Semi-automatic identification of reaction networks using Mixed-Integer Linear Programming (MILP)

A Thesis Submitted By

Christopher Dixon

For the Degree of Doctor of Philosophy

School of Engineering

Newcastle University

January 2021

Abstract

Understanding the processes taking place within a reaction network are of critical importance, particularly for new product development in the chemical processing industries. With more data from new and unknown processes being collected, an efficient method of converting this data into useful information explaining the underlying chemistry would be desirable. Traditionally the method of identification of reaction kinetics was carried out manually using the integral and differential methods. However, when multiple and/or complex reactions are taking place these methods do not compute accurate values. This work takes these manual methods and uses Mixed Integer Linear Programming (MILP) to automate the identification of the stoichiometric and kinetic models of a simulation of a simplified biodiesel production reaction network under a variety of different conditions, and an experimental dataset of the thermal decomposition of α-pinene.

The stoichiometries of the biodiesel production reaction network were identified under a variety of noise (0-5%) and measurement levels (2-75) using MILP using only concentration measurements as inputs. The results indicate that there is a minimum required number of measurements – 25 measurements for this experimental data, for the algorithm to identify stoichiometries. The quality of results also decreases once the amount of noise increases over approximately 2.5%, although it is still possible to find the stoichiometries above that point.

The identification of chemical reaction kinetics of the biodiesel production reaction network is also achieved using MILP at a varying noise levels (0-5%) and number of measurement levels (2-75). It has shown that kinetic structures can be identified with only concentration measurements as inputs. As with the stoichiometric identification algorithm, the kinetics identification requires a minimum of 10 measurements to find an accurate model, but the algorithm is most effective when there is at least 30 measurements for this experimental dataset. Similarly once the noise level exceeds approximately 2.5%, the algorithm struggles to reliably identify kinetic models.

Acknowledgement

I would like to thank my PhD supervisors: Dr Mark Willis, and Dr Chris O'Malley, with additional thanks to Dr Moritz von-Stoch for his supervision at the beginning of my PhD. Additional thanks to the Process Intensification Group (PIG) within Newcastle University's Chemical Engineering department. I am also grateful to my friends and colleagues in C415 and throughout the Department of Chemical Engineering who made my studies as enjoyable as they were challenging.

Special thanks to my family and Pichaya In-na, without their support I would not have been able to complete my studies.

Publications and conferences

Publications

- **C. J. Dixon**, C. J. O'Malley, M. J. Willis. "Identification of the structure and parameters of kinetic models using mixed integer linear programming". (In Preparation)

Conferences

- **C. J. Dixon**, C. J. O'Malley, M. J. Willis. "Identification of the structure and parameters of kinetic models using mixed integer linear programming". Keynote Lecture at the 24th International Congress Of Chemical And Process Engineering, CHISA 2020 Prague, Czech Republic. (Postponed to 2021 due to COVID-19).
- **C. J. Dixon**, C. J. O'Malley, M. J. Willis. "An MILP approach for identification of reaction networks from noisy data". Poster Presentation at the 23rd International Congress of Chemical And Process Engineering, CHISA 2018, Prague, Czech Republic.
- **C. J. Dixon**, C. J. O'Malley, M. J. Willis. "An MILP approach for identification of reaction networks from noisy data". Poster presentation at the 12th European Symposium on Biochemical Engineering Sciences (ESBES), 2019, Lisbon, Portugal.

Table of content

List of figures

List of tables

Abbreviations/Nomenclature

AIC	Akaike Information Criterion
BD	Biodiesel
CRN	Chemical Reaction Network
DG	Diglyceride
GAs	Genetic Algorithms
GL	Glycol
MeOH	Methanol
MG	Monoglyceride
MILP	Mixed Integer Linear Programming
SVD	Singular Value Decomposition
TG	Triglyceride
TFA	Target factor analysis
LASSO	Least Absolute Shrinkage and Selection Operator

1. Chapter 1 Introduction

1.1 Project background

Understanding the processes taking place within a reaction network are of critical importance, particularly for new product development in the chemical processing industries. Generally, the development of new products involves several steps of experimentation, discovering production mechanistic routes, and confirmation. This is an important step in the production of any product as it will determine the efficiency – with respect to economics and resource management, and with a sustainable, environmentally aware approach too (Blau et al., 2000). Being able to identify the chemical processes taking place quickly and accurately is an important step along the road to development. This experimental procedure would typically involve the creation of large quantities of data. With more data from new and unknown processes being collected, an efficient method of converting this data into useful information explaining the underlying chemistry would be desirable. Traditionally the method of identification of reaction kinetics was carried out manually using the integral and differential methods. However, when multiple and/or complex reactions are taking place, these methods do not compute accurate values. Machine learning and optimisation techniques have been and are being used in an attempt to take advantage of this large quantity of data to identify chemical reactions and reaction networks. This work adds to the body of knowledge by implementing Mixed Integer Linear Programming (MILP) – a well-established optimisation technique that has been implemented successfully on many of the well-known benchmark test problems, to the identification of chemical reaction networks.

1.2 Aims and Objectives

The aims of this thesis are to develop an automatic method of identifying chemical reaction networks from concentration data using a mixed integer linear programming (MILP) approach. This includes the objectives:

1. The identification of stoichiometric models of chemical reaction networks (CRNs),
2. Identification of chemical reaction kinetics – including the kinetic structure and the kinetic coefficients of CRNs.

1.3 Outline of thesis

Chapter 1 - the literature review, starts with an overview of chemical reactions, stoichiometries and kinetics. This is followed with some basic reaction engineering to introduce the idea of a

batch reactor. The literature review then continues to discuss the methods that have been attempted to automatically identify different elements of chemical reactions, and CRNs.
Chapter 1 is the method results and discussion of an MILP approach to identify stoichiometric models under varying conditions, including the changing of the amount of noise and number of measurements used.
Chapter 3 contains the method, results and discussion of an MILP approach to identify chemical kinetic structures and parameters. As with chapter 1, a variety of different conditions is considered, including noise levels and number of measurements.
Finally, Chapter 5 is the conclusions which summarises the results and conclusions of this thesis and discusses potential future work to follow on from this work.

2 Chapter 2 Literature review

2.1 Overview

In this chapter an introduction to chemical reactions is presented. Including reaction stoichiometries, reaction kinetics and kinetic parameters. This is followed by a review of the methods that have been attempted in the identification of each of these fields. The benefits and limitations of each of these methods is discussed throughout.

2.1 Introduction

Chemical reactions are a core component of the production of pharmaceuticals, high value products and new product development, among other fields. To perform these reactions efficiently and safely, the modelling of chemical reactions and the relevant processes is essential. These models need to be accurate and precise to be used to predict the change in the quantity of any component in a reaction network. A variety of optimisation techniques have been applied to try to identify CRN stoichiometries, and kinetics. Throughout this chapter some of the most well-known techniques are evaluated and compared.

2.2 Chemical Reactions

Reaction kinetics is the study of the dynamics of the change in chemical species (Aris and Mah, 1963). The study of Chemical reaction kinetics as we know it today started in the nineteenth century and was focused on empirical measurements of rates of chemical change of reactants into products(Hoff, 1884). Hoff (1884) summarised the overall reaction and is the first example of what we would today call a stoichiometric expression. However, in many cases this description of reactions is a simplified summary of what can be taking place, reaction mechanics and/or elementary reactions are often complex, and in some cases, the intermediates species can be difficult to detect without specific, sensitive sensors. Descriptions of reactions can come in a variety of forms, the two forms of interest in this work are stoichiometric descriptions and kinetic descriptions.

2.2.1 Stoichiometric Models

A Stoichiometric description often appears in the following form (Equation 2-1):

$$\nu_1 x_1 + \nu_2 x_2 \rightarrow \nu_3 x_3 \qquad \text{Equation 2-1}$$

where v_i is the stoichiometric coefficient associated with chemical species x_i. In this example x_1 and x_2 are both being consumed by the reaction to produce x_3. Traditionally The reactants of the reaction are on the left-hand side and the products are on the right-hand side with the coefficient values representing the proportion of each species relative to one another. However stoichiometric descriptions of reactions can be summarised in vector form (Equation 2-2). Similarly, there is a standard form to make the coefficients of the reactants negative, and the coefficients of the products are positive (Equation 2-2):

$$[-v_1 \quad -v_2 \quad v_3]$$ Equation 2-2

To illustrate multiple reactions, the Van de Vusse CRN (Equation 2-3) and its stoichiometric matrix (N) (Equation 2-4) are show below:

$$x_1 \rightarrow x_2$$ Equation 2-3a

$$x_2 \rightarrow x_3$$ Equation 2-3b

$$2x_1 \rightarrow x_4$$ Equation 2-3c

$$\mathrm{N} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ -2 & 0 & 0 & 1 \end{bmatrix}$$ Equation 2-4

The form shown in Equation 2-4 is the method of displaying the stoichiometry which will be used throughout this work. Where each column represents each species, and each row represents each individual, or elementary reaction of a reaction network. From this we can show that the stoichiometric matrix N is of dimensions with r reactions, and c components $(\mathrm{N}_r \times \mathrm{N}_c)$.

2.2.2 Reaction Kinetics

Reaction kinetics are a mathematical method for describing the change in quantity of a component within a reaction. The rate of reaction can be influenced by concentration, temperature, pressure, the presence of a catalyst, the physical state of the components, and light. In many instances, the effect these variables can be non-impactful on the rate of reaction, for example, the Haber process requires an iron catalyst for the operating temperatures and pressures to be viable, the rate of reaction is also heavily impacted by the temperature and pressure of the reactants.

The rate of reaction of component i, (r_i) is the number of moles of i reacting (being consumed by the reaction or being produced by the reaction) per unit time, per unit volume

($mol.dm^{-3}.s^{-1}$). Equation 2-5 shows an example of a first order reaction with the rate reaction is determined by the concentration of component j. The dimensions of the rate constant k depends on the order of the reaction, in the case of a first order reaction, the rate constant has units of s^{-1}.

$$r_i = k.\left[C_j\right]^n \qquad \text{Equation 2-5}$$

Reactions can be of zero order, first or second order, pseudo-first order, fractional orders, negative orders, or (rarely) ternary orders. It is also worth noting that it is possible to have an effective order of reaction that is different to the true order of reaction, this would typically occur when one component is in large excess so that the change in concentration of that component is negligible and therefore not impactful on the order of reaction.

The rate constant of a reaction is typically described using the Arrhenius equation shown in Equation 2-6. The Arrhenius equation identifies the rate constant based upon the temperature (T) – in Kelvin, the universal gas constant ($\boldsymbol{R}$), the activation energy of the reaction ($\boldsymbol{E_a}$), and the pre-exponential factor (A).

$$k = A.\exp\left(-\frac{\boldsymbol{E_a}}{\boldsymbol{RT}}\right) \qquad \text{Equation 2-6}$$

The pre-exponential factor is associated with the frequency that the energised reactants convert into products, and the frequency of collisions of reactants. The activation energy of the reaction is associated with the energy requirements to achieve a transitional state that will lead to the reaction taking place (Ravi et al., 2017). The activation energy of a reaction can be modified with the use of a catalyst, providing an alternate pathway for the chemical reaction to take place. Typically, the addition of a catalyst will reduce the activation energy for the reaction to take place – increases the overall value of k and therefore the rate of reaction.

2.3 Reactor Engineering – Batch Reactor

According to the law of conservation of mass, the mass balance of any reactor is described by Equation 2-7.

$$\text{Accumulation} = \begin{matrix}\text{Moles}\\\text{in}\end{matrix} - \begin{matrix}\text{Moles}\\\text{out}\end{matrix} + \begin{matrix}\text{Production}\\\text{or}\\\text{Consumption}\\\text{(Reactions)}\end{matrix} \qquad \text{Equation 2-7}$$

In the case of a batch reactor there is no inlet and no outlet, the reactants are all within the reactor at the start of the reaction i.e. Moles in $=$ Moles out $= 0$. Therefore, the rate of change of amount of any component in a reactor is the accumulation of that component, and the amount of it reacting or being produced by a reaction. This is under the assumptions that the reactor is well-mixed and at constant density. Therefore, this simplifies the rate of change in the concentration of a component to be a function of only the reactions that component is taking part in (Equation 2-8).

$$\frac{d[C]}{dt} = r_C \qquad \text{Equation 2-8}$$

This work is focused on the cases studies (biodiesel production) of batch reactors.

2.3.1 Chemical Reaction Networks

A Chemical Reaction Network (CRN) is a set of reactions that are connected. In some CRNs the product of the first reaction is the reactant of another reaction- shown in Table 3-3.This can make the identification of the CRNs more difficult- particularly when trying to identify the kinetics that describe the CRN. This is a consequence of the coupling between the reactions (Fogler, 1999).

There are a variety of computational methods that have been attempted when trying to identify models that accurately describe reaction networks. Methods of identification of reaction networks are discussed in Section 2.4.

2.4 Identification of Reactions

The identification of chemical reactions continues to be of interest to both industry and academia. Aris and Mah (1963) is a central piece of research in this field as the authors developed a framework for identification of stoichiometric coefficients without any prior knowledge about the composition of the reacting species. Using measurements of the extent of reaction they identified the number of independent reactions taking place by identifying the rank of the extents matrix. This method was extended by using the atomic matrix of the species to identify the stoichiometric coefficients. This method was limited in its application however as when noise is introduced into the measurements, the identified rank of the extents matrix becomes less reliable. This method has been built upon using multiple computational methods including target factor analysis (TFA), Linear Regression, and Mixed Integer Linear Programming (MILP).

2.4.1 Target Factor Analysis

Target Factor Analysis (and its variants) is an accepted form of stoichiometric identification which has been applied in a variety of ways to identify stoichiometries(Brendel et al., 2006, Prinz and Bonvin, 1994, Amrhein et al., 1999, Maria, 2004). TFA was first used to identify CRNs by Bonvin and Rippin (1990), they used a combination of Singular Value Decomposition (SVD) and Factor Analysis to identify stoichiometric coefficients. SVD is used to identify the number of independent reactions. Following this, a hypothesised stoichiometry, or target stoichiometry can be generated from *a priori* information or from deductive reasoning and target testing. This method is an improvement on the method developed by Aris and Mah (1963), as it does not require the atomic matrix of the species involved. However, it does still require a hypothesised stoichiometric model – this could be problematic if no expert is present to hypothesise a stoichiometric model. When the data matrix contains measurement noise, SVD can still be unreliable at identifying the correct rank of the data matrix. A methodological improvement was developed by Fotopoulos et al. (1994). This included a structured approach to the target testing stage without requiring more *a prioi* information about the reaction being identified. It defined a set of postulated stoichiometries that must include a certain number of reactants and products.

Prinz and Bonvin (1994) used incremental target factor analysis to determine and identify chemical species from their absorbance spectra, which could in turn be used to identify the extent of reaction and consequentially the reaction stoichiometries. The incremental approach proved to be a benefit when dealing with noisy data as is commonly collected. They did apply their method as part of an online monitoring system to infer concentration measurements which would otherwise be impossible to measure directly.

Amrhein et al. (1999) advanced the TFA techniques further by applying the methods to both reactions and mass transfer problems without explicit knowledge of the rate expressions of each of them. The approach also operates with considerable measurement noise present successfully and with high statistical confidence in their results. The limitation on this method is the set of proposed, possible candidate kinetic structures is relatively small and would require considerable expert knowledge to identify these structures.

Brendel et al. (2006) adapted the strategy used by the previous authors to so that the model structures did not need to be postulated. Unfortunately, the method is still sensitive to measurement noise which can result in partially identified solutions.

2.4.2 Tendency Models

Tendency models have been used both in conjunction with and as an alternative to TFA. Tendency models utilise the principles of the extent of reaction being related to concentration measurements through the appropriate stoichiometric matrix and a least squares algorithm (Filippi et al., 1986). Tendency models are an effective way of approximating models when fundamental understanding is not possible. However, the process-model mismatch is more likely to occur as the number of reactions being identified increases and/or when the number of reactions taking place is unknown (Fotopoulos et al., 1995). Fotopoulos (1998) used tendency models to approximate a state-based model of a reaction network which was used as part of a Kalman filter to control a jacketed batch reactor.

