bmatrix} = \begin{bmatrix} c_a \\ c_b \\ c_c \\ c_d \end{bmatrix} \tag{15}$$

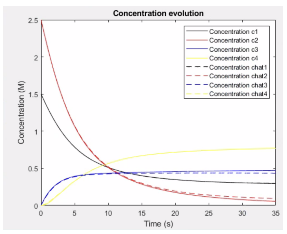
The initial concentration c_0 is defined as

$$c_0 = [1.5,\ 2.5,\ 0,\ 0]^T \tag{16}$$

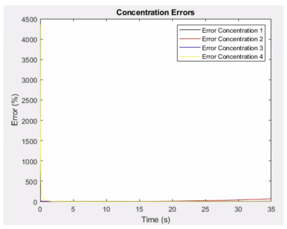
1) *Input 1:*

$$Input = [0.001,\ 0.001,\ 0,\ 0] \tag{17}$$

The first input variables influence the discovered dynamics, as can be seen in figure 2. Initially looking at the evolution of the concentrations, the graphs look fairly similar but when looking at the relative errors, there is a big spike in the first second of the graph. This spike can be explained by looking closely at the concentration evolution, as the discovered concentration of c_4 rises much quicker than the original. Because this concentration starts at zero, the relative error will be much greater when the discovered dynamics rise faster than the original dynamics.



(a) Evolution of concentration



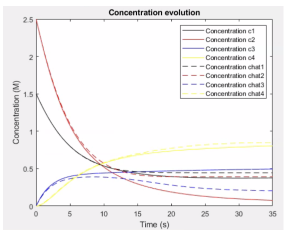
(b) Relative concentration error

Fig. 2: Concentrations Input 1

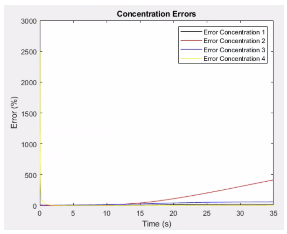
2) *Input 2:*

$$Input = [0.005,\ 0.005,\ 0,\ 0] \tag{18}$$

In the second simulation, the input variables were deliberately raised to assess the algorithm's sensitivity to such changes. It became evident that the algorithm faced difficulties in identifying the accurate dynamics of the original system under these increased input values. Furthermore, a prominent observation was the stark contrast in performance between the independent and dependent variables. The independent variables consistently outperformed the dependent variables, showing their superior predictive capability. This shows the significance of carefully evaluating and selecting the input variables to ensure the algorithm's ability to capture the desired system behavior accurately, particularly when faced with significant variations in input.



(a) Evolution of concentration



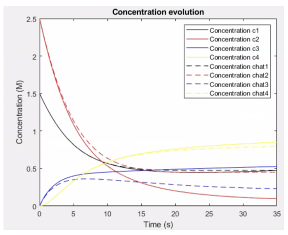
(b) Relative concentration error

Fig. 3: Concentrations Input 2

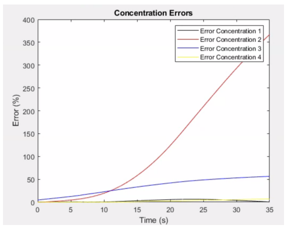
3) *Input 3:*

$$Input = [0.01,\ 0.01,\ 0,\ 0] \tag{19}$$

Lastly, upon raising the input values to values mentioned in (19), an analysis of the graphs and table reveals that the algorithm exhibits superior performance compared to the second input simulation. However, a notable observation is the presence of a significant spike in relative error of the concentration c_4. Once again, it is worth mentioning that the independent variables outperform the dependent variables in terms of accuracy and consistency. This discrepancy emphasizes the importance of carefully considering the input variables' influence on the algorithm's predictive capabilities,.



(a) Evolution of concentration



(b) Relative concentration error

Fig. 4: Concentrations Input 3

To illustrate and see the connection between increasing the input and the change in relative error, the values are shown at time, t(5), t(15), t(25), and t(35) in the following tables

Relative error c_1			
	Input 1	Input 2	Input 3
Relative Error t(5)	0.0685	0.1027	0.0751
Relative Error t(15)	0.3769	5.3193	3.0051
Relative Error t(25)	0.0129	17.9878	6.1009
Relative Error t(35)	0.3887	18.4933	0.3994

Table 1: Relative Error c_1, Input

Relative error c_2			
	Input 1	Input 2	Input 3
Relative Error t(5)	0.5684	2.1632	4.4652
Relative Error t(15)	4.1798	40.9175	57.9149
Relative Error t(25)	21.9624	201.6532	209.0132
Relative Error t(35)	67.8819	412.8429	366.5858

Table 2: Relative Error c_2, Input

Relative error c_3			
	Input 1	Input 2	Input 3
Relative Error t(5)	1.7813	7.7419	12.3292
Relative Error t(15)	3.0856	24.8907	33.0378
Relative Error t(25)	5.5960	50.9800	48.5592
Relative Error t(35)	7.8450	58.6996	56.4458

Table 3: Relative Error c_3, Input

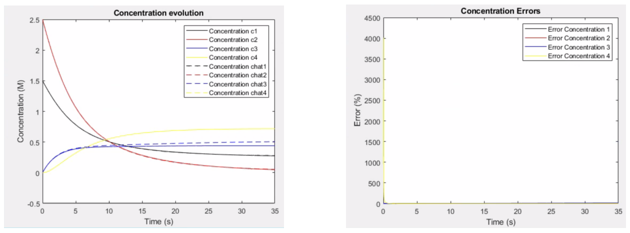
Relative error c_4			
	Input 1	Input 2	Input 3
Relative Error t(5)	0.4477	1.9851	0.0079
Relative Error t(15)	0.0425	2.3183	1.1796
Relative Error t(25)	0.1131	6.7705	4.1290
Relative Error t(35)	0.1043	5.7313	6.4352

Table 4: Relative Error c_4, Input

4) Output 1:

$$Output = [0,\ 0,\ 0.001,\ 0.001] \tag{20}$$

Looking at the simulation for the first output, the spike in relative error for concentration c_4 is again very noticeable. But overall the algorithm is still able to discover the dynamics fairly well. Also it is again very present that the independent variables perform much better than the dependent variables.



(a) Evolution of concentration

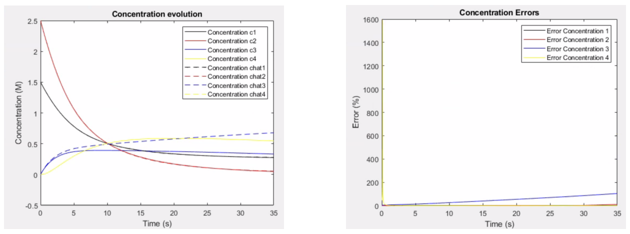
(b) Relative concentration error

Fig. 5: Concentrations Output 1

5) Output 2:

$$Output = [0,\ 0,\ 0.005,\ 0.005] \tag{21}$$

The second simulation shows that the algorithm is very sensitive for a change in output value, as can be seen in figure 6b, the discovered dynamics keep rising while the actual dynamics decrease in value. Again just as in the previous simulation the spike in relative error for concentration c_4 is very noticeable.



(a) Evolution of concentration

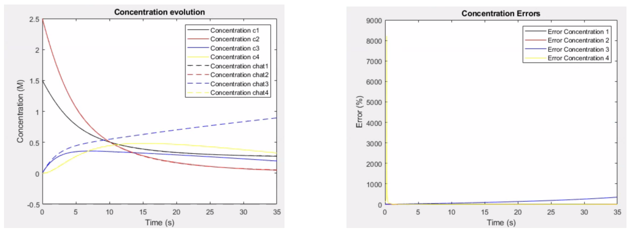
(b) Relative concentration error

Fig. 6: Concentrations Output 2

6) Output 3:

$$Output = [0,\ 0,\ 0.01,\ 0.01] \tag{22}$$

Lastly, the third simulation again shows the sensitivity of the algorithm as the relative error of concentration c_3 increases even further.