2.4.3 Mixed Integer Linear Programming

Mixed Integer Linear Programming (MILP) is an optimisation technique that searches for the optimal point within a set on constraints. MILP is a well-established technique that has been shown to be effective for a variety of standard optimisation problems including the travelling salesman problem (Radmanesh et al., 2016), in which MILP is used to find the shortest and/or fastest route a salesman should take across a map (or country). This shows the efficacy of MILP for solving complex problems which are NP-hard (non-deterministic polynomial-time) a term from the field of computer science to denote how difficult it is to both find a solution, and the difficulty of checking that solution – an NP-hard problem is difficult to identify a solution, but easy to check the solution. MILP is a well-established technique that has been used throughout all engineering industries and beyond (Richards and How, 2002, Lauinger et al., 2016). Floudas and Lin (2005) provide a summary of MILP being applied throughout the chemical processing industries, focusing primarily on scheduling problems considering both discrete and continuous time models. A variety of techniques and modifications to the standard MILP technique have been developed over the years to improve upon the original method created by George Dantzig in 1947, including computational advancements, expansions of the branch and bound method (Benichou et al., 1971, Lima and Grossmann, 2011, Floudas and Lin, 2005, Lubin et al., 2018).

Willis and Stosch (2016) applied this method to generate both stoichiometric and kinetic models of CRNs. Stoichiometric matrices were identified by generating all possibilities, then removing the infeasible reactions by applying the law of conservation of mass. i.e. all of the atomic components that appear in the reactants must appear in the products. Willis and Stosch (2016) used the example of a Van de Vusse reaction network, shown in Equation 2-9.

$$2x_1 \rightarrow x_2 \qquad \text{Equation 2-9a}$$

$$x_1 \rightarrow x_3 \qquad \text{Equation 2-9b}$$

$$x_3 \rightarrow x_4 \qquad \text{Equation 2-9c}$$

From the stoichiometric model of the Van de Vusse model, it can be deduced that x_2 must have twice the atomic mass of x_1 and x_1 has the same atomic mass as x_3. Therefore, using this logic, $x_3 \rightarrow x_2$ is an infeasible solution, but $2x_3 \rightarrow x_2$ is a feasible solution.

Willis and Stosch (2016) combined this with an integer cut technique to identify multiple solutions within the same search space. It therefore becomes possible to find entire stoichiometric networks that describe a CRN. This method requires the molecular makeup of all of the species taking place in the CRN to guarantee correct solutions, however as shown in the example of Equation 2-9 simply using the total atomic mass could provide partial information to reduce the search-space to a smaller . Langary and Nikoloski (2019) improved upon this method by using SVD to reduce the size of the reaction invariant subspace. They also implemented the MILP method at steady state rather than in a batch or semi-batch form that other methods have used.

2.5 Identification of Reaction Kinetics

The automatic identification of Chemical Reaction Kinetics continues to be of great interest to industry and academia, and a variety of methods that have been implemented to find kinetic parameters. There are a variety of methods that utilise the principles of reaction kinetics along with statistical and optimisation techniques, the three main methodologies are discussed here: Linear Regression, Genetic Algorithms, and Mixed Integer Linear programming.

2.5.1 Regression

Regression is a statistical method for identifying a linear model to find a line of best fit through to give the best approximation of a dependant variable using one or several independent variables. Regression methods are used in economics, the social sciences, throughout the sciences and engineering disciplines, and many more that are not listed for the sake of brevity. It has been used in so many fields as it is easy to apply to all sorts of data and can be easily interpreted without significant training.

There are a variety of methods that use linear regression and its variants (non-linear regression, multivariate linear regression, etc) in the chemical engineering field, but this review will focus on the application of regression techniques to identify kinetic parameters (Burnham et al., 2008, Opfermann, 2000). The nature of reaction kinetics means that the problem has a large search space. This means that the domain of potential solutions is large with many independent variables that could predict the dependent variable. This large search space can often result in false positive solutions being identified. This is particularly problematic for regression methods as a large search space often results in finding local optima, rather than a global optimum. Cai et al. (2009) have developed a method of improving linear regression with large numbers of dependent variables by separating the dependant variables into groups, and through random regrouping, the identified models can converge onto a true model. Whilst this is an interesting method, it is less applicable to the goals of identifying reaction networks which tend to have sparse solutions. A sparse solution is a solution that has few non-zero terms. There have been a variety of methods developed to generate sparse solutions, i.e. minimise the cardinality, including the L-norms and Least Absolute Shrinkage and Selection Operator (LASSO). Yang et al. (2020) have developed an efficient method of applying the L1-regularlisation to large scale dynamic systems, and use a chemical reaction network as an example.

2.5.2 Genetic Algorithms

Genetic Algorithms (GAs) apply the principles of biological evolution and survival of the fittest to find the optimal solution to a problem. This is generally done by randomly creating a set of possible solutions (a generation), evaluating their success and/or failure, taking a subset of the best solutions, and mutating these solutions by making minor modifications to them and generating more random solutions which are like the previous successful solutions (create a new generation, based on the success of the previous generation). This cycle is repeated until a solution either hits an acceptable threshold, or a set number of generations have passed. Genetic algorithms are highly effective at finding optimal solutions, but they tend to be computationally expensive and slow to identify accurate solutions.

There are a variety of sub-categories that use GAs and therefore will be considered under the umbrella term of GA, including: differential evolution, Symbolic regression, and Simulated Annealing.

GAs and their variants have been used for a wide variety of classic optimisation problems such as identifying the optimal playstyle for the prisoners dilemma (Mitchell, 1995), identifying the shortest route for the travelling salesman problem (Potvin, 1996) and identifying optimal solutions to the knapsack problem – an optimisation problem for maximising the value of a bag while being constrained by the weight (Hristakeva and Shrestha, 2004). Each of these problems show that genetic algorithms and its variants can be applied to a variety of problems in a broad selection of fields. In computer science, genetic algorithms are considered suitable for solving NP-hard problems- problems where finding the solution is difficult, but testing the solution is easy.

A genetic algorithm typically has a ‘generation’ or the population of possible solutions, and each individual within that generation has a series of properties – a chromosome made up of individual genes (binary variable(s)), associated with it. These chromosomes and individual genes can be set up by the user, or automatically generated if the algorithm is given the ability to do so. Koza developed the idea and translated the binary structure into a tree diagram (Koza, 1992); this is also known as symbolic regression where the individual genes could refer to independent variables, operators (addition, subtraction, multiplication, etc), or constants. This has been used in a variety of fields including the identification of physical fluid flow models (Neumann et al., 2019).

Genetic algorithms have been used to identify stoichiometric and kinetic structures under multiple frameworks (Hii et al., 2014, McKay et al., 1997). Whilst these methods are capable of being very powerful, the ability to identify complex systems can take a long time to converge on optimal solutions. GAs also tend to overfit to data, and several methods have been investigated to overcome overfitting and reduce the chances of identifying local minima (Langdon, 2011, Gonçalves and Silva, 2013).

2.5.3 Mixed Integer Linear Programming

Mixed Integer Linear Programming (MILP) and its variants (Mixed Integer Quadratic programming, Integer Linear Programming, etc) has been used for a variety of problems including the travelling salesman problem, scheduling problems and other classic optimisation problems. Within the chemical industry the primary use of MILP has been used for process scheduling and process synthesis (Raman and Grossmann, 1991, Grossmann and Santibanez, 1980, Maia and Galvão, 2009, Floudas and Lin, 2005). Outside of the scheduling problems, MILP has been used by Patsiatzis et al. (2004) to identify optimal plant layouts with the primary concern being safety, when implemented this isn't dissimilar from a scheduling problem, however with different dimensions of interest.

There has been a growing number of articles and proceedings that use MILP to optimise the operational efficiency of a plant of some kind (Haikarainen, 2013). Adamson et al. (2017) followed suit with this approach and applied MILP to a steady state cryogenic air separation unit and compressor plant to optimise the operation of the entire system and improving the efficiency by reducing the power consumption by up to 5%. Laing et al. (2020) also applied MILP to a large system of a gas distribution and wastewater treatment works plant, to develop a more robust and efficient method of operating the plant. This has shown that MILP has a place in operations, in doing so these authors typically would follow principles laid out in scheduling problems in a different format so that MILP can be applied to their respective operational problems.

Logic and systemic modelling are closely related to the production routing of chemical processes, Raman and Grossmann (1991) combined the ideas of propositional logic, with MILP and chemical processes. This method used qualitative data rather than quantitative datasets to provide representations of knowledge that can be used to generate control strategies. Willis and Stosch (2016), used MILP to identify a subset of all possible kinetic structures, which was reduced by applying the law of conservation of mass to the system. By identifying multiple solutions to the same search space it allows entire reaction networks to be identified. The methodology creates multiple models that are compared using the Akaike or Bayesian information criterion. The work in this thesis is built upon the work on Willis and Stosch (2016).

2.2 Conclusion

This chapter has discussed the fundamentals of chemical reactions and the difficulty that is often found in the identification of those models. The difficulties in the identification of CRNs are typically due to the confounding of measurements caused by the highly coupled nature of kinetics of CRNs. Each of the techniques applied to these processes have their limitations, for example: TFA, whilst the most well-established method for identifying CRNs, has its performance heavily influenced by measurement noise, which is near-inevitable for a real process. Genetic Algorithms can be an effective method of identification of any of the chemical reactions, however time it takes for the genetic algorithm to find the solution can be unfeasible. MILP appears to be able to compensate for some of the flaws in each of the methods discussed and the research of MILP in the identification of chemical reaction networks is relatively sparse.

3 Chapter 3 Stoichiometric method results and analysis

3.1 Methodology

This methodology has the goal of being able to identify all the reaction stoichiometries. This is done in two stages:

1. Use the concentration profiles to identify reaction invariants,
2. Use the identified invariants to identify stoichiometries.

There are a variety of additional, optional variables that can be included or manipulated in the method to improve its efficacy. A flowchart with a low-resolution overview of the method is shown in Figure 3.1.

Starting with the concentration measurements of a reaction or set of reactions, the concentration profiles of each component are converted into the change in moles for each component at each time point. The algorithm automatically then generates a set of constraints to make the MILP method effective. These constraints are grouped into 5 categories:

3.1.1 Data constraints

The data constraints show where the feasible invariants lie within the search space. There is a slack variable associated with the data constraints to allow for measurement noise to be considered.

$$\boldsymbol{\Delta moles.I} - \varepsilon \leq tol \qquad \text{Equation 3-1a}$$

$$-\boldsymbol{\Delta moles.I} - \varepsilon \leq tol \qquad \text{Equation 3-1b}$$

Where ε is a slack variable that allows for a small amount of error to occur and tol is a tolerance of acceptable error in the identified invariant. This is a small value close to zero.

The optimisation is set up in this form so that by minimising the magnitude of ε Therefore, for the equations to hold true for all time points the coefficient associated with $\boldsymbol{\Delta moles}$, must be approximately equal to zero (within a tolerance threshold) – An Invariant.

An optional addition to this data set is a to integrate the $\boldsymbol{\Delta moles}$ data and use that data as an additional set of constraints. The integrated change in moles acts as a transformation of the data that minimises the effect of the noise on the system. This is particularly useful for systems with a higher percentage of noise and could replace the change in moles term. The inequalities created from the rate of change in moles becomes the core of the MILP algorithm. From this point forward, the inputs will be concatenated to the bottom of the matrix labelled $\boldsymbol{A}$.

3.1.2 Slack variable

The slack variable has been briefly discussed within the data constraints section. Slack variables are non-negative and are used to make the two inequality constraints a single equality constraint. The slack variable also acts like an error term that is minimised to allow the coefficients associated with the data to identify an invariant within the tolerance. The tolerance within this method is associated with the 'Big-M' method, the M is a large value which is an order of magnitude larger than the expected error – In this case it is set to a value of 10.

$$I - \varepsilon \leq (M + tol) \qquad \text{Equation 3-2a}$$

$$\varepsilon \leq tol \qquad \text{Equation 3-2b}$$

Equation 3-2a is an inequality designed to allow for negative terms to be identified as acceptable invariant values, and Equation 3-2b states that the acceptable error must be of magnitude less than the accepted tolerance.

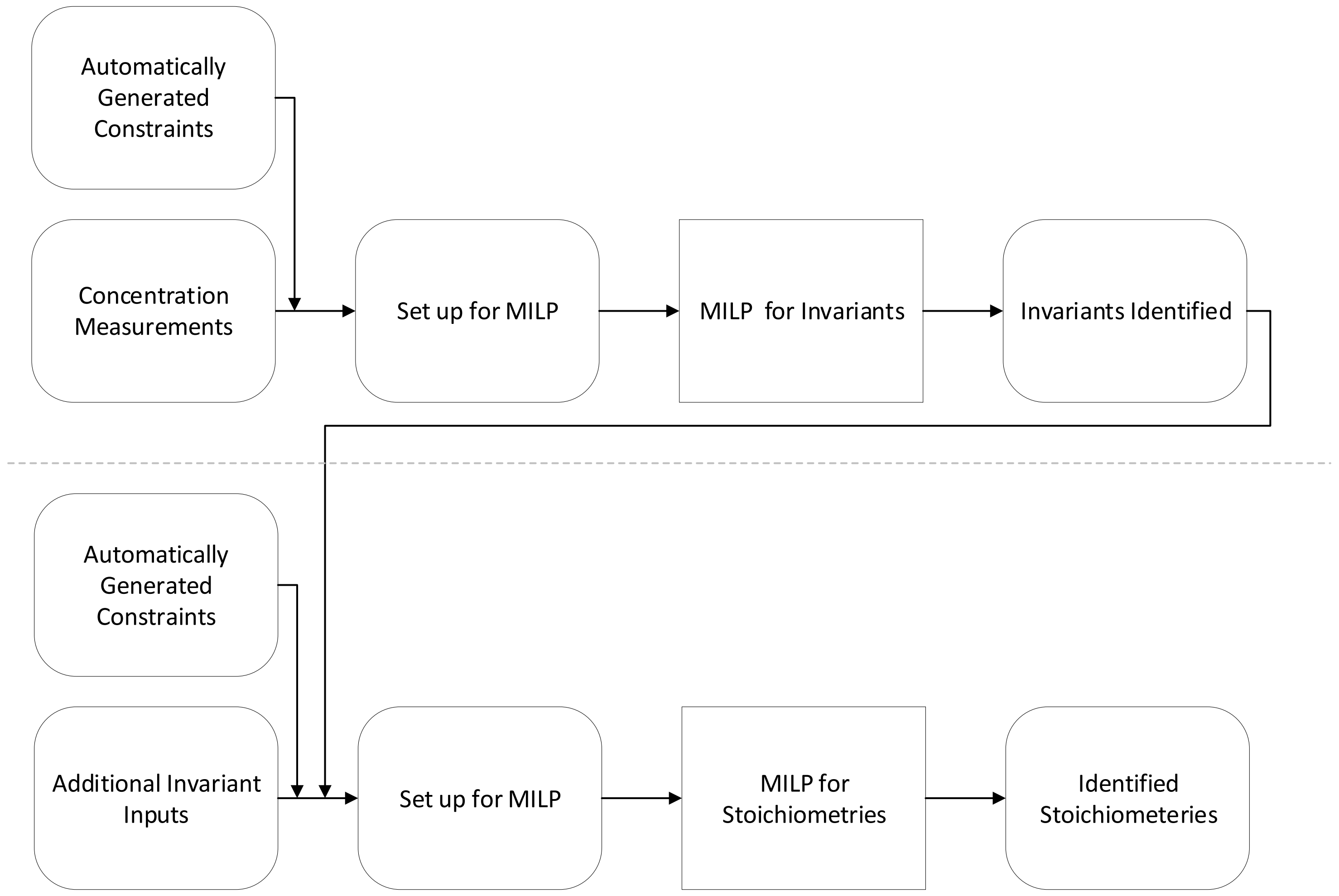

Automatically Generated Constraints
Concentration Measurements
Set up for MILP
MILP for Invariants
Invariants Identified
Automatically Generated Constraints
Additional Invariant Inputs
Set up for MILP
MILP for Stoichiometries
Identified Stoichiometeries

Figure 3.1 - Flow sheet of the identification of stoichiometries from concentration measurements via reaction invariants using MILP. Where the rounded cornered blocks are either data input or minor manipulations of the data, and the sharp cornered blocks are operations performed on the data. The dashed line separates the two parts of the algorithm with the upper-half being the algorithm for the identification of reaction invariants, and the lower half being the utilisation of the invariants to identify stoichiometries.

3.1.3 Binary constraints

There are 3 sets of binary variables ($\boldsymbol{\sigma}$) for each component to be added to the $\boldsymbol{A}$ matrix. The first $\boldsymbol{\sigma_1}$ is activated when the $\boldsymbol{A}$ matrix is non-positive, these are identified by two inequality constraints. The second binary variable ($\boldsymbol{\sigma_2}$) is activated when $\boldsymbol{A}$ is non-negative, and the third set of binary variables ($\boldsymbol{\sigma_0}$) that is activated when the associated $\boldsymbol{A}$ is zero. The first and second set of binary variables are described through two inequality constraints (Equation 3-3a-d). The third set of binary variables is described through Equation 3-3e. From these two sets of binary variables the number of zero terms within the vector $\boldsymbol{A}$ can be identified within the linear programming algorithm as another variable- $\boldsymbol{\sigma_0}$**.** The final variable ($\sum\sigma$) that counts the number of active $\boldsymbol{\sigma_0}$ terms. With the sum of the number of zero terms within a solution, this number can be minimised or maximised as needed by the algorithm, for this particular setup the goal is to make the solutions as sparse as possible therefore by maximising the number of zero terms in the solution and simultaneously minimising the data-model error, this increases the likelihood of finding an accurate invariant.

$$x.I + UB\sigma_1 \leq UB \qquad \text{Equation 3-3a}$$

$$-x.I + (LB + tol)\sigma_1 \leq tol \qquad \text{Equation 3-3b}$$

$$-x.I - LB\sigma_2 \leq -LB \qquad \text{Equation 3-3c}$$

$$x.I + (LB + tol)\sigma_2 \leq -tol \qquad \text{Equation 3-3d}$$

$$-\sigma_1 - \sigma_2 + \sigma_0 = -1 \qquad \text{Equation 3-3e}$$

$$\sum_{c=1}^{N_c} -\sigma_0 + \sum\sigma = 0 \qquad \text{Equation 3-3f}$$

3.1.4 Integer constraints

The identified Invariants must be an integer between within an upper and lower boundary. This boundary is a variable that can be varied at the user's discretion, the maximum magnitude of each value of the invariant is set to 5, the invariants are anticipated to be values of magnitude between 0 and 3. The binary variables are also constrained to be integers (either 0, or 1).

3.1.5 Cost function

The cost function is what is being minimised under the constraints set up previously. The goal is to find a point where the error term (slack variable) is zero, and the solution is as sparse as possible (maximum number of zero terms whilst still having minimum error).

$$\mathrm{J} = \min\,(j_1\varepsilon + j_2\textstyle\sum\sigma) \qquad \text{Equation 3-4}$$

In Equation 3-4 there is a cost to the errors in the model, and a cost to the number of zero terms used. The coefficient associated with $\sum\sigma$ can be manipulated by the user at their discretion, but for the purposes of this method, having both the errors and the number of zero terms have the same weight within the cost function. The error should always be minimised, however depending on the purposes, the desired number of zero terms could change – This can be done by changing the coefficient before the $\sum\sigma$ term.

3.1.6 Integer Cut

To find multiple invariants the algorithm must discover minima within the search space multiple times and find different solutions. To prevent the algorithm for repeatedly identifying the same minimum point the integer cut is used. The integer cut is an additional constraint that is added to the algorithm after a solution has been found so that when searching for an alternative minimum point the solution that has already been identified is considered a sub-optimal option.

The new constraint can be described by a set of conditional statements stated in Table 3-1. From this another line of the $\boldsymbol{A}$ matrix. The associated $\boldsymbol{B}$ with this new constraint is calculated using Equation 3-5.

Table 3-1 - Conditions to generate the new constraint using the integer cut method. If one of the binary variables is 1, then the corresponding constraint value is 1, etc.

Identified Invariant Value	**Constraint Value**
$\boldsymbol{\sigma = 1}$	$\sigma = 1$
$\boldsymbol{\sigma = 0}$	$\sigma = -1$
Non-binary variable	0

$$B_{new} = (Number\ of\ binary\ variables) \\ - (Number\ of\ inactive\ binary\ variables) - 1 \qquad \text{Equation 3-5}$$

With this constraint the optimisation can now be run on repeat until a termination condition is met. These conditions are:

- A limit to the maximum number of invariants identified, or
- The optimiser fails to find an invariant.

Once the invariants have been identified, they can be used to identify stoichiometries. The integer cut is also applied in the same manner in the stoichiometric algorithm (Section 3.1.9).

3.1.7 Identifying stoichiometries

The identification of stoichiometries is done using reaction invariants. As shown in Figure 3.1, there are optional, additional invariants that can be included. Additional information can include the atomic matrix, and/or any a priori information already known about the system. For example, knowledge that one of the components is a reactant or product, can be implemented as an additional invariant. Once these invariants are collected, they need to be converted into a form that is conducive to identifying stoichiometries – much like was done with the measurements for identifying invariants. The reaction invariants can be identified if and only if Equation 3-6a is true (Waller and Makila, 1981).

To identify stoichiometries,

$$\boldsymbol{I}.\boldsymbol{S} = \boldsymbol{0} \qquad \text{Equation 3-6a}$$

As with the identification of invariants algorithm, the identification of stoichiometries needs to be done in inequality form, as shown in Equation 3-6b & c.

$$\boldsymbol{I}.\boldsymbol{S} - \boldsymbol{\varepsilon} \leq \boldsymbol{0} \qquad \text{Equation 3-6b}$$

$$-\boldsymbol{I}.\boldsymbol{S} - \boldsymbol{\varepsilon} \leq \boldsymbol{0} \qquad \text{Equation 3-6c}$$

This has some similar constraints to the invariants identification function including slack variables, binary variables, and other additional, optional constraints include the a priori information that can be included. There is more a priori knowledge that can be implemented in this method depending on what is known about the reaction network. For example, if a component is known to be a reactant, and cannot be an intermediate, nor a product, then a constraint stating that the stoichiometric coefficient of the component must either have a negative or zero term. This is not a reaction invariant, however within the context of this algorithm, this constraint can be treated in the same way as an invariant.

In Equation 3-6b & c there is an error term (ε) this – as with the invariants algorithm, acts as both a slack variable, and an error term. As with the Invariants Identification Algorithm there are binary variables that are used to manipulate the number of zero terms when identifying stoichiometries.

$$I.S + UB\sigma_1 \leq UB \qquad \text{Equation 3-7a}$$

$$-I.S + (LB + tol)\sigma_1 \leq tol \qquad \text{Equation 3-7b}$$

$$-I.S - LB\sigma_2 \leq -LB \qquad \text{Equation 3-7c}$$

$$I.S + (LB + tol)\sigma_2 \leq -tol \qquad \text{Equation 3-7d}$$

$$-\sigma_1 - \sigma_2 + \sigma_0 = -1 \qquad \text{Equation 3-7e}$$

$$\sum_{c=1}^{N_c} -\sigma_0 + \sum\sigma = 0 \qquad \text{Equation 3-7f}$$

Equation 3-7 operates in the same way as previously described in Section 3.1.3: Equation 3-7a & b identify non-positive values within the identified stoichiometries and activates σ_1. Equation 3-7c & d identify non-negative identified stoichiometric values to activate σ_2. Equation 3-7e & f are both used to identify and count the number of stoichiometric coefficients that have a value of zero, and to count the zero terms.

There are integer constraints set to all variables, except for the error term. The upper and lower bounds for each variable are shown in Table 3-2. The stoichiometric coefficients ($\boldsymbol{S}$) can be expanded to a larger range if necessary, however the increased search space will affect the speed of the algorithm and stoichiometric coefficients are not typically larger than this value. The slack/error variable (ε) has an infinite upper boundary which appears to be excessive, however since this variable is being minimised, the upper limit is of less significance. The number of zero terms ($\sum\sigma$) is constrained to be fewer than the number of components in the system. This means that there must always be a non-zero term within every identified stoichiometry.

Table 3-2 - Upper and Lower bounds of each of the elements of the MILP formulation where s is the bounds on the stoichiometric values, and σ are binary variables.

S_{lower}	S_{upper}	ε_{lower}	ε_{upper}	$\sigma_{all,lower}$	$\sigma_{all,upper}$	$\sum\sigma_{lower}$	$\sum\sigma_{upper}$
-3	3	0	Inf	0	1	0	$c-1$

3.1.8 Cost function

The cost function of the stoichiometric identification function is identical to the cost function of the invariant identification algorithm, as follows:

$$\mathrm{J} = \min\left(j_1\varepsilon + j_2\textstyle\sum\sigma\right) \qquad \text{Equation 3-8}$$

As before, there are two variables being optimised, the error within the algorithm, and the total number of zero terms. The coefficients j_1& $j_2$can be adjusted depending on the goals and complexity of the system. For all the experiments performed, both coefficients are set to a value of one, giving each of them an equal effect on the identified solutions.

3.1.9 Integer cut

An Integer Cut is implemented on the stoichiometric algorithm in the same method as way applied to the Invariants Identification Algorithm. The integer cut generates a constraint – based upon the identified stoichiometry, that can be added to the inequality constraints matrix. This additional constraint prevents the algorithm from identifying the same stoichiometry and therefore finding a variety of stoichiometric vectors that will (partially) describe the reaction network. A mathematical description of the integer cut is shown in Table 3-1 and Equation 3-5.

3.2 Results and discussions

3.2.1 The use of atomic matrix

There are two parts to the stoichiometric identification algorithm, the identification of reaction invariants, and the identification of reaction stoichiometries. The inputs to the stoichiometric identification algorithm are reaction invariants and any a priori information about the reaction network that can be added. If the atomic matrix of a reaction network is known, then the atomic matrix can act as a set of reaction invariants. The atomic matrix of the biodiesel reaction network is known and can be used to identify stoichiometries shown in Table 3-3.

Table 3-3 - Identified stoichiometries using only the atomic matrix.

	TG	MeOH	DG	BD	MG	GL
$\boldsymbol{\nu_1}$	-2	-2	2	2	0	0
$\boldsymbol{\nu_2}$	-2	0	2	0	2	-2
$\boldsymbol{\nu_3}$	-1	-2	0	2	1	0
$\boldsymbol{\nu_4}$	-1	0	2	0	-1	0
$\boldsymbol{\nu_5}$	0	-1	-1	1	1	0
$\boldsymbol{\nu_6}$	0	-1	0	1	-1	1
$\boldsymbol{\nu_7}$	0	0	-1	0	2	-1
$\boldsymbol{\nu_8}$	0	0	1	0	-2	1
$\boldsymbol{\nu_9}$	0	2	0	-2	2	-2
$\boldsymbol{\nu_{10}}$	0	2	1	-2	0	-1
$\boldsymbol{\nu_{11}}$	0	2	2	-2	-2	0
$\boldsymbol{\nu_{12}}$	1	-1	-2	1	0	1
$\boldsymbol{\nu_{13}}$	1	0	-2	0	1	0
$\boldsymbol{\nu_{14}}$	1	1	-1	-1	0	0
$\boldsymbol{\nu_{15}}$	1	1	0	-1	-2	1
$\boldsymbol{\nu_{16}}$	2	0	-2	0	-2	2

This method identifies all the feasible stoichiometries based upon the atomic matrix, including the desired stoichiometries of the reaction network (identified stoichiometries ν_1, ν_5 and ν_6). It can also be seen that the reverse reactions are also identified, this can be remedied by including some a priori information. Looking at Table 3-2 there are two components that are reactants that start at a high concentration and tend towards zero (TG and MeOH). There are also two intermediates (DG and MG) that can be identified by their concentrations start at zero, rise to a peak, and then tend towards zero again. Finally, two products (GL and BD) are identified by starting with a concentration of zero and tending towards a final concentration without falling. It is worth noting that this matrix could be simplified with Gaussian elimination to identify the row echelon form or Gauss-Jordan elimination for reduced row echelon form. When Table 3-3 is simplified, the reaction network is simplified to Table 3-4.

Table 3-4 – The row Echelon form identified via Gaussian Elimination to show the simplified version of the results shown in Table 3-3.

	TG	MeOH	DG	BD	MG	GL
$\boldsymbol{\nu_1}$	-2	-2	2	2	0	0
$\boldsymbol{\nu_2}$	0	2	0	-2	2	-2
$\boldsymbol{\nu_3}$	0	0	2	0	-4	2

Table 3-5 – The Reduced Row echelon form identified via Gauss-Jordan Elimination to show the simplified version of the results shown in Table 3-3.

	TG	MeOH	DG	BD	MG	GL
$\boldsymbol{\nu_1}$	1	0	0	0	-3	2
$\boldsymbol{\nu_2}$	0	1	0	-1	1	-1
$\boldsymbol{\nu_3}$	0	0	1	0	-2	1

When the results are simplified with Gaussian Elimination, or Gauss-Jordan Elimination, there are some results that are simplified. Table 3-4 shows the Row Echelon form which reduces the number of identified reactions to three, which is accurate, however ν_3 is not the simplest form of reaction 3 shown in Table 3-3The identified ν_3 is a combination of reactions 2 and 3. This indicates that the Row Echelon form of the identified stoichiometries provides some information, however it cannot confirm the exact reactions taking place. Similarly with the Reduced Row Echelon form – shown in Table 3-5, has a similar problem. The nature of Row Echelon, and Reduced Row Echelon form results in there must be zero terms beneath the diagonal of the matrix. This means that in the case of this example, the row echelon and reduced row echelon form of these results cannot include component 2 (MeOH) in all three of the reactions. When the order of the chemicals within the identified stoichiometric matrix are rearranged, Gaussian Elimination can simplify the identified reaction networks to the true stoichiometric matrix shown in Table 3-3**Error! Reference source not found.**, the G aussian Elimination of the re-arranged identified stoichiometries are shown in Table 3-6.

Table 3-6 – Gaussian Elimination performed on the rearranged identified stoichiometries. Note that in most of the tables, MeOH is in the second column, but in this instance MeOH is in the 6th column.

	TG	**DG**	**BD**	**MG**	**GL**	**MeOH**
$\boldsymbol{\nu_1}$	-1	1	1	0	0	-1
$\boldsymbol{\nu_2}$	0	1	-1	-1	1	1
$\boldsymbol{\nu_3}$	0	0	-1	1	-1	1

The Reduced Row Echelon form of a reaction network defines that all values off the diagonal should be zero, this results in a simplification of the CRN that does not show stoichiometries as would normally be shown, but rather as combinations of the true stoichiometries. Therefore, from this point onwards the results are shown in full without simplification. It is acknowledged that the Row Echelon form of the results is likely to be the most simplified form of the CRN, but identification would either require a manual rearrangement of the columns or a priori knowledge about the reaction network.

These insights to the reaction network can be included in the identification process as additional constraints. With these constraints the number of identified stoichiometries is reduced, removing all the inverse reactions and many of the other feasible reactions.

Table 3-7 - Identified stoichiometries when using the atomic matrix, and a priori knowledge about the system.

	TG	**MeOH**	**DG**	**BD**	**MG**	**GL**
$\boldsymbol{\nu_1}$	-1	-2	0	2	1	0
$\boldsymbol{\nu_2}$	-1	-1	1	1	0	0
$\boldsymbol{\nu_3}$	-1	0	2	0	-1	0
$\boldsymbol{\nu_4}$	0	-2	-1	2	0	1
$\boldsymbol{\nu_5}$	0	-1	-1	1	1	0
$\boldsymbol{\nu_6}$	0	-1	0	1	-1	1
$\boldsymbol{\nu_7}$	0	0	1	0	-2	1

Table 3-7 shows that there are four identified stoichiometries that are feasible according to this identification method, these are discussed further in the next section.

3.2.2 Uncorrupted data

The results generated are highly dependent on the data provided, therefore this set of results discuss a relatively simple problem for the stoichiometric identification algorithm, a system with 75 measurements and 0% artificial noise added to the data. The identified invariants are shown in Table 3-8, and the stoichiometries are identified in two ways, firstly without the atomic matrix, and then with the atomic matrix, summarised in Table 3-10 and

Table 3-11. To interpret Table 3-8 it is useful to look for an understanding of what each of the identified invariants could be with respect to the chemicals and what is reacting. For example, I_9 shows that the sum of all the number of moles of material present throughout the reaction network is constant. This shows that the law of conservation of mass is being followed in this reaction – Since this reaction takes place in a batch reactor, this is expected. Although this will probably not be particularly useful for the identification of stoichiometries, it is an important observation about the data used. I_9 also implies that all the components are part of this reaction network and none are non-reactive substances. Another invariant with easily interpretable information found is I_{11}. By investigating I_{11} alongside the atomic matrix the number of carbon atoms is constant throughout the reaction network, and the total number of carbon atoms can be in any of the four locations identified in $\boldsymbol{I_{11}}$. This invariant is easily explainable with the atomic matrix available, this also shows the fact that the atomic matrix is an efficient method of displaying invariants.

Table 3-8 - Identified Invariants with 75 measurements and 0% artificial noise.