(a) Evolution of concentration

(b) Relative concentration error

Fig. 7: Concentrations Output 3

Relative error c_1			
	Output 1	Output 2	Output 3
Relative Error t(5)	0.1153	0.1164	0.1160
Relative Error t(15)	0.5657	0.5578	0.5782
Relative Error t(25)	0.2029	0.1993	0.2102
Relative Error t(35)	0.9949	0.9957	0.9954

Table 5: Relative Error c_1, Output

Relative error c_2			
	Output 1	Output 2	Output 3
Relative Error t(5)	0.1694	0.1710	0.1704
Relative Error t(15)	1.5624	1.5407	1.5973
Relative Error t(25)	1.1247	1.1047	1.1650
Relative Error t(35)	11.1915	11.2007	11.1995

Table 6: Relative Error c_2, Output

Relative error c_3			
	Output 1	Output 2	Output 3
Relative Error t(5)	2.3232	13.2137	27.6268
Relative Error t(15)	7.2681	39.6266	92.6568
Relative Error t(25)	11.4753	69.3979	188.5538
Relative Error t(35)	15.2772	103.9990	350.1838

Table 7: Relative Error c_3, Output

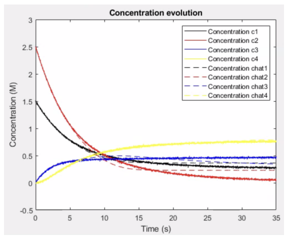
Relative error c_4			
	Output 1	Output 2	Output 3
Relative Error t(5)	0.0254	0.0320	0.9697
Relative Error t(15)	0.1296	0.0442	0.0877
Relative Error t(25)	0.0335	0.1729	0.4502
Relative Error t(35)	0.0342	0.2633	0.3003

Table 8: Relative Error c_4, Output

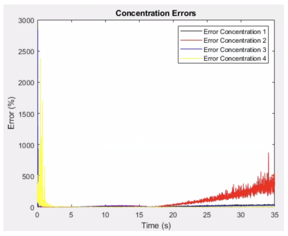
7) Noise 1: For the first simulation the data will be made noisy, this will be achieved by adding a noise variable as is stated in equation (23), the noise is set to 0.01.

$$Input = Input + Noise \tag{23}$$

Adding noise into the system reduces the predictive capability of the SINDy-CRN algorithm, the predicted dynamics severely alter from the initial system.



(a) Evolution of concentration

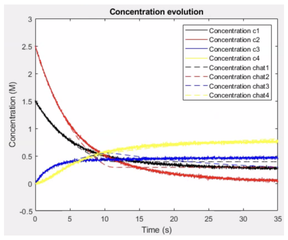


(b) Relative concentration error

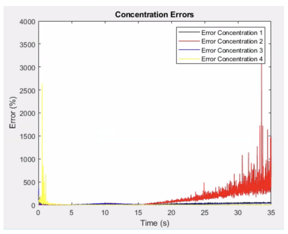
Fig. 8: Concentrations with Noise 1

8) Noise 2: Secondly, the noise is increased further and set to 0.015

Looking at the results of the predicted dynamics in figure 9 and tables 9-12, it can be seen that the relative error increases even further. Also still very prominent is the big spike in relative error of the fourth concentration. Furthermore the dependent variables again perform much worse than the independent variables.



(a) Evolution of concentration



(b) Relative concentration error

Fig. 9: Concentrations with Noise 2

Relative error c_1		
	Output 1	Output 2
Relative Error t(5)	1.0428	1.0673
Relative Error t(15)	7.0483	3.4300
Relative Error t(25)	17.0676	21.3759
Relative Error t(35)	40.4061	31.4798

Table 9: Relative Error c_1, Noise

Relative error c_2		
	Output 1	Output 2
Relative Error t(5)	0.5496	3.5000
Relative Error t(15)	16.6476	4.7929
Relative Error t(25)	104.3032	147.3930
Relative Error t(35)	332.9589	1495.66

Table 10: Relative Error c_2, Noise

Relative error c_3		
	Output 1	Output 2
Relative Error t(5)	2.4745	8.1936
Relative Error t(15)	11.1492	18.1927
Relative Error t(25)	11.2351	25.2865
Relative Error t(35)	27.1437	36.4941

Table 11: Relative Error c_3, Noise

Relative error c_4		
	Output 1	Output 2
Relative Error t(5)	4.8437	1.7824
Relative Error t(15)	4.6070	5.6682
Relative Error t(25)	2.8064	0.5583
Relative Error t(35)	4.8305	3.0111

Table 12: Relative Error c_4, Noise

9) Recommendations: Looking at these initial simulations, there are a few key takeaways from the obtained data. Firstly, when having a simulation start with concentration close to zero or at zero, it can provide a big error in the beginning stages of the predicted dynamics, this happens because the increase from the predicted dynamics is larger than the original dynamics. Secondly, the independent variables always outperform the dependent variables, if the predicted dynamics need to have the smallest error as possible it can be usefull to calculate more independent variables than necessary. And thirdly, when the data is very noisy, and/or not smooth, a smoothing algorithm can be applied to alter the data before passing it through the SINDy-CRN algorithm for better performance.

In conclusion, the following key takeaways need to be taken into account:

1) Take extra notice of dynamics starting at zero
2) Use more independent variables when the error gets to high
3) Use smoothing algorithm beforehand for noisy data

B. *Simulation 1*

In order to discover the dynamics of the reaction using the SINDy-CRN approach we need to use the steps mentioned above. First, we will test the model using generated data with no input and/or output to check how the model will work and respond using the simplified chemical reaction network of the HIsarna process.

1) Initial parameters: The initial parameters of this simulation are the following, the reaction rates are stated in equation (4), the concentrations of the different materials present inside the CRN are

$$c = \begin{bmatrix} [CO] \\ [CO_2] \\ [O_2] \\ [Fe] \\ [C] \\ [Fe_2O_3] \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{bmatrix} \tag{24}$$

The initial concentration c_0 for the different materials is

$$c_0 = [0,\ 0,\ 2,\ 0,\ 2,\ 1]^T \tag{25}$$

2) Library formation: Performing the first calculations, the singular values of the CRN are shown to be 161.7439, 7.4197, 0.0783, 0, 0, 0. This implies that the rank of the matrix D is 3, checking this by calculating the rank of matrix D we discover that this is in fact the case.

The rank of a matrix is the maximum number of linearly independent rows or columns it contains [5]. In this context, the rank of matrix D represents the number of significant reactions or dynamics in the chemical system. Since the rank of D is 3, it suggests that the CRN consists of three important reactions that contribute significantly to the overall behavior of the system.

The matrix corresponding to the last zero singular values is

$$Q = U_2^T = \begin{bmatrix} -0.0261 & -0.5974 & 0.3402 & -0.3603 & 0.6235 & 0.0901 \\ -0.0137 & 0.0687 & -0.6257 & -0.7518 & -0.0550 & 0.1880 \\ -0.8160 & 0.4264 & -0.0004 & 0.0241 & 0.3895 & -0.0060 \\ 0.4528 & 0.5577 & 0.2099 & -0.2800 & 0.3478 & -0.4904 \\ -0.1078 & 0.2254 & 0.6664 & -0.4091 & -0.4410 & 0.3628 \\ 0.3416 & 0.3077 & -0.679 & 0.2420 & 0.3755 & 0.7645 \end{bmatrix} \tag{26}$$

The different partitions of Q are examined and based on this the dependent variables are chosen to be c_1, c_2, and c_6. This leaves the dependent variables matrix to be the following

$$Q_d = \begin{bmatrix} 0.3402 & -0.3603 & 0.6235 \\ -0.6257 & -0.7518 & -0.0550 \\ -0.0004 & 0.0241 & 0.3895 \\ 0.2099 & -0.2800 & 0.3478 \\ 0.6664 & -0.4091 & -0.4410 \\ -0.679 & 0.2420 & 0.3755 \end{bmatrix} \tag{27}$$

The rank of Q_d corresponds to the number of zero singular values, thus the dependent variables c_3, c_4, and c_5 are proven to be a good choice.