	TG	MeOH	DG	BD	MG	GL
***I*_1**	-2	-1	-2	-1	-2	-2
***I*_2**	-2	-1	-1	-2	0	1
***I*_3**	-2	2	-1	1	0	1
***I*_4**	-1	1	-1	1	-1	-1
***I*_5**	-1	2	0	1	1	2
***I*_6**	1	-2	0	-1	-1	-2
***I*_7**	1	-1	1	-1	1	1
***I*_8**	1	1	0	2	-1	-2
***I*_9**	1	1	1	1	1	1
***I*_10**	2	-2	1	-1	0	-1

***I*_11**	1	0	1	0	1	1
***I*_12**	2	1	1	2	0	-1

Another Invariant of interest that is easily interpretable when also considering the atomic matrix is I_1. I_1 is easier to interpret once I_{11} has already been identified as I_{11} is a factor/contained within I_1. This means that there are invariants combinations of multiple invariants. Whilst in many cases the combination invariants could be ignored, the extraction of the basic invariants from the combination invariants holds no benefit to the stoichiometry identification algorithm. The information contained in one combination invariant is the same as the information within two basic invariants, repeating this information does not over-constrain the algorithms. It is also observing that there are a pair of invariants that are proportionally identical: I_2 and I_{12} provide the same information, as with the combination invariants this does not over-constrain the stoichiometric identification algorithm however, some of the Invariants have less obvious interpretations, I_6 for example is difficult to interpret what the physical cause of the invariant is, it can however be graphically shown to be an invariant as shown in Figure 3.2.

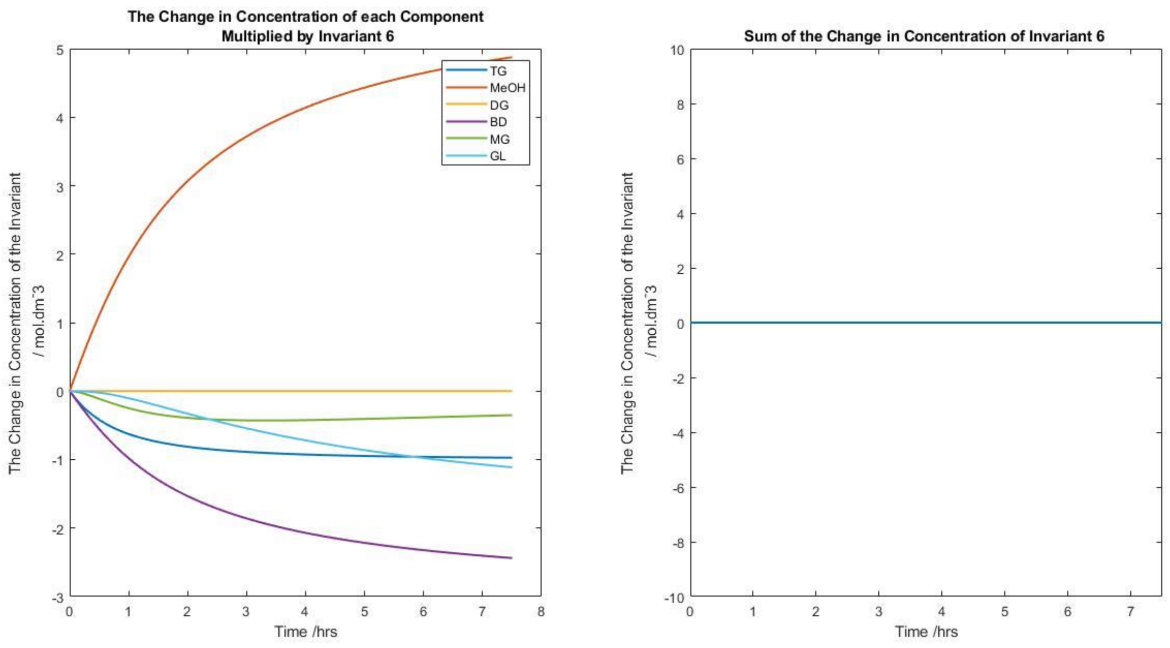


Figure 3.2 – Graphs showing the change in concentration of each of the individual components multiplied by the respective Invariant coefficient, and the sum of the change in concentration of the components multiplied by the respective Invariant.

It is likely that Invariant 6 is a combination of several basic invariants, however it can be difficult to separate the basic invariants from the combination invariants. By removing the known basic invariants from the combination invariants, it could uncover more basic invariants, however as stated previously, this offers no advantage over using the combination invariants as they are automatically identified. Table 3-9 also shows the effectiveness of the stoichiometric identification algorithm in and of itself. In theory the information within the atomic matrix is enough to generate accurate stoichiometries.

Table 3-9 - Stoichiometric coefficients identified using only the atomic matrix.

	TG	**MeOH**	**DG**	**BD**	**MG**	**GL**
$\boldsymbol{\nu_1}$	-1	-2	0	2	1	0
$\boldsymbol{\nu_2}$	-1	-1	1	1	0	0
$\boldsymbol{\nu_3}$	-1	0	2	0	-1	0
$\boldsymbol{\nu_4}$	0	-2	-1	2	0	1
$\boldsymbol{\nu_5}$	0	-1	-1	1	1	0
$\boldsymbol{\nu_6}$	0	-1	0	1	-1	1
$\boldsymbol{\nu_7}$	0	0	1	0	-2	1

Comparing the stoichiometries of the reaction to the stoichiometries identified using this algorithm (Table 3-9) it is immediately obvious that there are more stoichiometries identified than exist within the reaction network. This is because the algorithm identifies combinations of correct stoichiometries alongside the correct stoichiometries. It can be seen that ν_2, ν_5, and ν_6 are the correct stoichiometries, and that the other results are combinations of these. For example, $\nu_1 = \nu_2 + \nu_5$. As expected, the combinations would be accurate within the same set of constraints as the true stoichiometries.

Table 3-11 also uses the atomic matrix as and additional set of identified reaction invariants to identify the stoichiometries.

Table 3-10 - Stoichiometric identified using only the invariants identified in Table 3-8

	TG	**MeOH**	**DG**	**BD**	**MG**	**GL**
$\boldsymbol{\nu_1}$	-2	-2	2	2	0	0
$\boldsymbol{\nu_2}$	-1	-2	0	2	1	0
$\boldsymbol{\nu_3}$	-1	0	2	0	-1	0

$\boldsymbol{\nu_4}$	0	-2	-2	2	2	0
$\boldsymbol{\nu_5}$	0	-2	-1	2	0	1
$\boldsymbol{\nu_6}$	0	-1	0	1	-1	1
$\boldsymbol{\nu_7}$	0	0	1	0	-2	1

Table 3-11 - Stoichiometric coefficients identified using the invariants identified in Table 3-8 and also the atomic matrix.

	TG	**MeOH**	**DG**	**BD**	**MG**	**GL**
$\boldsymbol{\nu_1}$	-2	-2	2	2	0	0
$\boldsymbol{\nu_2}$	-1	-2	0	2	1	0
$\boldsymbol{\nu_3}$	-1	0	2	0	-1	0
$\boldsymbol{\nu_4}$	0	-2	-2	2	2	0
$\boldsymbol{\nu_5}$	0	-2	-1	2	0	1
$\boldsymbol{\nu_6}$	0	-1	0	1	-1	1
$\boldsymbol{\nu_7}$	0	0	1	0	-2	1

The primary observation of Table 3-10 and

Table 3-11 is that their results are identical. The only observable difference between these results and those in Table 3-9 is a proportional difference for some of the identified stoichiometries. i.e. ν_1 (Table 3-10) $= 2 * \nu_2$(Table 3-9). This also means that the results of all three tables contain the correct stoichiometries, and the same combination stoichiometries. This implies that the same information is contained within the identified invariants as is within the atomic matrix. This would indicate that the identified invariants can be used when there is no atomic matrix available.

This simple example has shown the efficacy of this algorithm under simple conditions. The correct stoichiometries are found both with minimal information provided. The addition of this optional information removes some of the excess combination-stoichiometries and gives a smaller set of stoichiometries to be identified. The algorithm should now have a more difficult test, with less measurements, with artificially created noise corrupting the data.

3.2.3 Reduced measurements with noise

To test the algorithms further, a more difficult scenario is considered – by decreasing the number of measurements used (from seventy-five) to twenty-five and increasing the amount of gaussian noise up to two percent. The identified invariants are shown in Table 3-12, with the identified stoichiometries with and without the atomic matrix shown in Table 3-13 and Table 3-14 respectively.

Table 3-12 – Identified Invariants when 25 measurements are used, and 2% artificial noise is added.

	TG	MeOH	DG	BD	MG	GL
I_1	1	1	1	1	1	1
I_2	2	1	1	2	0	-1
I_3	1	1	0	2	-1	-2
I_4	1	-2	1	-2	1	1
I_5	1	0	1	0	1	1
I_6	2	-2	1	-1	0	-1
I_7	-2	2	-1	1	0	1
I_8	-1	1	-1	1	-1	-1
I_9	0	1	0	1	0	0
I_{10}	-1	1	0	0	1	2

By comparing the identified invariants when lower quality data is provided (Table 3-12) to when the data provided is higher quality (Table 3-8) it can be seen that seven of the invariants are the identical, there are therefore five invariants that were identified in Table 3-8, but do not appear in Table 3-12, and two are still present, but the sign has been inverted – as discussed in 3.2.2 a proportional change to the invariants gives the same effective constraint. If the noise could be removed then the unidentified invariants are more likely to be identified, Figure 3.3 shows an invariant that was not identified when noise was present but was shown to be an invariant when no noise was present in Table 3-8. This graph shows that there are some invariants that have not been identified when noise is present but are still exist as an invariant present within the system.

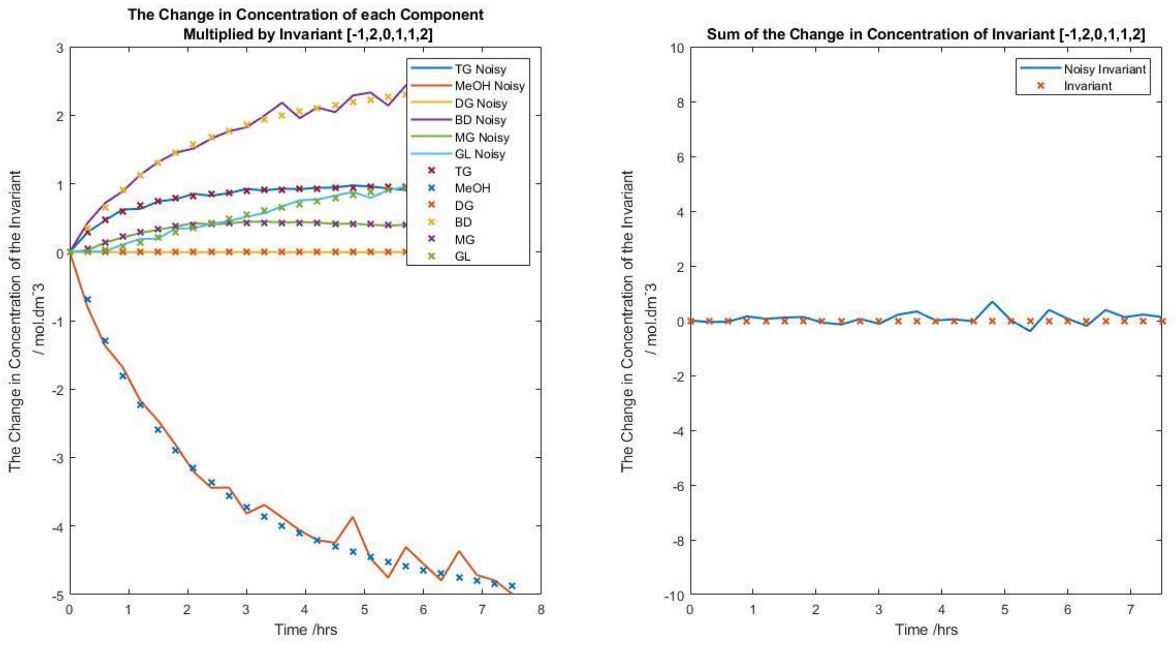


Figure 3.3 - Graphs of an invariant that was identified without any noise present but could not be identified in the more difficult scenario. The noise-free and the noisy invariant data are both shown.

Within the algorithm there is a tolerance that allows invariants to be accepted, this tolerance is made two considerations: it must filter out non-invariants, and it must allow true invariants to be identified when noise is present. In the case of the invariant displayed in Figure 3.3, the noise levels were particularly high at approximately five hours into the experiment and was not within the accepted tolerance levels. Fortunately, the information contained within this invariant can be accounted for within other invariants and the identified stoichiometries are correct and are the same both with and without the atomic matrix as shown in Table 3-13 and Table 3-14.

Table 3-13 -Identified stoichiometries when only the invariants found in Table 3-12 are used.

	TG	MeOH	DG	BD	MG	GL
ν_1	-2	-2	2	2	0	0
ν_2	-1	-2	0	2	1	0
ν_3	-1	0	2	0	-1	0
ν_4	0	-2	-1	2	0	1
ν_5	0	-2	0	2	-2	2
ν_6	0	-1	-1	1	1	0

$\boldsymbol{\nu_7}$	0	0	1	0	-2	1

Table 3-14 - Identified stoichiometries when the invariants found in Table 3-12 and the atomic matrix are used together.

	TG	**MeOH**	**DG**	**BD**	**MG**	**GL**
$\boldsymbol{\nu_1}$	-1	-2	0	2	1	0
$\boldsymbol{\nu_2}$	-1	-1	1	1	0	0
$\boldsymbol{\nu_3}$	-1	0	2	0	-1	0
$\boldsymbol{\nu_4}$	0	-2	-1	2	0	1
$\boldsymbol{\nu_5}$	0	-1	-1	1	1	0
$\boldsymbol{\nu_6}$	0	-1	0	1	-1	1
$\boldsymbol{\nu_7}$	0	0	1	0	-2	1

The identified stoichiometries are unaffected by the addition of the atomic matrix, this implies that all the identified invariants are correct as a false invariant would make it impossible to find the stoichiometries even with the atomic matrix.

This set of results has also shown that this method is still effective in identifying stoichiometries when noise is present, and fewer datapoints are used. However, to fully understand how noise and the number of measurements affects the efficacy of this algorithm, a more thorough study of these effects should be done.

3.2.4 Varying noise and measurements

The artificial noise present in this example is normally distributed pseudo-randomly generated within MATLAB and therefore the noise used is different each repeat, therefore resulting in different identified invariants. It is therefore necessary to perform a Monte Carlo simulation, repeating the algorithm multiple times to see the effect the noise has over a large sample size. To illustrate the point a variety of conditions were simulated for 50 iterations. The different conditions are expressed over an x-y plane where the x-axis shows the different noise levels, and the y-axis shows the different number of time intervals are used. In Figure 3.4 the z-axis shows the percentage of the time that the that the algorithm is successful in identifying the entire stoichiometric matrix. Figure 3.5 shows the same data without the z-axis, and instead identifies the percentage of the repeats that identify all the reactions through a colour gradient.

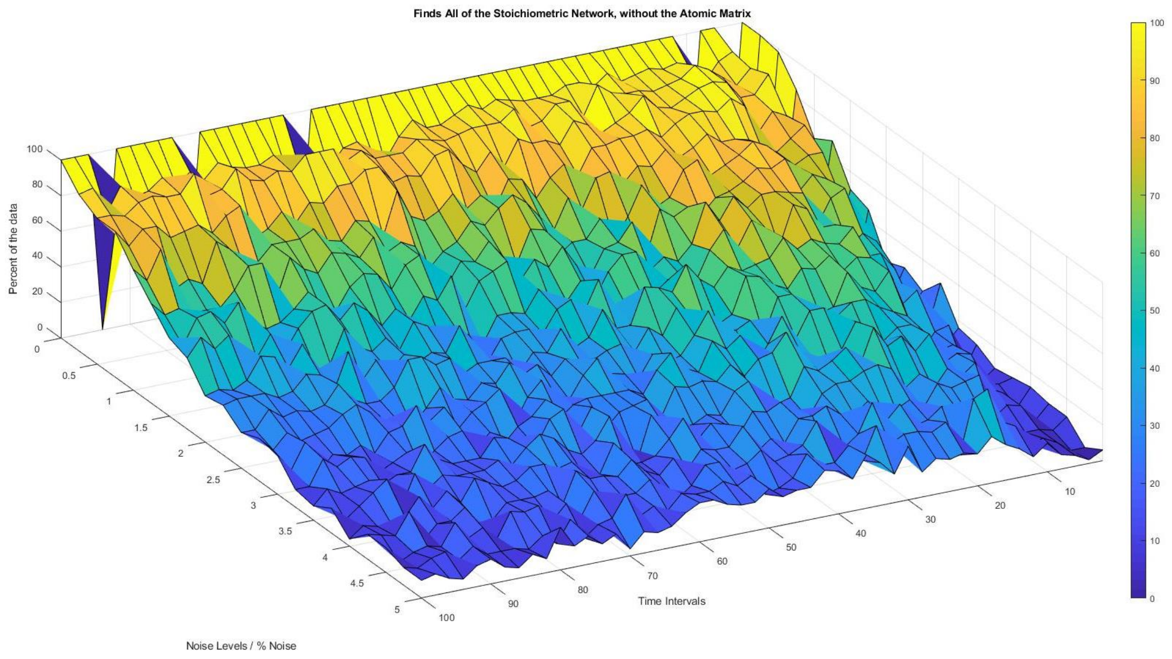


Figure 3.4 - 3D plot showing the percentage of the time that the Stoichiometric Identification Algorithm finds the entire stoichiometric network without using the atomic matrix under different conditions. Fifty repeats at each condition.