Next, the libraries for the above-mentioned dependent variables need to be chosen. Looking at the chemical reactions the first reaction will depend only on c_3 (O_2), for this library the powers are up to the third degree and its time derivative. The second reaction generates c_4 (Fe) will also contain only itself to the third degree and its time derivative. Lastly, the third reaction is only dependent on the concentration of c_5 (C) thus the library will only consist of c_5 up to the third degree and its time derivative. Thus the libraries for the dependent variables will be

$$Lib3 = [c_3 \ \dot{c}_3 \ 1 \ c_3^2 \ c_3^3] \tag{28}$$

$$Lib4 = [c_4 \ \dot{c}_4 \ 1 \ c_4^2 \ c_4^3] \tag{29}$$

$$Lib5 = [c_5 \ \dot{c}_5 \ 1 \ c_5^2 \ c_5^3] \tag{30}$$

Finally, the threshold parameter is chosen to be 0.07. Now we have all the information needed to perform the sequentially threshold least-squares method to identify the dynamic equations.

The dynamic equations that are obtained using this method are the following

Concentration $\dot{c}_3$

$$\dot{c}_3 = \frac{4366c_3^2 - 2880c_3}{10000c_3 - 32381} \tag{31}$$

Concentration $\dot{c}_4$

$$\dot{c}_4 = \frac{14037c_4^2}{5000} - \frac{59267c_4}{10000} + \frac{6143}{2000} \tag{32}$$

Concentration $\dot{c}_5$

$$\dot{c}_5 = \frac{8202c_5^2 + 4655c_5}{10000c_5 - 32094} \tag{33}$$

Using these equations the dynamic equations of the dependent kinetics can be calculated. The other kinetics are calculated as

Concentration $\dot{c}_2$

$$\dot{c}_2 = \dot{c}_5 + (-2\dot{c}_3 + 1.5\dot{c}_4) \tag{34}$$

$$\dot{c}_2 = \frac{42111c_4^2}{10000} - \frac{177801 * c_4}{20000} + \frac{8202c_5^2 + 4655c_5}{10000c_5 - 32094} + \frac{5760c_3 - 8732c_3^2}{10000c_3 - 32381} + \frac{18429}{4000} \tag{35}$$

Concentration $\dot{c}_1$

$$\dot{c}_1 = \dot{c}_5 - \dot{c}_2 \tag{36}$$

$$\dot{c}_1 = \frac{177801c_4}{20000} - 2 * \frac{-4366c_3^2 + 2880c_3}{10000c_3 - 32381} - 2 * \frac{8202c_5^2 + 4655c_5}{10000c_5 - 32094} - \frac{42111c_4^2}{10000} - \frac{18429}{4000} \tag{37}$$

Concentration $\dot{c}_6$

$$\dot{c}_6 = -\frac{1}{4}\dot{c}_4 \tag{38}$$

$$\dot{c}_6 = \frac{59267c_4}{40000} - \frac{143037c_4^2}{20000} - \frac{6143}{8000} \tag{39}$$

3) Graphs of the dynamics: Plotting the found dynamics together with the initial dynamics gives the following figure

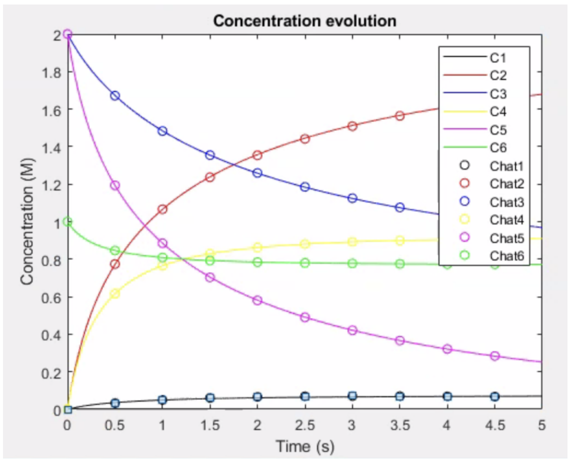
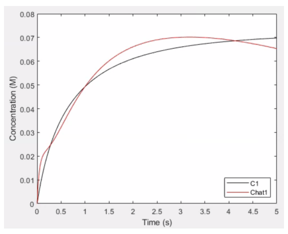


Fig. 10: Discovered dynamics and original dynamics

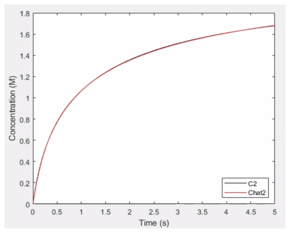
As can be seen the discovered dynamics $Chat_i$ and the original dynamics C_i are the same.

Relative errors c_1-c_6			
	c_1	c_2	c_3
Relative Error t(1)	0.1281	0.0593	0.0677
Relative Error t(3)	6.2685	0.2312	0.1494
Relative Error t(5)	6.3677	0.1678	0.0742
	c_4	c_5	c_6
Relative Error t(1)	0.0594	0.0785	0.0141
Relative Error t(3)	0.0373	0.1515	0.0107
Relative Error t(5)	0.0179	0.6461	0.0053

Table 9: Relative Errors c_1-c_6

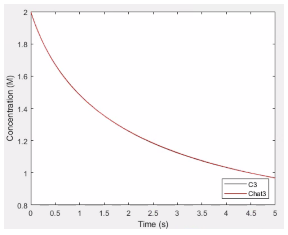


(a) Concentrations C1, and Chat1

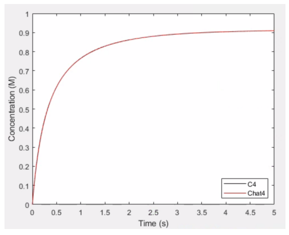


(b) Concentrations C2, and Chat2

Fig. 11: Concentrations C, and Chat

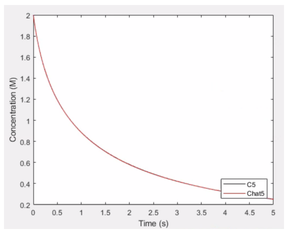


(a) Concentrations C3, and Chat3

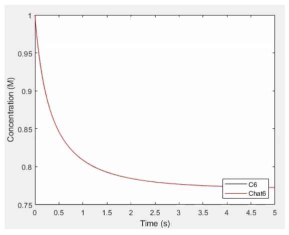


(b) Concentrations C4, and Chat4

Fig. 12: Concentrations C, and Chat



(a) Concentrations C5, and Chat5



(b) Concentrations C6, and Chat6

Fig. 13: Concentrations C, and Chat

IV. Simulation 2

Next, the simulation will be expanded by including input, and output. The knowledge and recommendations from simulation 0 and simulation 1 will be applied on this simulation.