Figure 3.4 shows that the noise level appears to have a stronger influence than the number of time intervals. When the noise is zero percent the stoichiometries are all found regardless of the number of time intervals used until the number of measurements decreases below five at which point it is inferred that there is not enough data provided for there to be any information gleamed about the system. Similarly, when the noise levels are below 1.5%, the algorithm seems to have a near-100% success rate in identifying the entire reaction network's stoichiometry. However, it deteriorates as the noise increases to a near-0% chance of identifying the entire reaction network at the noise increases to 5%. However, in Figure 3.5 there appears to be a synergy between the number of data points and the amount of noise – when the number of datapoints is between 5 and 30, the noise can be higher, and still achieve a high probability of finding the entire reaction network.

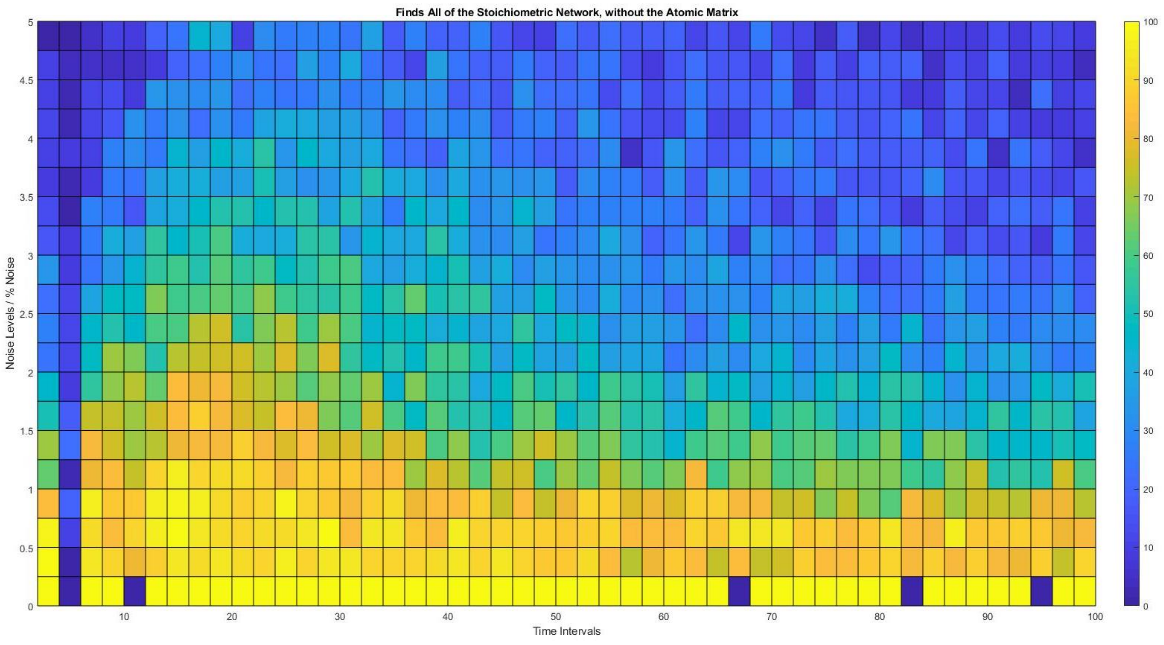


Figure 3.5 -Plot showing the percentage of the time that the Stoichiometric Identification Algorithm finds the entire stoichiometric network without using the atomic matrix under different conditions. Fifty repeats at each condition. This is the same graph as Figure 3.4. Instead of the z-axis the percentage is shown through colour instead of height. There also appears to be some anomalous readings when zero noise is present, upon repeating the experiment at these conditions, a more expected high percentage chance of identifying the correct stoichiometries. The author can only speculate as to why these anomalous readings occurred, when the results were initially collected. It is seen below (Figure 3.8) that the collected results identified had some of the correct stoichiometries, but not all three stoichiometries.

In a range of 5 to 30 time intervals the percentage of the time correct stoichiometries are found is significantly improved. This appears to be a region where the lack of measurements compensates for the noise levels. This can be explained by the effect of noise being less impactful when the measurements have a larger space between them.

The lack of successful identified stoichiometries at high noise levels and large numbers of measurements is likely because there are more points where the noise can exceed the tolerance of the algorithm for finding correct invariants. This is confirmed by the number of identified invariants being low, when the noise is high, and invariants that are identified are less likely to be identified as the corruption caused by the noise exceeds the acceptable tolerance.

This does provide an optimal operating region when considering a real system. The level of noise in real measurements cannot be controlled very tightly. However, the number of observations can be controlled, therefore when operating with a real system the number of measurements should ideally be between ten and thirty or the maximum number possible which can then be reduced to a more suitable number with even spacing.

It is worth comparing when the atomic matrix is not used (Figure 3.4) and when the atomic matrix is used (Figure 3.6). It has already been shown that if only the atomic matrix is used then the correct stoichiometries are found. Therefore, if stoichiometries are not identified in both figures it implies that false positive invariants have been selected and used in the stoichiometric identification algorithm. A solution to this would be to tighten the tolerance on the invariants identification algorithm, however this will reduce the chances of finding any invariants that can be identified at larger noise percentages. Figure 3.5 and Figure 3.7 can also be compared in a similar way.

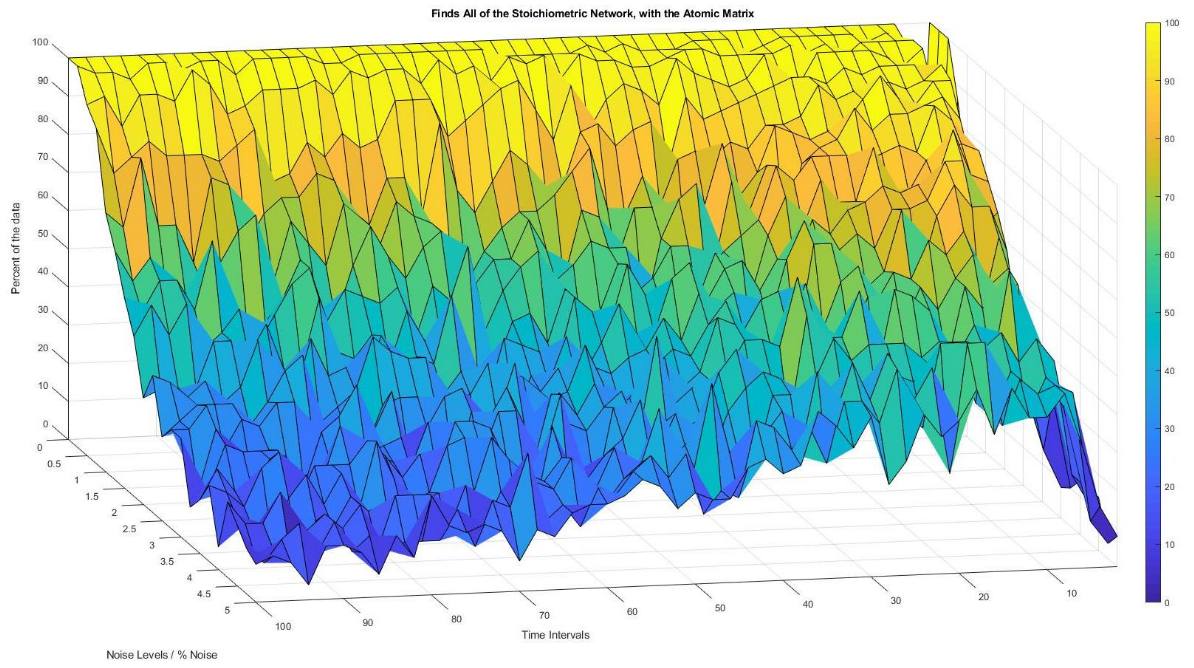


Figure 3.6 - 3D plot showing the percentage of the time that the Stoichiometric Identification Algorithm finds the entire stoichiometric network when using both the identified invariants and the atomic matrix under different conditions.

Figure 3.6 appears to be similar to Figure 3.4, with strong gradient with respect to the noise level, and a much weaker gradient with respect to the time intervals axis. However, the influence of the time intervals appears to be slightly stronger where the atomic matrix is used.

This is seen more clearly in Figure 3.7, where the 10-30 time intervals region appears to be larger, and extends deeper into the higher noise levels.

Using this information it can be inferred that the identified invariants are more likely to be accurate when the number of time intervals used is within the lower (10-30 range) and the noise levels are higher. However, these invariants are not providing enough information to identify correct stoichiometries and by including the atomic matrix the probability of finding correct stoichiometries increases in this region.

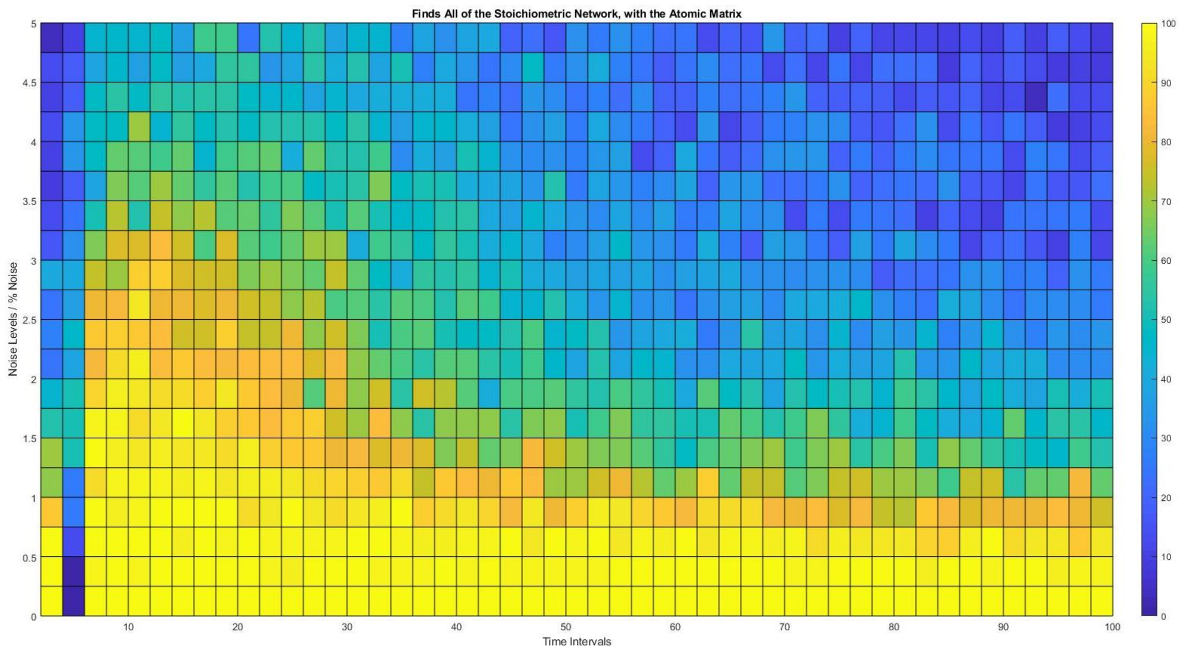


Figure 3.7 - Plot showing the percentage of the time that the Stoichiometric Identification Algorithm finds the entire stoichiometric network using the atomic matrix under different conditions. Fifty repeats at each condition. This is the same graph as Figure 3.6. Instead of the z-axis the colour bar shows the different percentages rather than the height.

It is also worth noting in Figure 3.7 that when there are five measurements, there is a very low percentage chance of identifying all of the stoichiometries. This shows that the invariants identified when five measurements are used are all incorrect, and this conflicts with the atomic matrix. However, when the number of measurements is reduced to 3, there is a higher chance of identifying the stoichiometries. This is explained when investigating the number of invariants identified. The algorithm fails to identify any invariants when there are three measurements.

Under all conditions the chance of finding partial stoichiometric models of the algorithm are relatively stable (Figure 3.8), however the atomic matrix appears to remove almost all possibility of finding a partial solution. The only way for there to be partially correct solutions when using the atomic matrix is for there to be a false invariant that makes it impossible to identify one of the reactions but does not influence the other identified reactions. This does not appear to account for the anomalous results. It is suspected that there may have been some human error during the conditions with anomalous results. As stated above, upon re-running these conditions at a later date, an expected result occurs.

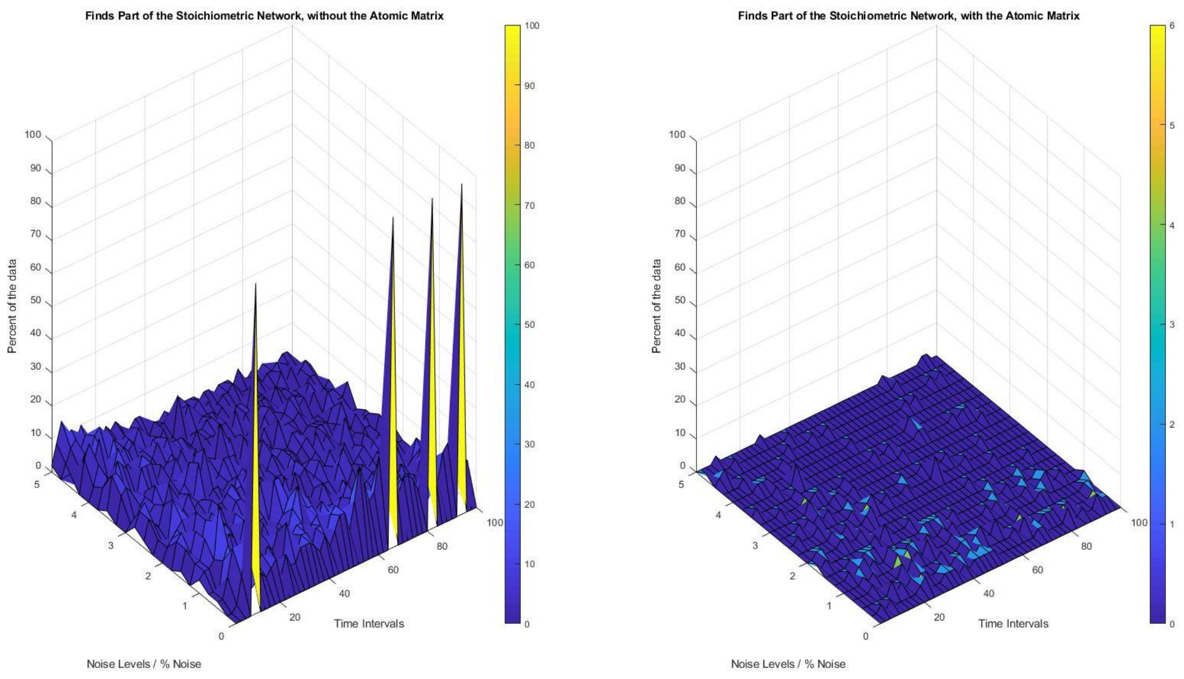


Figure 3.8 - Partially correct solutions at different conditions.

3.2.5 The effect of tolerance

By manipulating the tolerance on the invariant identification algorithm, the identified invariants change. By decreasing the tolerance the random noise will increase individual measured points to be outside of the acceptable tolerance (see Figure 3.3). By increasing the tolerance more invariants will be found as the effect of noise will be negated, however there is the possibility of allowing false positives being accepted. To show the effect of how the tolerance influences the identified invariants, the system considered is repeated thirty times at 2% artificial noise added, and 25 measurements used. Previous results show that this was relatively good condition with 80-90% success rate when tolerance set to a value of 1, shown in Figure 3.4 & Figure 3.6. Figure 3.9 shows the change in the number of identified invariants with respect to the tolerance.