1) Parameters: The parameters are chosen to be the following:

$$flux = 0.1 \tag{40}$$

Input variables:

$$u_1 = 3,\ u_2 = 3,\ u_3 = 0.1 \tag{41}$$

Output variables:

$$COout = 0.5,\ CO2out = 0.4,\ O2out = 0,\ Feout = 2 \tag{42}$$

Initial concentrations:

$$c_0 = [0,\ 0,\ 3,\ 0,\ 2,\ 1]^T \tag{43}$$

2) Discovered dynamics: For this simulation the discovered dynamics by applying the SINDy-CRN algorithm are the following:

Concentration $\dot{c}_1$

$$\begin{aligned}\dot{c}_1 = {} & \frac{13641c4^2}{2500} + \frac{51183c4}{5000} + \frac{27646c3 - 56250}{10000c3 - 40440} \\ & - \frac{18656c5^2 + 23772c5 - 9536}{10000c5 - 34773} - \frac{4707}{10000}\end{aligned} \tag{44}$$

Concentration $\dot{c}_2$

$$\begin{aligned}\dot{c}_2 = {} & \frac{9328c5^2 + 11886c5 - 4768}{10000c5 - 34773} - \frac{2 * (13823c3 - 28125)}{10000c3 - 40440} \\ & - \frac{51183c4}{5000} - \frac{13641c4^2}{2500} + \frac{47079}{10000}\end{aligned} \tag{45}$$

Concentration $\dot{c}_3$

$$\dot{c}_3 = \frac{13823c3 - 28125}{10000c3 - 40440} \tag{46}$$

Concentration $\dot{c}_4$

$$\dot{c}_4 = \frac{15693}{5000} - \frac{17061 * c4}{2500} - \frac{4547c4^2}{1250} \tag{47}$$

Concentration $\dot{c}_5$

$$\dot{c}_5 = \frac{9328c5^2 + 11886c5 - 4768}{10000c5 - 34773} \tag{48}$$

Concentration $\dot{c}_6$

$$\dot{c}_6 = \frac{4547c4^2}{5000} + \frac{17061c4}{10000} - \frac{15693}{20000} \tag{49}$$

As can be seen in the graphs stated below, the performance of the algorithm has decreased by implementing the different input, and output variables into the system. What is primarily noticeable is that in this simulation unlike in simulation 0, one independent kinetic (c_4 has not been identified properly where before this did happen. Because of this the identified concentration of c_6 is influenced directly and thus is also not correct.
Furthermore, the two remaining dependent variables could also not be identified correctly.

3) Graphs: The graphs of the initial concentrations over time and the discovered dynamics are shown below.

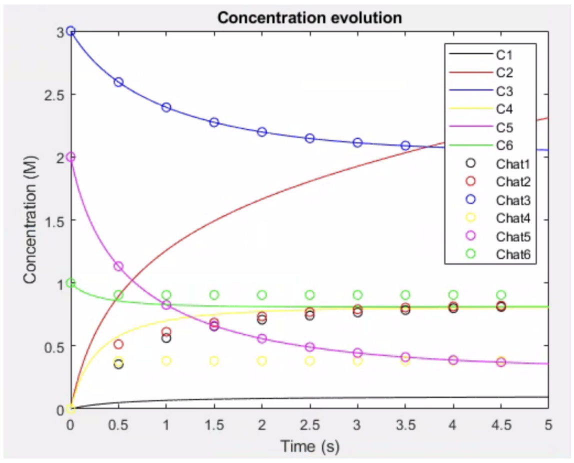
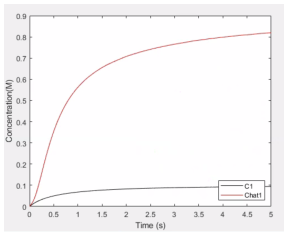
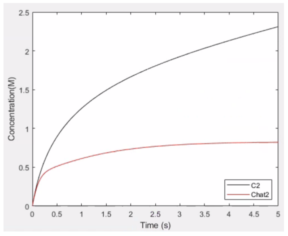


Fig. 14: Discovered dynamics and original dynamics, sim 2

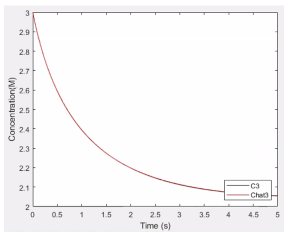


(a) Concentrations C1, and Chat1

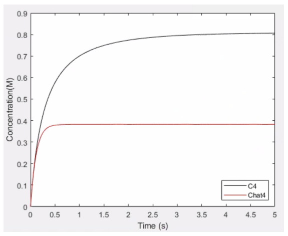


(b) Concentrations C2, and Chat2

Fig. 15: Concentrations C, and Chat

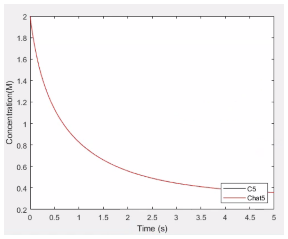


(a) Concentrations C3, and Chat3

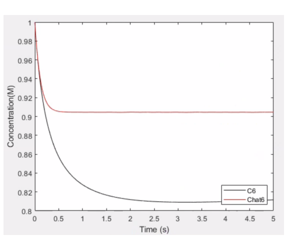


(b) Concentrations C4, and Chat4

Fig. 16: Concentrations C, and Chat



(a) Concentrations C5, and Chat5



(b) Concentrations C6, and Chat6

Fig. 17: Concentrations C, and Chat

V. Conclusion

To summarise, during this research the SINDy-CRN has been expanded to include input, output, and noisy data, the influence of these alterations has been simulated and measured and based on these results some suggestions have been made. These recommendations have than been translated into a new simulation and tested.

Looking at simulation 1, and 2, there is a clear difference in performance of the SINDy-CRN algorithm. The input and output have made drastic changes in the outcome of the simulation and identified dynamics. To counter this, more research has to be performed on the influence of input and output in the SINDy algorithm because the recommendations found did not perform properly. Also in further research, the impact of slag should be taken into account because the production of the iron is heavily dependent on this [8].

In conclusion, some steps have been taken into expanding existing knowledge of the SINDy-CRN algorithm by including input, output, and noise. The proper identification using these variables was not satisfactory, thus different approaches should be used for the identification of these dynamics.

References

[1] Kaheman, K., Kutz, J.N., Brunton, S.L. (2020). Sindy-pi: A robust algorithm for parallel implicit sparse identification of nonlinear dynamics. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 476(2242).

[2] Bhatt, N., Jayawardhana, B., Sánchez-Escalonilla, S,. (2023). SINDy-CRN: Sparse Identification of Chemical Reaction Networks from Data.

[3] Kulakowski, B.T., Gardner, J.F., Shearer, J.L. (2007). Dynamic Modeling and control of Engineering Systems. Cambridge University Press.

[4] Raol, J.R. and Ayyagari, R. (2020) Control Systems Classical, modern, and ai-based approaches. Boca Raton: CRC Press, Taylor Francis Group.

[5] Shores, T.S. (2018). Applied linear algebra and matrix analysis. Springer.

[6] He, J., Fu, Z.-F. (2001). Mathematics for modal analysis. Modal Analysis, 12–48.

[7] Zhang, L., Schaeffer, H. (2019). On the convergence of the Sindy algorithm. *Multiscale Modeling and Simulation*, 17(3), 948–972.

[8] Khasraw, D., Yan, Z., Hage, J.L., Meijer, K., Li, Z. (2022). Reduction of feo in molten slag by solid carbonaceous materials for hisarna alternative ironmaking process. *Metallurgical and Materials Transactions B*, 53(5), 3246–3261.

[9] Whiston, J., Spooner, S., Meijer, K., Li, Z. (2021). Observation of the reactions between iron ore and metallurgical fluxes for the alternative ironmaking HIsarna process. *Ironmaking & Steelmaking*, 48(10), 1142–1150.

[10] Crook, M. (2022). Iron Making: Exploring Traditional and Innovative Techniques for Sustainable Production. *Journal of Steel Structures & Construction*.

VI. Appendix

A. Code Simulation 0

Main function

```
clear all; clc; close all;
load('data_simulation.mat');

%%%%%%%%%%%%%%%%%%%%%%

% Lets obtain the original CRN
% using SINDy
```

```
%%%%%%%%%%%%%%%%%%%%%%%

size_C = size(C,1);
% Noisy Data
Noise = 0;
C_N = C + randn(size_C,4)*Noise;
C_dot_N = C_dot + randn(size_C,4)*
    Noise;

% Smooth Data
Smooth = 0;
if Smooth==1
    C_S = smoothdata(C_N,'rloess');
    C_dot_S = smoothdata(C_dot_N,'
        rloess');
    c1 = C_S(:,1);
    c2 = C_S(:,2);
    c3 = C_S(:,3);
    c4 = C_S(:,4);

    %load derivatives
    dotc1 = C_dot_S(:,1);
    dotc2 = C_dot_S(:,2);
    dotc3 = C_dot_S(:,3);
    dotc4 = C_dot_S(:,4);
else
    c1 = C_N(:,1);
    c2 = C_N(:,2);
    c3 = C_N(:,3);
    c4 = C_N(:,4);

    %load derivatives
    dotc1 = C_dot_N(:,1);
    dotc2 = C_dot_N(:,2);
    dotc3 = C_dot_N(:,3);
    dotc4 = C_dot_N(:,4);

end

%%%%%%%%%%%%%%%%%%%%%%%

dt = 0.01; % Sampling period

% Calculate derivatives using finite
    differences
% dotc1 = dot_fn(c1, dt);
% dotc2 = dot_fn(c2, dt);
% dotc3 = dot_fn(c3, dt);
% dotc4 = dot_fn(c4, dt);

% Define libraries
[nlib1, lib1] = libgen(1);
[nlib4, lib4] = libgen(4);

lib_c1= eval(lib1);
lib_c4 = eval(lib4);

size_lib1 = size(lib_c1,2); size_lib4
     = size(lib_c4,2);

index_c1 = 1:size_lib1;
index_c4 = 1:size_lib4;

lambda_sweep = 0.0:0.05:1;

lambda_sweep = 0.1;

size_sweep = size(lambda_sweep, 2);

ec1 = zeros(size_sweep, 1); % error
    array
pc1 = zeros(size_sweep, 1); %
    polynomial size array
minc1 = 99999999;
lminc1=1;
argminc1 = [];
zetaminc1 = [];

ec4 = zeros(size_sweep, 1);
pc4 = zeros(size_sweep, 1);
minc4 = 99999999;
lminc4 = 1;
argminc4 = [];
zetaminc4 = [];

%%%%%%%%%%%%%%%%%%%%%%%

% SINDY algo

%%%%%%%%%%%%%%%%%%%%%%%
% Reference values are
c1dotc1 = c1.*dotc1;
c1c4dotc4 = c1.*c4.*dotc4;

for i = 1:size_sweep
    index_c1 = 1:size_lib1;
    index_c4 = 1:size_lib4;

    lib_c1 = eval(lib1);
    lib_c4 = eval(lib4);

    prune = 1;
    while(prune == 1)
        zeta_c1 = inv(lib_c1'*lib_c1)
            *lib_c1'*(c1.*dotc1);
        zeta_c4 = inv(lib_c4'*lib_c4)
            *lib_c4'*(c1.*c4.*dotc4);


        max_zeta_c1 = max(abs(zeta_c1
            (1:end)));
        max_zeta_c4 = max(abs(zeta_c4
            (1:end)));
        prune_c1 = find(abs(zeta_c1
            (1:end))<lambda_sweep(i)*
            max_zeta_c1);
        prune_c4 = find(abs(zeta_c4
            (1:end))<lambda_sweep(i)*
            max_zeta_c4);

%          prune_c1 = find(abs(zeta_c1
    (1:end))<lambda_sweep(i));
%          prune_c4 = find(abs(zeta_c4
    (1:end))<lambda_sweep(i));


        index_c1(prune_c1) = [];
        index_c4(prune_c4) = [];

        lib_c1(:,prune_c1) = [];
        lib_c4(:,prune_c4) = [];

        if(isempty(prune_c1) == 1)&&
            (isempty(prune_c4)==1)
            prune = 0;
        end;
    end;

    % store the value that gives the
        minimum error
    error_c1 = mean(sqrt((c1dotc1-sum
        (zeta_c1'.*lib_c1, 2)).^2));
    if error_c1 < minc1
        minc1 = error_c1;
        argminc1 = index_c1;
        lminc1 = lambda_sweep(i);
        zetaminc1 = zeta_c1;
    end
    ec1(i) = error_c1;
    pc1(i) = size(zeta_c1, 1);

    error_c4 = mean(sqrt((c1c4dotc4-
        sum(zeta_c4'.*lib_c4, 2)).^2))
        ;
    if error_c4 < minc4
        minc4 = error_c4;
        argminc4 = index_c4;
        lminc4 = lambda_sweep(i);
        zetaminc4 = zeta_c4;
    end
    ec4(i) = error_c4;
    pc4(i) = size(zeta_c4, 1);
end
zetaminc1, zetaminc4
%%%%%%%%%%%%%%%%%%%%%%%

lib1 % original library
[nsparse1, mon1] = idx2mon(argminc1,
    1); % reduced library
fprintf("C1 start size %d, sparse
    size %d, lambda %d, min error %d\n
    ", nlib1, nsparse1, lminc1, minc1)
transpose(mon1) % extracted monomials

lib4 %original library
[nsparse4, mon4] = idx2mon(argminc4,
    4); % reduced library
fprintf("C4 start size %d, sparse
    size %d, lambda %d, min error %d\n
    ", nlib4, nsparse4, lminc4, minc4)
transpose(mon4) % extracted monomials

dyn = dot_sym(mon1, zetaminc1, mon4,
    zetaminc4); % put the sparse
    system together

fprintf("dotc1: \n\n"); pretty(dyn(1)
    )
fprintf("dotc4: \n\n"); pretty(dyn(2)
    )
%%%%%%%%%%%%%%%%%%%%%%%

% SOLVE THE SYSTEM

%%%%%%%%%%%%%%%%%%%%%%%

tspan = [0:0.01:35];
c0 = [1.5;2.5;0;0];
[that, c] = ode15s(@(t,c)dyn_fn(t, c,
     mon1, zetaminc1, mon4, zetaminc4)
    , tspan, c0);

% Error between initial data and
    discovered dynamics
for i=1:size_C
    Error_over_time_c1(i,1) = abs((
        C_N(i,1)-c(i,1))/C_N(i,1))
        *100;
    Error_over_time_c2(i,1) = abs((
        C_N(i,2)-c(i,2))/C_N(i,2))
        *100;
    Error_over_time_c3(i,1) = abs((
        C_N(i,3)-c(i,3))/C_N(i,3))
        *100;
    Error_over_time_c4(i,1) = abs((
        C_N(i,4)-c(i,4))/C_N(i,4))
        *100;
end

% Plot Error in concentrations
figure(1)
plot(t,Error_over_time_c1,'black')
hold on
plot(t,Error_over_time_c2,'red')
plot(t,Error_over_time_c3,'blue')
plot(t,Error_over_time_c4,'yellow')
hold off
title('Concentration Errors')
legend({'Error Concentration 1','
    Error Concentration 2','Error
    Concentration 3','Error
    Concentration 4'},'location','
    southwest')
xlabel('Time')
ylabel('Error')

% Plot Concentrations
figure(2)
plot(t,C_N(:,1),'black')
hold on
plot(t,C_N(:,2),'red')
plot(t,C_N(:,3),'blue')
plot(t,C_N(:,4),'yellow')
plot(t,c(:,1),'--red')
plot(t,c(:,2),'--black')
plot(t,c(:,3),'--yellow')
plot(t,c(:,4),'--blue')
hold off
title('Concentration evolution')
xlabel('Time')
ylabel('Concentration')

%%%%%%%%%%%%%%%%%%%%%%%

% Formulate the Library
D = C_N - ones(size_C,4).*c0.';
[U,S,V] = svd(D.');
Rank_D = rank(D);
Rank_C = rank(C_N);
U = U.';

%%%%%%%%%%%%%%%%%%%%%%%

function dotC = dot_fn(C, dt)
    % C: vector of size N containing
       the function values
    % dt: scalar representing the
       time step
    % calculate the first derivative
       using finite differences
    dotC = diff(C) / dt;

    % append the last element to the
       end to match the size of the
       input vector
    dotC = [dotC; dotC(end)];
end

function dyn = dot_sym(m1, z1, m4, z4
   )
    % combine monomials and zeta from
        the sparse identification
       into symbolic form dynamics
    syms c1 c4 dotc1 dotc4;
    eq1 = c1*dotc1-sum(z1'*m1);
    dc1 = solve(eq1, dotc1);
    eq4 = c1*c4*dotc4-sum(z4'*m4);
    dc4 = solve(eq4, dotc4);
    dyn = [collect(dc1); collect(dc4)
       ];
end

function dyn = dyn_fn(t, x, m1, z1,
   m4, z4)
    % dynamical system to be passed
       to ode solver
    c1 = x(1); c2 = x(2); c3 = x(3);
       c4 = x(4);
    dyn = eval(dot_sym(m1, z1, m4, z4
       ));
    dyn2 = 2*dyn(1); dyn3 = -dyn(1)-
       dyn(2);
    dyn = [dyn(1); dyn2; dyn3; dyn(2)
       ];
end

%%%%% Simulator for reaction systems
   CDC 2023
function y=mySimulator()
%%% Define Reaction stoichiometric
% R1: A+2B -> C and R2: C <-> D
%%% Rate expressions
% r1=k1*ca*cb/(k2+k3*ca*cb);
% r2=k4*cc/(k5+cc)-k6*cD/(k7+cD);
N=[-1 -2 1 0;0 0 -1 1];
dt = 0.01
%%% Kinetic Parameters
k1=0.5;
k2=3.5;
k3=1.5;
k4=2;
k5=5;
k6=1.5;
k7=6;
%%% Parameter vector
theta=[k1;k2;k3;k4;k5;k6;k7];

%%%% Time span
tspan=[0:dt:35];
%%% Initial concentrations
c0=[1.5;2.5;0;0];
%%% Integration
[t,C]=ode15s(@(t,c)Simulator_ode(t,c,
   N,theta),tspan,c0);

size_C = size(C,1);
C_dot = zeros(size_C,4);
```

```
for i = 1:size_C
    C_dot(i,:) = Simulator_ode(0,C(i
        ,:),N,theta);
end

%%% Plot for 4 concentrations
plot(t,C,'o');xlabel('Time [Unit]');
   ylabel('Concentrations [km m^{-3}]
   ');legend('A','B','C','D')
%%%% Save file for Identification
save data_simulation_1 t C C_dot
end


function dy=Simulator_ode(t,x,N,
   theta)
%%% Find number of ODEs
[n_r,n_c]=size(N');
%%% Initialization dy
dy=zeros(n_r,1);
%%%% Concentration variables
ca=x(1);
cb=x(2);
cc=x(3);
cD=x(4);
%%%% Parameters
k1=theta(1);
k2=theta(2);
k3=theta(3);
k4=theta(4);
k5=theta(5);
k6=theta(6);
k7=theta(7);
%%% Reaction rates
r1=k1*ca*cb/(k2+k3*ca*cb);
r2=k4*cc/(k5+cc)-k6*cD/(k7+cD);

Input = [0; 0; 0; 0];
Output = [0; 0; 0; 0];

%%% Reaction rate vector
r=[r1;r2];
%%% ODEs for 4 concentrations [ca,cb,
   cc,cD]';
dy=N'*r+Input-Output;
end
```