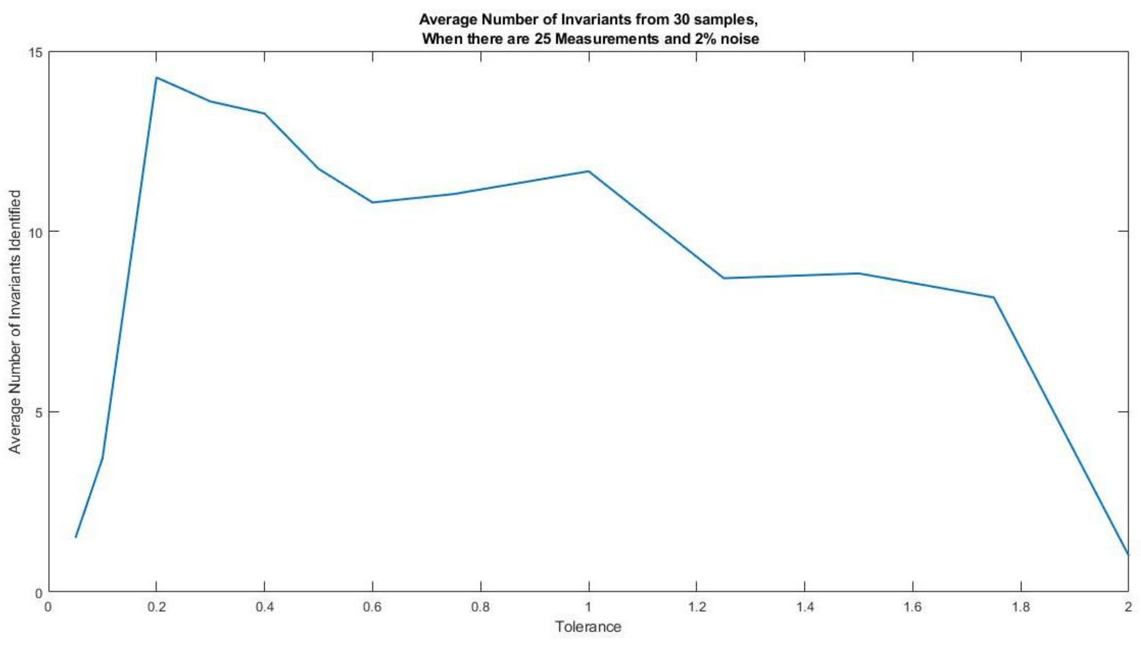


Figure 3.9 – The average number of identified invariants when varying the tolerance for acceptable invariants

The tolerance appears to have a considerable effect on the number of identified invariants. Once the tolerance gets below approximately 0.2 the average number of invariants identified reduces to 1.5. Additionally, the invariants identified tend not to contain useful information. For example, one of the repeats at a tolerance of 0.2, identifies only 1 invariant, a vector of zeros. For a tolerance range between 0.2 and 1.8, the average number of invariants appears to have a shallow decline in the number of invariants identified. This decrease in the average number of identified invariants is explained as the tolerance increases, the number of false positives that are identified increases. These false positives are then removed automatically by the algorithm on the grounds of them appears to have a trend that is not based around zero. Once the tolerance is increased above the value of approximately 1.8 the number of identified invariants tends towards zero. To explain this the identified invariants are considered. The only invariant that appears in all the repeats at this condition is a vector of ones. This is probably because of the two stages of invariant generation – Identification, and selection. To be selected, the invariants identified must have errors with a near-zero mean and the errors should have a line of best fit with a near-zero gradient and near-zero intercept:

$$-0.1 \leq \overline{Error} \leq \ 0.1 \qquad \text{Equation 3-9}$$

$$-0.01 \leq \dot{f} \leq 0.01$$

$$-0.05 < f(0) < 0.05$$

This secondary condition to the acceptance is preventing the identification of more acceptable invariants could prevent the identification of acceptable invariants, however this would unlikely if the noise is gaussian. It is also worth considering the number of identified stoichiometries, and their accuracy.

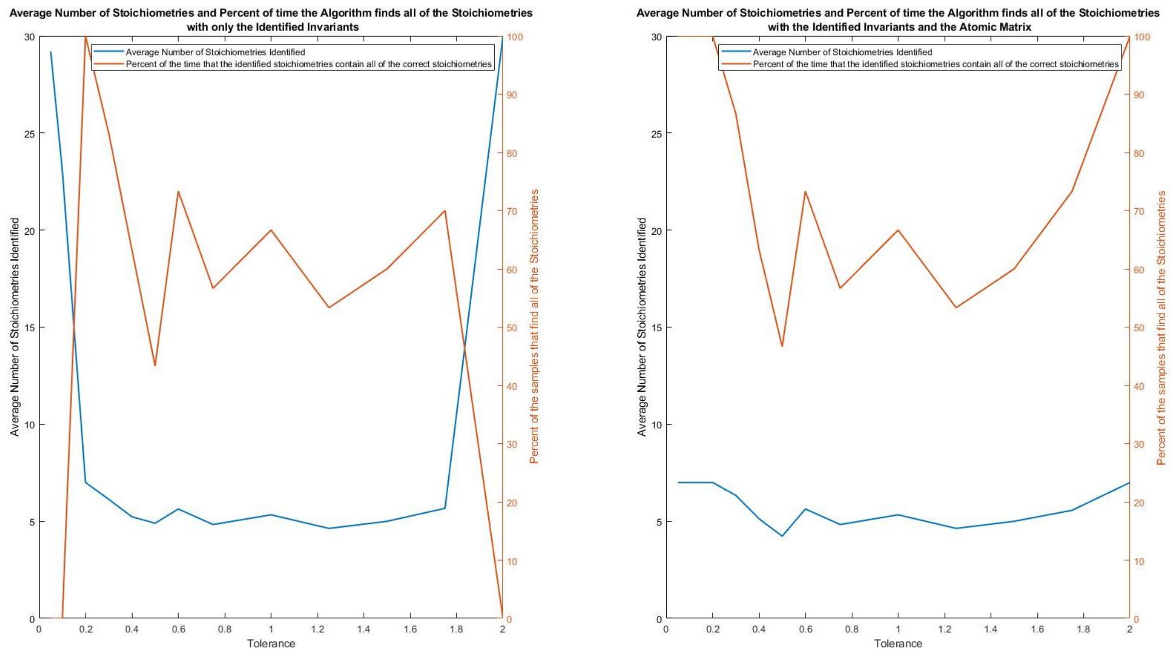


Figure 3.10 – Average number of stoichiometries identified vs the amount of tolerance for 30 samples both with and without the atomic matrix & the percentage of the time the identified stoichiometries contained all of the correct stoichiometries. Figure 3.10 shows the number of identified stoichiometries, and the percentage of the time that the identified stoichiometries contain all the correct stoichiometries both with and without the atomic matrix used. When not using the atomic matrix and at a tolerance of less than 0.2 there are many stoichiometries identified with none of them being correct. When considering that the only invariant being provided to the stoichiometric identification algorithm is a vector of zeros, the reasoning for this becomes clear. This hypothesis is further bolstered by adding the atomic matrix identifying all the stoichiometries, with the same results as when only the atomic matrix is used. At a tolerance of 0.2 the identified invariants appear to be reliable and numerous as they result in reliably identified stoichiometries without the atomic matrix. This infers that when the tolerance is very strict, the identified invariants are more likely to be accurate and therefore the

stoichiometries are very accurate. However, when the tolerance is increased, the number of accepted invariants appears to decrease, therefore the percentage chance of finding all the correct stoichiometries also decreases. The number of stoichiometries identified remains approximately constant as the number of invariants decreases between tolerance 0.2-1.8. This suggests that the reduction in the number of identified invariants does not reduce the amount of information within the invariants provided.

When the tolerance increases above 1.8, the number of identified invariants has once again decreased to zero, by comparing the two graphs in Figure 3.10 it can be seen that the lack on invariants when the tolerance is 1.8 means that without the atomic matrix it becomes impossible to find accurate stoichiometries.

This data would infer that when the system has 2% noise, the optimal tolerance value would be near to 0.2. However, by changing the amount of noise, the optimal tolerance value also changes. When considering this, it is also worth considering how the tolerance affects the invariants identified as well as the identified stoichiometries.

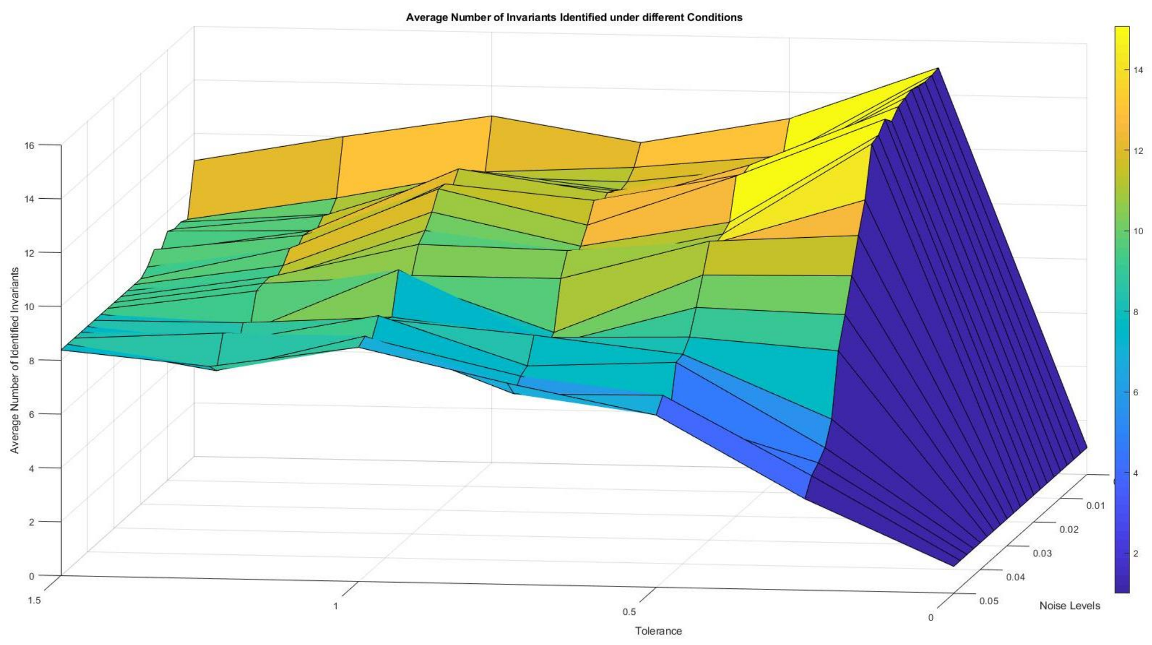


Figure 3.11 - Surface plot showing the relationship between noise within the data used, the tolerance that the algorithm considers acceptable and the average number of invariants that the algorithm identifies. The colour bar is a visual aid to assist in the identification of number of invariants identified.

Figure 3.11 shows that when the tolerance is below a threshold of approximately 0.2, the tolerance becomes too strict for the algorithm to be able to identify reaction invariants even with low noise levels. However, the number of identified invariants appears to be relatively stable with respect to tolerance for each noise level. There does appear to be a slight increase in the number of invariants as the noise level decreases which implies that there are some invariants that are difficult to identify unless the noise level is low regardless of the tolerance. By considering the number of identified invariants alongside the number of stoichiometries identified the accuracy of the invariants can be inferred, as if there are inaccuracies in the invariants it may result in internal inconsistencies and therefore that there are no feasible stoichiometries.

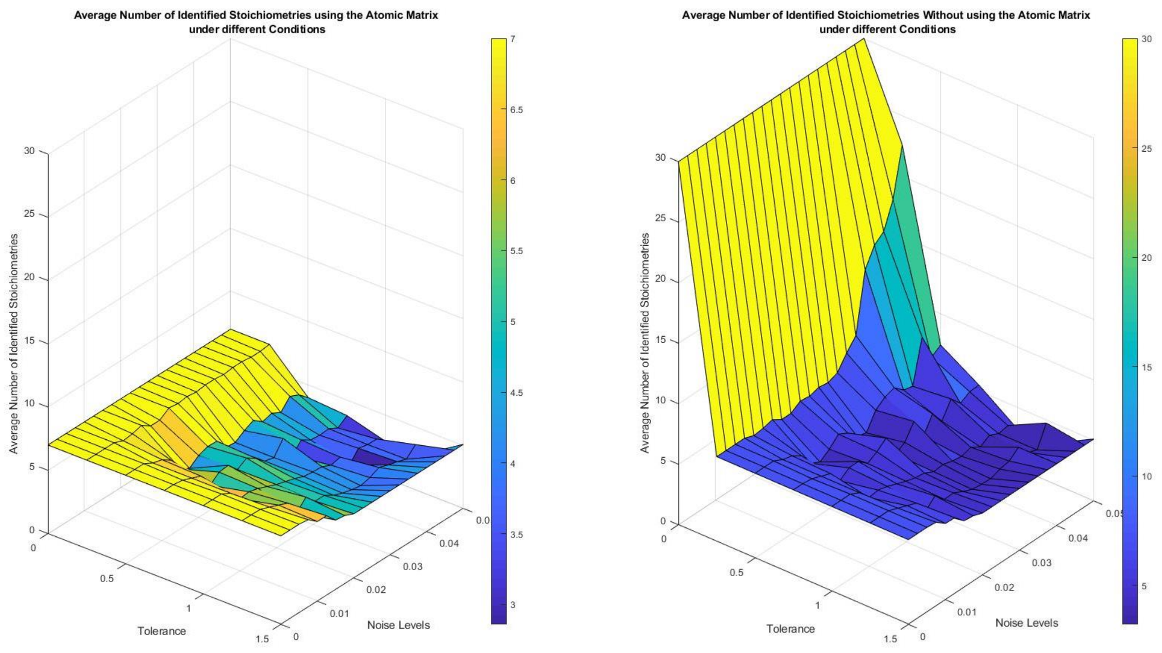


Figure 3.12 - The average number of Identified stoichiometries using the identified invariants under different conditions. Note: the zeros on the x-y plane have rotated so the surface can be seen more easily. The colour bar is a visual aid to assist in identifying the number of identified stoichiometries.

When the tolerance is very strict, there were no identified invariants, this means that when the atomic matrix is not used, there are very few constraints on the stoichiometric identification algorithm, and therefore the algorithm continues to identify stoichiometries indefinitely (in this case the algorithm was stopped if more than 30 potential stoichiometries were identified) and when the atomic matrix is used, the number of identified stoichiometries plateaus at seven identified stoichiometries. Excluding where the tolerance is zero the stoichiometries identified with and without the atomic matrix appear to be similar. The number of identified stoichiometries does not however provide information of the accuracy of the identified stoichiometries.

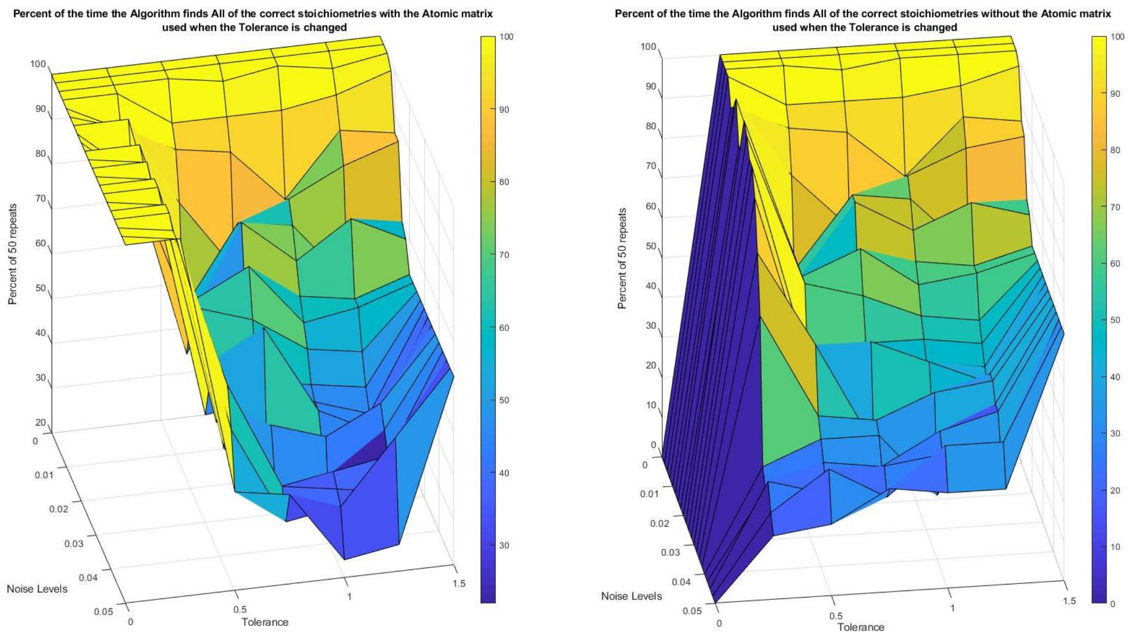


Figure 3.13 - With 50 samples what is the percentage of the time that the stoichiometry algorithm identifies all of the correct stoichiometries under different tolerance and noise levels. The colour bar is a visual aid to help identify the percentage of accurate stoichiometries. Note: the x-y plane is rotated compared to previous images.