B. Code Simulation 1

Main function

```
clear all; clc;

% Time parameters
t0 = 0;
tend = 5;
dt = 0.001;
tspan = [t0:dt:tend];

% Initial State of the system
Concentration_CO = 0;
Concentration_CO2 = 0;
Concentration_O2 = 2;
Concentration_Fe = 0;
Concentration_C = 2;
Concentration_Fe2O3 = 1;
State0 = [Concentration_CO;
    Concentration_CO2;
    Concentration_O2;
     Concentration_Fe; Concentration_C
        ; Concentration_Fe2O3];

% Data Collecting
[t,X] = ode45(@SRV_CRN, tspan, State0
    );

size_X = size(X,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%

% Formulate the Library
D = X - ones(size_X,6).*State0.';
[U,S,V] = svd(D.');
Rank_D = rank(D);
U = U.';
Qd = [U(:,3) U(:,4) U(:,5)];
Rank_Qd = rank(Qd);

%%%%%%%%%%%%%%%%%%%%%%%%%%

% Derivative of the data
size_X = size(X,1);
dot_X = zeros(size_X,6);
for i = 1:size_X
    dot_X(i,:) = SRV_CRN(0,X(i,:));
end

%%%%%%%%%%%%%%%%%%%%%%%%%

%% Optimization
% Load data
c1 = X(:,1);
c2 = X(:,2);
c3 = X(:,3);
c4 = X(:,4);
c5 = X(:,5);
c6 = X(:,6);

% Load derivatives
dotc1 = dot_X(:,1);
dotc2 = dot_X(:,2);
dotc3 = dot_X(:,3);
dotc4 = dot_X(:,4);
dotc5 = dot_X(:,5);
dotc6 = dot_X(:,6);
```

```
% Define Libraries
Library3 = "[c3, dotc3, ones(size(c3,
     1),1), c3.^2, c3.^3]";
size_lib3 = 5;
Library4 = "[c4, dotc4, ones(size(c4,
     1),1), c4.^2, c4.^3]";
size_lib4 = 5;
Library5 = "[c5, dotc5, ones(size(c5,
     1),1), c5.^2, c5.^3]";
size_lib5 = 5;

%%%%%%%%%%%%%%%%%%%%%

lambda_sweep = 0.07;
size_sweep = size(lambda_sweep, 3);

for i = 1:size_sweep
    index_c3 = 1:size_lib3;
    index_c4 = 1:size_lib4;
    index_c5 = 1:size_lib5;

    lib_c3 = eval(Library3);
    lib_c4 = eval(Library4);
    lib_c5 = eval(Library5);

    prune = 1;
    while prune==1
        Zeta_c3 = inv(lib_c3'*lib_c3)
            *lib_c3'*(c3.*dotc3);
        Zeta_c4 = inv(lib_c4'*lib_c4)
            *lib_c4'*(c4.*dotc4);
        Zeta_c5 = inv(lib_c5'*lib_c5)
            *lib_c5'*(c5.*dotc5);

        Max_Zeta_c3 = max(abs(Zeta_c3
            (1:end)));
        Max_Zeta_c4 = max(abs(Zeta_c4
            (1:end)));
        Max_Zeta_c5 = max(abs(Zeta_c5
            (1:end)));

        prune_c3 = find(abs(Zeta_c3
            (1:end))<lambda_sweep(i)*
            Max_Zeta_c3);
        prune_c4 = find(abs(Zeta_c4
            (1:end))<lambda_sweep(i)*
            Max_Zeta_c4);
        prune_c5 = find(abs(Zeta_c5
            (1:end))<lambda_sweep(i)*
            Max_Zeta_c5);

        index_c3(prune_c3) = [];
        index_c4(prune_c4) = [];
        index_c5(prune_c5) = [];

        lib_c3(:,prune_c3) = [];
        lib_c4(:,prune_c4) = [];
        lib_c5(:,prune_c5) = [];

        if (isempty(prune_c3)==1)&& (
            isempty(prune_c4) == 1)&&
            (isempty(prune_c5)==1)
             prune = 0;
        end
    end
end

%% Reduced library
% L3 = "[c3 dotc3 c3.^2]"
% L4 = "[c4 c4.^2 c4.^3]"
% L5 = "[c5 dotc5 c5.^2]"

syms c3 dotc3 c4 dotc4 c5 dotc5
Equation3 = c3*dotc3-(-0.288*c3
    +3.2381*dotc3+0.4366*c3.^2);
Dot_c3 = solve(Equation3, dotc3);
Equation4 = c4*dotc4-(3.0715*c4
    -5.9267*c4.^2+2.8074*c4.^3);
Dot_c4 = solve(Equation4, dotc4);
Equation5 = c5*dotc5-(0.4655*c5
    +3.2094*dotc5+0.8202*c5^2);
Dot_c5 = solve(Equation5, dotc5);
dyn = [collect(Dot_c3); collect(
    Dot_c4); collect(Dot_c5)];

%% System that needs to be solved by
    ode
% 2C + O2 -> 2CO
% 2Fe2O3 + 3C -> 4Fe + 3CO2
% C + O2 -> CO2
% [CO; CO2; O2; Fe; C; Fe2O3]

dynamics_c2 = dyn(3)+(-2*dyn(1))+1.5*
    dyn(2);     % Concentration CO2
dynamics_c1 = -dyn(3)-dynamics_c2;
                  % Concentration CO
dynamics_c6 = -1/4*dyn(2);
                          %
    Concentration Fe2O3
dynamics = [dynamics_c1; dynamics_c2;
     dyn(1); dyn(2); dyn(3);
    dynamics_c6];

% Put ODE into Identified.m
% Run ODE
[that, C] = ode45(@Identified, tspan,
     State0);

for i=1:size_X
    Error_over_time_c1(i,1) = abs((X(
        i,1)-C(i,1))/X(i,1)) *100;
    Error_over_time_c2(i,1) = abs((X(
        i,2)-C(i,2))/X(i,2)) *100;
    Error_over_time_c3(i,1) = abs((X(
        i,3)-C(i,3))/X(i,3)) *100;
    Error_over_time_c4(i,1) = abs((X(
        i,4)-C(i,4))/X(i,4)) *100;
    Error_over_time_c5(i,1) = abs((X(
        i,5)-C(i,5))/X(i,5)) *100;
    Error_over_time_c6(i,1) = abs((X(
        i,6)-C(i,6))/X(i,6)) *100;
end

% Concentration Error
figure(1)
plot(t,X(:,1),'black')
hold on
plot(t,C(:,1),'red')
hold off
legend({'C1','Chat1'},'location','
    southeast')
xlabel('Time (s)')
ylabel('Concentration (M)')

% Plot Concentrations
figure(2)
plot(t,X(:,1),'black')
hold on
plot(t,X(:,2),'red')
plot(t,X(:,3),'blue')
plot(t,X(:,4),'yellow')
plot(t,X(:,5),'magenta')
plot(t,X(:,6),'green')
plot(that,C(:,1),'blacko')
plot(that,C(:,2),'redo')
plot(that,C(:,3),'blueo')
plot(that,C(:,4),'yellowo')
plot(that,C(:,5),'magentao')
plot(that,C(:,6),'greeno')
hold off
title('Concentration evolution')
xlabel('Time')
ylabel('Concentration')
```