The similarities in the two graphs implies two things. That the identified invariants provide the same information as the atomic matrix when the correct solutions are identified, and that false positive identified invariants are preventing the identification of stoichiometries when the atomic matrix is present.

3.2.6 Identification of stoichiometries from observed measurements

This reaction network was first analysed by Fuguitt and Hawkins (1945), (Fuguitt and Hawkins, 1947) the reaction conditions and normalised yields are summarised and shown in

Table 3-15.

Table 3-15 - Dataset from (Fuguitt and Hawkins, 1945) with the inferred yields on each individual component. * refers to data that was originally reported, but not observed. The data is split into the observed yields (Left half of the table), and the inferred observations from the

dataset (Right half of the table). The entire dataset generated by Fuguitt and Hawkins (1947) is not shown, only the data that is used within the algorithm.

Feed	**Temperature (°C)**	**Time (mins)**	**A**	**A+B**	**A+B+C**	**E**	**A**	**B**	**B+C**	**D**	**E**
A	189.5	1230	88.3	*	96.2	2.2	88.3	0	7.9	1.6	2.2
A	189.5	1230	88.2	*	95.7	1.3	88.2	0	7.5	3	1.3
A	189.5	3060	76.4	*	92.7	2.8	76.4	0	16.3	4.5	2.8
A	189.5	4920	64.8	*	88.9	5.8	64.8	0	24.1	5.3	5.8
A	189.5	7800	50.3	*	84.7	9.3	50.3	0	34.4	6	9.3
A	189.5	10680	37.5	*	82	12	37.5	0	44.5	6	12
A	189.5	15030	25.9	*	77.1	17	25.9	0	51.2	5.9	17
A	189.5	22620	14	*	73.9	21	14	0	59.9	5.1	21
A	189.5	36420	4.5	7.4	70.5	25.7	4.5	2.9	63.1	3.8	25.7

The method proposed in this work will only work using isothermal, complete time series data sets. This means that not all the data in the table can be used and it will need to be parsed into separate subsections. Whilst the concentration measurements are not present, the yields can be considered concentrations if the initial 'concentration' of the reactants is 100% and all other components are of initial 'concentration' 0%.

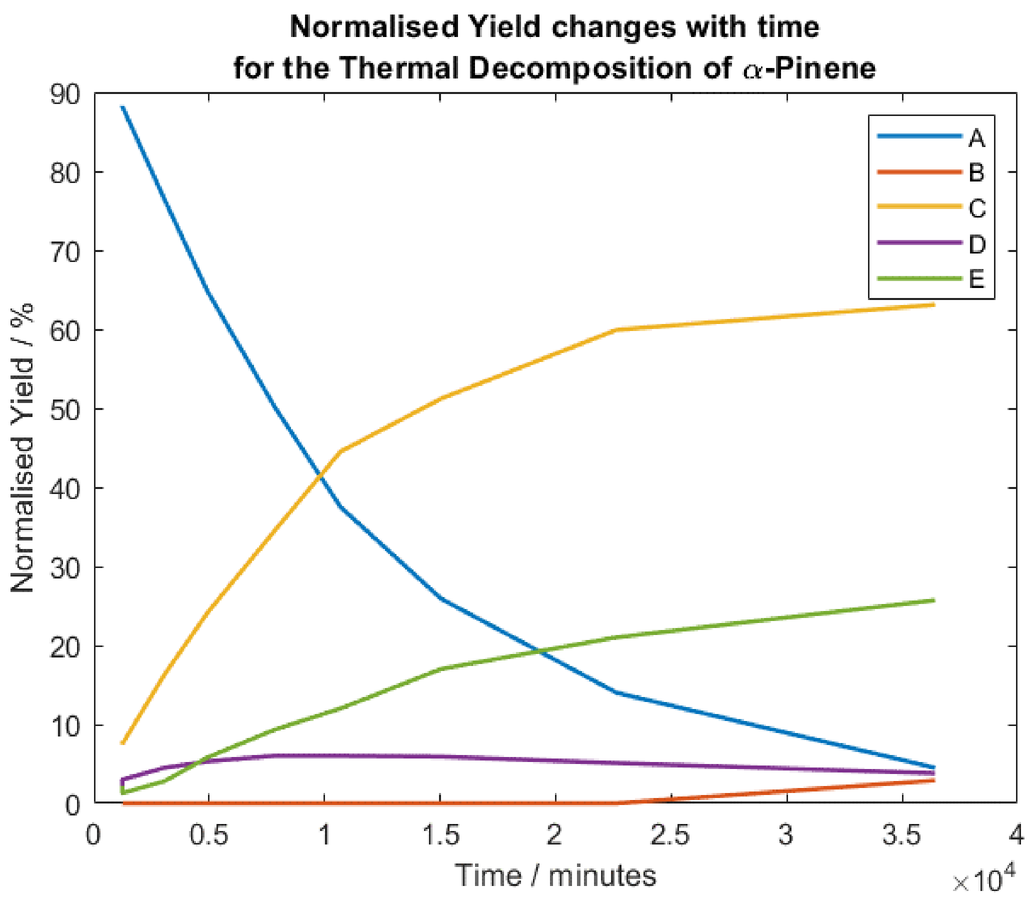


Figure 3.14 - plot of the change in normalised yields of the alpha-pinene reaction network

A priori information about the chemical reaction network can also be inferred from the concentration profiles (Figure 3.14). Component A is the initial reactant, B, C & E appear to be final products that do not react further, and component D is likely to be an intermediate component since the concentration appears to drop slightly towards the end of the time period being measured (this is also shown in

Table 3-15). The method has a single dataset to find invariants and one invariant is found (Table 3-16):

Table 3-16 - Identified Invariants from the α-pinene reaction network

	A	**B**	**C**	**D**	**E**
I_1	2	2	2	2	1

The only a priori information provided is that A must be a reactant, there are no other constraints on what can be identified. This gives a bit more freedom for the algorithm to search so more reactions can be found. This does mean that many of the reactions that are found, and their reverse reaction are feasible, and can be eliminated after the algorithm has been completed.

Table 3-17 - Identified Stoichiometries using the identified invariants.

	A	B	C	D	E
r_1	-3	0	3	0	0
r_2	0	-3	3	0	0
r_3	0	-1	2	0	-2
r_4	0	0	-1	1	0
r_5	0	0	0	0	0
r_6	0	0	1	0	-2
r_7	0	0	2	-1	-2
r_8	0	0	3	-3	0

There is some a priori information known about the process that was not included in the algorithm. For example, it was determined that component E is only ever a product, however this constraint was not implemented and the identified stoichiometries only include component E as a reactant. Due to the constraint not being implemented, reverse of the identified reactions are also feasible. This gives more feasible reactions and removes some of the infeasible reactions. By using the known a priori knowledge and knowing that the reverse of some of these reactions is feasible reveals the following set of possible reactions:

Table 3-18 - Feasible stoichiometries that are identified and fulfil all the a priori knowledge

	A	B	C	D	E
r_1	-1	0	1	0	0
r_2	0	1	0	-1	0
r_3	0	0	1	-1	0

Table 3-18 shows that not all the stoichiometries have been identified, however all the identified stoichiometries are accurate. Whilst this is a partial solution and there are some solutions that would be feasible but have not been identified. For example, Component D is not restricted and could be a product or a reactant, yet it only appeared as a reactant. This could be because component D is always at a relatively low concentration throughout the reaction run time and that there are some confounding variables due that are too complex to be considered by the algorithm in this form, with the data provided.

A flaw of this example is that it is an isomerisation reaction network. This means that the number of invariants within the reaction network are very limited, upon looking at the atomic matrix of the components within the alpha-pinene reaction network the rank of the matrix is one and from that we can infer that there is only one invariant within this reaction network. This could be translated as meaning that there is a very small amount of information about the system that can be identified using this method.

A consideration that may be affecting the results is that the data provided is not truly time series data, it is instead yield data from multiple experiments. This could lead to a variety of errors in the data used which are difficult to quantify,

3.3 Conclusions

This method has been shown to be effective in identifying reaction invariants and reaction stoichiometries for chemical reaction networks with some reliability. There are some procedures that can be followed that have been shown to improve the effectiveness of the algorithm. Using Datasets that have minimal noise levels to the data and using between 10 and 30 data points provides most reliability of results.

A variable that should be considered when setting up the algorithm is the scale of the data being used, and the acceptable tolerance. It may be prudent to normalise the data or treat the data appropriately.

With the alpha-pinene example it has been shown that this algorithm has flaws if the data is not formulated appropriately and/or this reaction type is inappropriate for this method.

4 Chapter 4 Kinetic structure identification

4.1 Overview

This chapter has the goal of automatically identifying reaction kinetic models using MILP. As with the Stoichiometric identification chapter, this will be using the simulation of the biodiesel production from triglyceride reaction network. This is because the effect of the quantity, and quality of the data used is being investigated and a simulated environment is ideal.

The methodology of this chapter shows the general approaches which are common for the entire method.

The results first show the proof of concept, and that the method can be effective. This is followed by comparing different differential and integral approximation techniques. These results are compared at a low-resolution level, to reduce the number of approximation techniques.

The reduced number of techniques are then tested more thoroughly against datasets with artificial gaussian noise added to the measurements to emulate real system noise. This section also shows the efficacy of the method for real measurement data that would have either measurement or system noise corrupting the measurements.

The efficacy of the approaches being considered are evaluated when the number of measurements is reduced. Taking measurements is labour expensive (and potentially financially expensive) and ideally the minimum number of measurements would be used. Finally the efficacy of the approaches being considered are compared using a central composite design to show the combined effect of the two confounding variables (noise and number of measurements).

4.2 Background

The methodology applies the theory of chemical kinetics. In its generalized form of the rate law, or rate equation is typically written as follows:

$$\boldsymbol{\frac{dC}{dt} = k \prod_{i} C_i^{m_i}} \qquad \text{Equation 4-1}$$

Where the change in concentration of a component within a reaction network is a function of the molar concentration of a component. When appropriate measurements are available, approximation of the differential term can be solved a variety of ways including the standard Euler method, the Taylor series approximation for example. The Molar Concentration term ($\boldsymbol{C}$) can be identified with some assumptions: That the order of the reaction is zero, one or two- where $[C_1]^2$ and $[C_1].[C_2]$ are both considered to be second order reaction, all possibilities of this can be approximated from the measured concentrations. The rate constants ($\boldsymbol{k}$) should be sparse, therefore the number of concentration combinations that are contributing to the equation will also be sparse. This leaves the rate constants to be the unknown that the linear programming technique is designed to identify. Assuming that the system is operating at constant pressure and temperature, these should be constant, and an accurate approximation of the system being identified. Alternatively, the integral of this equation can be solved:

$$\Delta\boldsymbol{C} = \boldsymbol{k} \int_{t_1}^{t_2} \boldsymbol{C}_i^{m_i}\, dt \qquad \text{Equation 4-2}$$

There the change in molar concentration ($\Delta\boldsymbol{C}$) is the change over a time period. The integral is to be approximated numerically from the combination of viable reactions considered (zero, first and second order possibilities). This can be done in a variety of ways, including the trapezium rule or numerical integration using Simpsons rule. As with the differential approach, the integral approach requires the $\boldsymbol{k}$ vector to be sparse, and for the reaction to be taking place in isothermal and isobaric conditions.

4.3 Method

Figure 4.1 shows a flowchart overview of how the algorithm operates. A set of written instructions of how to generate the algorithm as if it were being used on the measurements from a real reactor is below.

1. Collect Data.

The amount of data required varies throughout the results section of this algorithm and therefore multiple runs may be required to generate sufficient data. If this data is graphically displayed, some *a priori* information should be identifiable which can be used if the user desires. An example of this *a priori* information would be identifying which components within the reaction is a reactant, a product, or an intermediate product. The collected data should be separated into training, testing and validation sets.

2. Transform the Data.

All the possible kinetic structures and the relevant approximation need to be identified. The approximation of the differential or integral can be done by one of several approaches discussed in the results. The different approaches are considered and evaluated in 4.5.

3. Set up MILP algorithm.

This transformed data was then configured into the MILP form and the suitable binary and slack variables should be included. As discussed in section 3.1.2 a slack variable is introduced to act as an error term and binary variables are introduced to constraint the kinetic structure to be both sparse and can be used to manipulate the number of reactants and products within a reaction. These inequality constraints are shown below for a differential approach, the only change for an integral approach would be that Equation 4-3a & b modified to be in integral form:

$$-\frac{d[C]}{dt} + [C] - \varepsilon \leq tol$$ Equation 4-3a

$$\frac{d[C]}{dt} - [C] - \varepsilon \leq tol$$ Equation 4-3b

$$[C_i] + (UB * tol)\sigma_{1,i} \leq UB - tol$$ Equation 4-3c

$$-[C_i] - UB\sigma_{1,i} \leq 0$$ Equation 4-3d

$$-[C_i] - (LB + tol)\sigma_{2,i} \leq -(LB + tol)$$ Equation 4-3e

$$[C_i] + (LB + tol)\sigma_2, i \leq 0$$ Equation 4-3f

$$-\sum\sigma_1 - \sum\sigma_2 + \sum\sigma = 0$$ Equation 4-3g

$$\sum\sigma \leq N_c$$ Equation 4-3h

$$-\sum\sigma \leq 1$$ Equation 4-3i

Other optional constraints identified from the a priori information can be added as a set of constraints. The cost function is based upon minimising two terms, the $\sum\sigma$ and ε. There are integer constraints set up for all the binary variables (all σ and the $\sum\sigma$ terms).

4. Run the MILP.

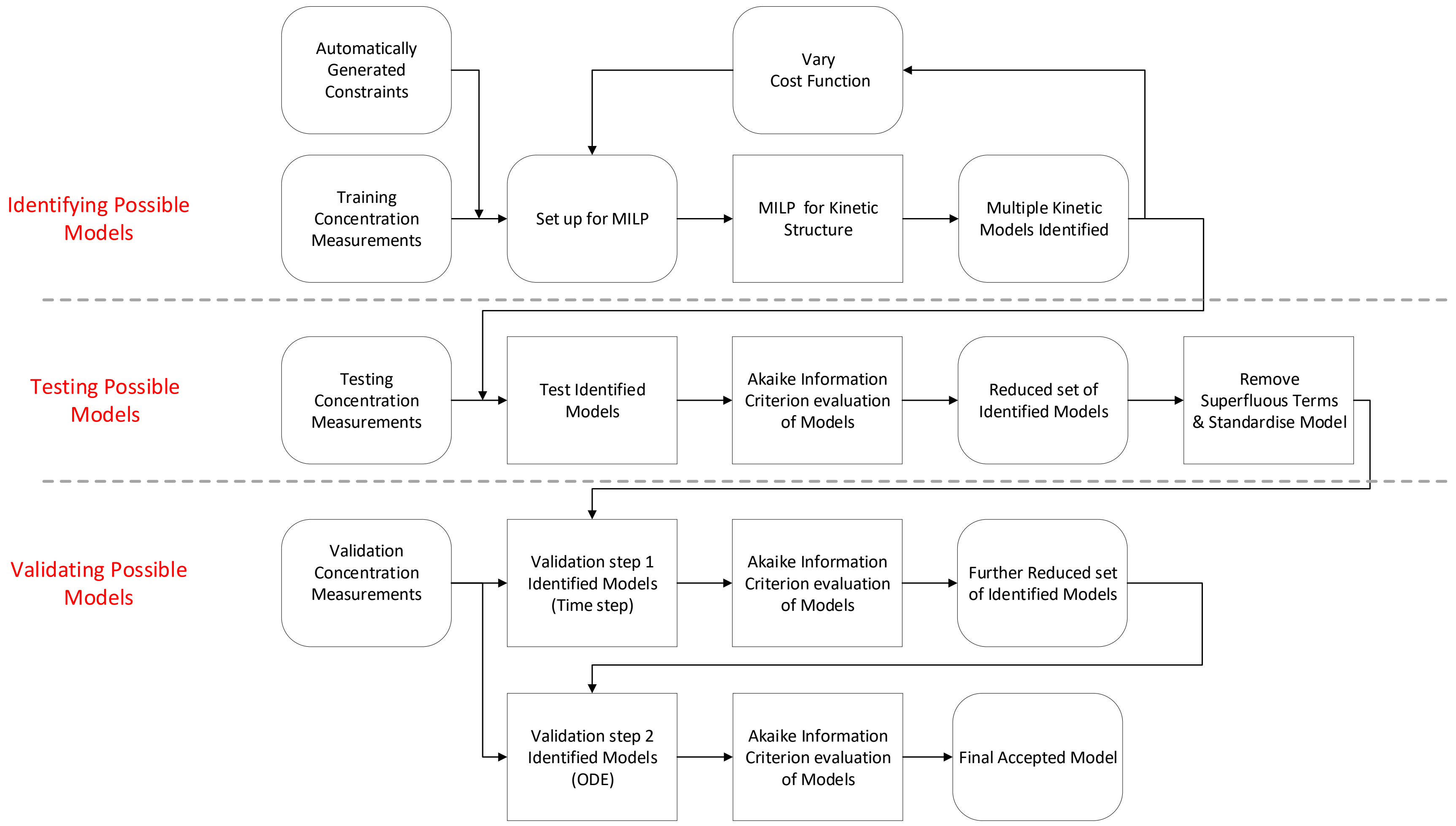
The MILP was run multiple times and collecting the results after each run. There were repeats at the same condition and repeats at a selection of the coefficients of the cost function to put emphasis on the minimisation of one term over another.