CRN function

```
function dX = SRV_CRN(t,X)

% Dynamics
% Flux reactor
phi = 0;

% Volume
V = 19.0405;

% Inflow
u1 = 0;
u2 = 0;
u3 = 0;

% Outflow
COout = 0;
CO2out = 0;
O2out = 0;
Feout = 0;
% Constants
k1 = 0.0083;
k2 = 0.1;
k3 = 0.225;

% ODE
dX(1) = 2*(k1*X(3)*X(5).^2)-phi*COout
    /V;
dX(2) = 3*(k2*X(5).^3*X(6).^2)+(k3*X
    (3)*X(5))-(phi*CO2out/V);
dX(3) = -k1*X(3)*X(5).^2-k3*X(3)*X(5)
    -(phi*O2out/V+u1/V);
dX(4) = 4*k2*X(5).^3*X(6).^2-phi*
    Feout/V;
dX(5) = -2*k1*X(3)*X(5).^2-3*k2*X(5)
    .^3*X(6).^2-k3*X(3)*X(5)+u2/V;
dX(6) = -k2*X(5).^3*X(6).^2+u3/V;
dX = dX';
```

Identified Function

```
function dC = Identified(t,C)

dC(1) = (177801*C(4))/20000 - (2*(-
    4366*C(3)^2 + 2880*C(3)))/(10000*C
    (3) - 32381) - (2*(8202*C(5)^2 +
    4655*C(5)))/(10000*C(5) - 32094) -
     (42111*C(4)^2)/10000 -
    18429/4000;
dC(2) = (42111*C(4)^2)/10000 -
    (177801*C(4))/20000 + (8202*C(5)^2
     + 4655*C(5))/(10000*C(5) - 32094)
     + (5760*C(3) - 8732*C(3)^2)
    /(10000*C(3) - 32381) +
    18429/4000;
dC(3) = (4366*C(3)^2 - 2880*C(3))
    /(10000*C(3) - 32381);
dC(4) = (14037*C(4)^2)/5000 - (59267*
    C(4))/10000 + 6143/2000;
dC(5) = (8202*C(5)^2 + 4655*C(5))
    /(10000*C(5) - 32094);
dC(6) = (59267*C(4))/40000 - (14037*C
    (4)^2)/20000 - 6143/8000;
dC = dC';
```

C. Code Simulation 2

Main function

```
clear all; clc;

% Time parameters
t0 = 0;
tend = 5;
dt = 0.001;
tspan = [t0:dt:tend];
```

```
% Initial State of the system
Concentration_CO = 0;
Concentration_CO2 = 0;
Concentration_O2 = 3;
Concentration_Fe = 0;
Concentration_C = 2;
Concentration_Fe2O3 = 1;
State0 = [Concentration_CO;
    Concentration_CO2;
    Concentration_O2;
     Concentration_Fe; Concentration_C
        ; Concentration_Fe2O3];

% Data Collecting
[t,X] = ode45(@SRV_CRN, tspan, State0
    );

size_X = size(X,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%

% Formulate the Library
D = X - ones(size_X,6).*State0.';
[U,S,V] = svd(D.');
Rank_D = rank(D);
U = U.';
Qd = [U(:,3) U(:,4) U(:,5)];
Rank_Qd = rank(Qd);

%%%%%%%%%%%%%%%%%%%%%%%%%%

% Derivative of the data
size_X = size(X,1);
dot_X = zeros(size_X,6);
for i = 1:size_X
    dot_X(i,:) = SRV_CRN(0,X(i,:));
end

%%%%%%%%%%%%%%%%%%%%%%%%%

%% Optimization
% Load data
c1 = X(:,1);
c2 = X(:,2);
c3 = X(:,3);
c4 = X(:,4);
c5 = X(:,5);
c6 = X(:,6);

% Load derivatives
dotc1 = dot_X(:,1);
dotc2 = dot_X(:,2);
dotc3 = dot_X(:,3);
dotc4 = dot_X(:,4);
dotc5 = dot_X(:,5);
dotc6 = dot_X(:,6);

% Define Libraries
Library3 = "[c3, dotc3, ones(size(c3,
    1),1), c3.^2, c3.^3]";
size_lib3 = 5;
Library4 = "[c4, dotc4, ones(size(c4,
    1),1), c4.^2, c4.^3]";
size_lib4 = 5;
Library5 = "[c5, dotc5, ones(size(c5,
    1),1), c5.^2, c5.^3]";
size_lib5 = 5;

%%%%%%%%%%%%%%%%%%%%%

lambda_sweep = 0.07;
size_sweep = size(lambda_sweep, 3);

for i = 1:size_sweep
    index_c3 = 1:size_lib3;
    index_c4 = 1:size_lib4;
    index_c5 = 1:size_lib5;

    lib_c3 = eval(Library3);
    lib_c4 = eval(Library4);
    lib_c5 = eval(Library5);

    prune = 1;
    while prune==1
        Zeta_c3 = inv(lib_c3'*lib_c3)
            *lib_c3'*(c3.*dotc3);
        Zeta_c4 = inv(lib_c4'*lib_c4)
            *lib_c4'*(c4.*dotc4);
        Zeta_c5 = inv(lib_c5'*lib_c5)
            *lib_c5'*(c5.*dotc5);

        Max_Zeta_c3 = max(abs(Zeta_c3
            (1:end)));
        Max_Zeta_c4 = max(abs(Zeta_c4
            (1:end)));
        Max_Zeta_c5 = max(abs(Zeta_c5
            (1:end)));

        prune_c3 = find(abs(Zeta_c3
            (1:end))<lambda_sweep(i)*
            Max_Zeta_c3);
        prune_c4 = find(abs(Zeta_c4
            (1:end))<lambda_sweep(i)*
            Max_Zeta_c4);
        prune_c5 = find(abs(Zeta_c5
            (1:end))<lambda_sweep(i)*
            Max_Zeta_c5);

        index_c3(prune_c3) = [];
        index_c4(prune_c4) = [];
        index_c5(prune_c5) = [];

        lib_c3(:,prune_c3) = [];
        lib_c4(:,prune_c4) = [];
        lib_c5(:,prune_c5) = [];

        if (isempty(prune_c3)==1)&& (
            isempty(prune_c4) == 1)&&
            (isempty(prune_c5)==1)
             prune = 0;
        end
    end
end

% SIM 2
Equation3 = c3*dotc3-(1.3823*c3
    +4.0440*dotc3-2.8125);
Dot_c3 = solve(Equation3, dotc3);
Equation4 = c4*dotc4-(3.1386*c4
    -6.8244*c4.^2-3.6376*c4.^3);
Dot_c4 = solve(Equation4, dotc4);
Equation5 = c5*dotc5-(1.1886*c5
    +3.4773*dotc5-0.4768+0.9328*c5.^2)
    ;
Dot_c5 = solve(Equation5, dotc5);
dyn = [collect(Dot_c3); collect(
    Dot_c4); collect(Dot_c5)];

%% System that needs to be solved by
    ode
% 2C + 02 -> 2CO
% 2Fe2O3 + 3C -> 4Fe + 3CO2
% C + 02 -> CO2
% [CO; CO2; O2; Fe; C; Fe2O3]

dynamics_c2 = dyn(3)+(-2*dyn(1))+1.5*
    dyn(2);    % Concentration CO2
dynamics_c1 = -dyn(3)-dynamics_c2;
                  % Concentration CO
dynamics_c6 = -1/4*dyn(2);
                            %
    Concentration Fe2O3
dynamics = [dynamics_c1; dynamics_c2;
     dyn(1); dyn(2); dyn(3);
    dynamics_c6];

% Put ODE into Identified.m
% Run ODE
[that, C] = ode45(@Identified2, tspan
    , State0);

for i=1:size_X
    Error_over_time_c1(i,1) = abs((X(
       i,1)-C(i,1))/X(i,1)) *100;
    Error_over_time_c2(i,1) = abs((X(
       i,2)-C(i,2))/X(i,2)) *100;
    Error_over_time_c3(i,1) = abs((X(
       i,3)-C(i,3))/X(i,3)) *100;
    Error_over_time_c4(i,1) = abs((X(
       i,4)-C(i,4))/X(i,4)) *100;
    Error_over_time_c5(i,1) = abs((X(
       i,5)-C(i,5))/X(i,5)) *100;
    Error_over_time_c6(i,1) = abs((X(
       i,6)-C(i,6))/X(i,6)) *100;
end

Time_Plot = [0:0.1:5];
% Plot Concentrations
figure(1)
plot(t,X(:,1),'black')
hold on
plot(t,X(:,2),'red')
plot(t,X(:,3),'blue')
plot(t,X(:,4),'yellow')
plot(t,X(:,5),'magenta')
plot(t,X(:,6),'green')
plot(that,C(:,1),'blacko')
plot(that,C(:,2),'redo')
plot(that,C(:,3),'blueo')
plot(that,C(:,4),'yellowo')
plot(that,C(:,5),'magentao')
plot(that,C(:,6),'greeno')
hold off
title('Concentration evolution')
legend({'C1','C2','C3','C4','C5','C6'
    ,'Chat1','Chat2','Chat3','Chat4','
    Chat5','Chat6' },'location','
    northeast')
xlabel('Time (s)')
ylabel('Concentration (M)')

plot(t,X(:,1),'black')
hold on
plot(t,C(:,1),'red')
hold off
legend({'C1','Chat1'},'location','
    southeast')
xlabel('Time (s)')
ylabel('Concentration(M)')
```

CRN function

```
function dX = SRV_CRN(t,X)

% Dynamics
% Flux reactor
phi = 0.1;

% Volume
V = 19.0405;

% Inflow
u1 = 3;
u2 = 3;
u3 = 0.1;

% Outflow
COout = 0.5;
```

```
CO2out = 0.4;
O2out = 0;
Feout = 2;
% Constants
k1 = 0.0083;
k2 = 0.1;
k3 = 0.225;

% ODE
dX(1) = 2*(k1*X(3)*X(5).^2)-phi*COout
    /V;
dX(2) = 3*(k2*X(5).^3*X(6).^2)+(k3*X
    (3)*X(5))-(phi*CO2out/V);
dX(3) = -k1*X(3)*X(5).^2-k3*X(3)*X(5)
    -(phi*O2out/V)+u1/V;
dX(4) = 4*k2*X(5).^3*X(6).^2-phi*
    Feout/V;
dX(5) = -2*k1*X(3)*X(5).^2-3*k2*X(5)
    .^3*X(6).^2-k3*X(3)*X(5)+u2/V;
dX(6) = -k2*X(5).^3*X(6).^2+u3/V;
dX = dX';
```

Identified function

```
function dC = Identified(t,C)
dC(1) = (177801*C(4))/20000 - (2*(-
    4366*C(3)^2 + 2880*C(3)))/(10000*C
    (3) - 32381) - (2*(8202*C(5)^2 +
    4655*C(5)))/(10000*C(5) - 32094) -
     (42111*C(4)^2)/10000 -
    18429/4000;
dC(2) = (42111*C(4)^2)/10000 -
    (177801*C(4))/20000 + (8202*C(5)^2
     + 4655*C(5))/(10000*C(5) - 32094)
     + (5760*C(3) - 8732*C(3)^2)
    /(10000*C(3) - 32381) +
    18429/4000;
dC(3) = (4366*C(3)^2 - 2880*C(3))
    /(10000*C(3) - 32381);
dC(4) = (14037*C(4)^2)/5000 - (59267*
    C(4))/10000 + 6143/2000;
dC(5) = (8202*C(5)^2 + 4655*C(5))
    /(10000*C(5) - 32094);
dC(6) = (59267*C(4))/40000 - (14037*C
    (4)^2)/20000 - 6143/8000;
dC = dC';
```