5. Test and Filter Identified Models.

Using the test dataset, the identified models are simulated through a stepwise approximation of the differential, and evaluating the model using the Akaike Information Criterion (AIC). The top 5 models are then simulated using Matlab's *ode45* function and the quality of the model is again evaluated using AIC. The filtering process includes the removal of some the superfluous terms and averaging the coefficients within one reaction to have a single kinetic coefficient value.

6. Validate Model.

The remaining models were validated against the validation data set, in a similar form to the testing step and evaluating using the AIC to reduce to a final model.



Identifying Possible Models
Automatically Generated Constraints
Training Concentration Measurements
Set up for MILP
Vary Cost Function
MILP for Kinetic Structure
Multiple Kinetic Models Identified
Testing Possible Models
Testing Concentration Measurements
Test Identified Models
Akaike Information Criterion evaluation of Models
Reduced set of Identified Models
Remove Superfluous Terms & Standardise Model
Validating Possible Models
Validation Concentration Measurements
Validation step 1 Identified Models (Time step)
Akaike Information Criterion evaluation of Models
Further Reduced set of Identified Models
Validation step 2 Identified Models (ODE)
Akaike Information Criterion evaluation of Models
Final Accepted Model

Figure 4.1 - Flow chart overview of the identification of kinetic models, where the hard cornered bubbles refer to operations performed on the data, and the round cornered bubbles refer to data, or results. The dashed line separates the processes into the training, testing and validation steps of the algorithm.

4.4 Results and discussion

There are two approaches being considered when identifying the kinetic structure: the differential form, or the integral form. It has been identified prior to starting this investigation that components 1 and 2 are the reactants, and the other components are either intermediates or products. By setting up the algorithm under the conditions shown in Table 4-1, and the two approaches are compared to the generated model in Table 4-2.

Table 4-1 - Operating conditions for the results shown in Table 4-2.

Condition	**Values**
Training Dataset	Initial Conditions - [1,3,0,0,0,0]
Testing Dataset	Initial Conditions – [3,4,0,0,0,0]
Validation Dataset	Initial Conditions – [2,2,0,0,0,0]
Data points per Dataset (evenly spread)	75
Gaussian Noise Added	0%
Differential Approximation Method	Forward Euler Method
Integral Approximation Method	Trapezium Rule

It is anticipated that these results are unlikely to be accurate, this is because the training data is a single dataset as it leaves the possibility of overfitting the model to the training data. Both the differential and integral approaches identified a similar model. The results show that the first reaction was found for both methods, however the sequential reactions were less likely to be found accurately with the third reaction unable to be found. This can be explained by looking at the training data. The initial conditions for training have triglyceride (TG) and Methanol (MeOH) as the highest values. It therefore follows that the reaction involving TG and MeOH is identified, but the sequential reactions have been more difficult to identify for the algorithm. Since the change in concentration of the Monoglyceride (MG) and Glycol (GL) were relatively low at all time points there was less information within the data about these components. It appeared that using no model has a small enough error for the algorithm to consider it a reasonably accurate model (see Figure 4.2).

Table 4-2 - Results of the conditions shown in Table 4-1 where the generated model is compared the two approaches being considered.

	$[TG].[MeOH]$			$[DG].[MeOH]$			$[MG].[MeOH]$		
	Generated Model	Differential Approach	Integral Approach	Generated Model	Differential Approach	Integral Approach	Generated Model	Differential Approach	Integral Approach
$\frac{d[TG]}{dt}$	-0.4	-0.367	-0.407	0	0	0	0	0	0
$\frac{d[MeOH]}{dt}$	-0.4	-0.367	-0.407	-0.5	0	0	-0.2	0	0
$\frac{d[DG]}{dt}$	0.4	0.367	0.407	-0.5	-0.242	-0.302	0	0	0
$\frac{d[BD]}{dt}$	0.4	0.367	0.407	0.5	0	0.302	0.2	0	0
$\frac{d[MG]}{dt}$	0	0	0	0.5	0.242	0.302	-0.2	0	0
$\frac{d[GL]}{dt}$	0	0	0	0	0	0	0.2	0	0

These results demonstrate that a single training dataset provided a good approximation of the first reaction. By including further datasets, which included data on the second and/or third reactions the algorithm demonstrates the potential to find all reactions to higher accuracy.

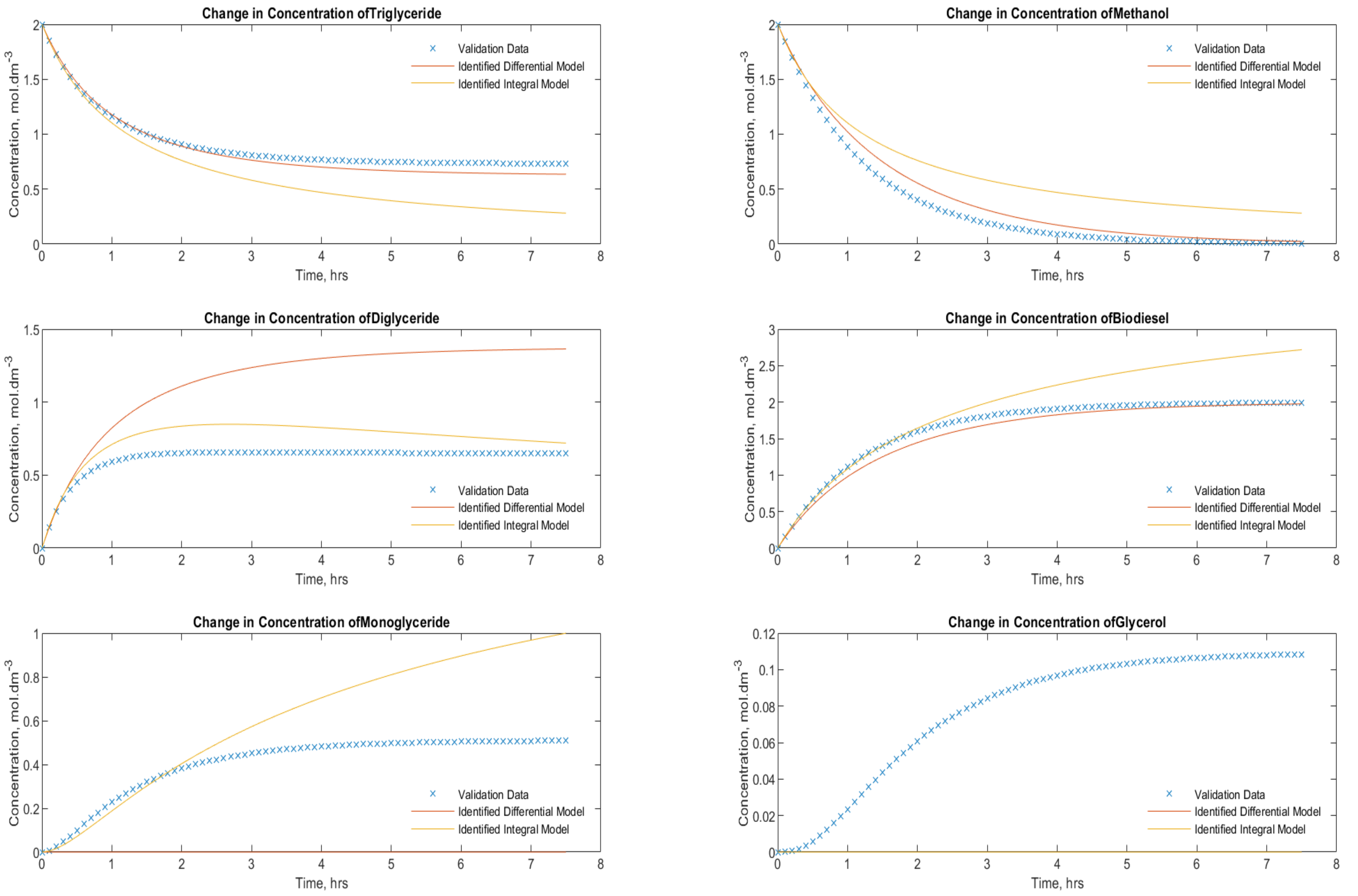


Figure 4.2 - The Change in Concentration of all components, with validation data, and the differential and integral approaches' models.

Increasing the amount of training data means running separate simulations of the reaction network under different initial conditions. The possible inputs (initial conditions and subsequent data) are categorised in Table 4-3.

Table 4-3 - Variations of all possible initial conditions. Where High, Medium and Low could be in any possible position within the vector.

Position:	**1**	**2**	**3**	**4**	**5**	**6**
Initial Condition	High	Zero	Zero	Zero	Zero	Zero
Initial Condition:	High	Low	Zero	Zero	Zero	Zero
Initial Condition:	Medium	Medium	Zero	Zero	Zero	Zero

Rather than implementing numerical values, the initial conditions can be three levels: High, Medium and Low. Where a high value is large relative to the low value- for example, an order of magnitude larger, and the 'medium' level refers to two values that are equivalent. These possible conditions are in all possible positions (Component number) of each of these levels. This results in sixty-six initial conditions. Increasing the amount of training data to include sixty-six datasets with 75 measurements results are shown in Table 4-4.

These results (Table 4-4) are a significant improvement on the results with a single training set (Table 4-2). All the relevant reactions have been identified in for all the differential equations. The concentration profiles of the differential equations are shown in Figure 4.3 From the concentration profiles it the models for MG and GL have the least accurate model. However, it is worth noting that these are identified as the third reaction in a sequence, therefore due to error accumulation from the first two reaction models the graphical results will always be less accurate for the third reaction than the for the first reaction in the sequence.

Table 4-4 - Models Identified using the differential and integral approaches. Sixty-six datasets were used to generate these models.

	$[TG].[MeOH]$			$[DG].[MeOH]$			$[MG].[MeOH]$		
	Generated Model	Differential Approach	Integral Approach	Generated Model	Differential Approach	Integral Approach	Generated Model	Differential Approach	Integral Approach
$\frac{d[TG]}{dt}$	-0.4	-0.253	-0.374	0	0	0	0	0	0
$\frac{d[MeOH]}{dt}$	-0.4	-0.253	-0.374	-0.5	-0.277	-0.392	-0.2	-0.116	-0.14
$\frac{d[DG]}{dt}$	0.4	0.253	0.374	-0.5	-0.277	-0.392	0	0	0
$\frac{d[BD]}{dt}$	0.4	0.253	0.374	0.5	0.277	0.392	0.2	0.116	0.14
$\frac{d[MG]}{dt}$	0	0	0	0.5	0.277	0.392	-0.2	-0.116	-0.14
$\frac{d[GL]}{dt}$	0	0	0	0	0	0	0.2	0.116	0.14

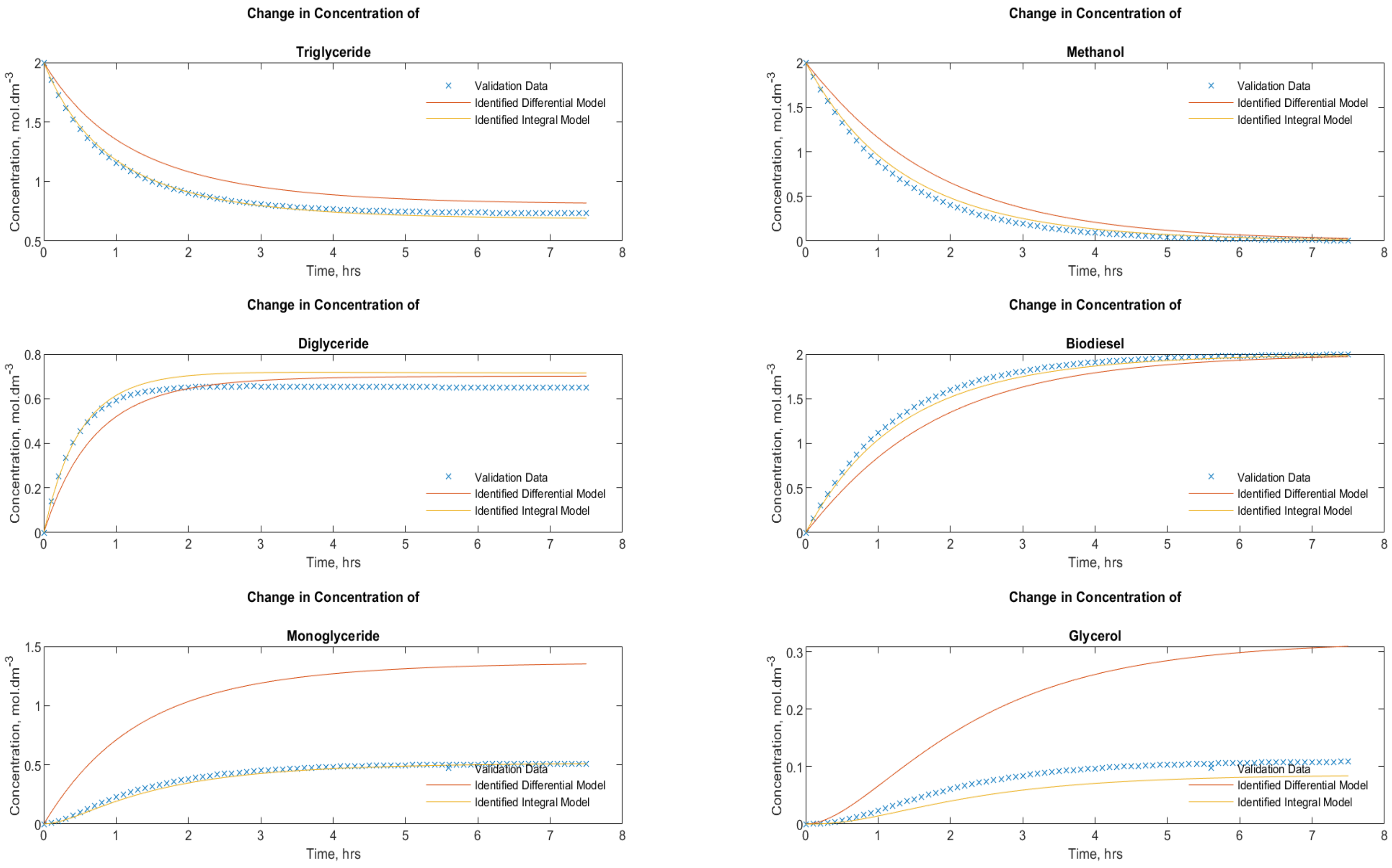


Figure 4.3 - Concentration Profiles of the validation data and the Differential and Integral Identified models. Sixty-six sets of training data were used to generate these models.

Comparing the identified models to the original (Generated) model (Table 4-4), numerically the integral approach is providing more accurate values than the differential approach from the same training data. This shows that both approaches are converging towards the same point with different levels of precision.

4.5 Differential Approximation Methods

The differential and integral need to be approximated from measurements. There are a variety of methods of approximating the differential and integral terms. For the previous examples, the Forward Euler method (differential) and the trapezium rule (Integral) have approximated the respective differential and integral terms. Several methods will be considered including: the Euler method, Taylor Series, and Differentiated Spline approaches for differential approximation, and the trapezium rule, Simpsons rule and Integrated Spline approximations for the integral approximations.

4.5.1 Euler Method

The Euler Method has three subcategories that can all provide accurate approximations depending on the function being approximated. The Forward Euler method is the standard formulation of the Euler method; however, the Backward Euler method and Central Euler methods are also variants that can be more useful under certain circumstances (Chapra, 2015). The Euler method uses the current point and an adjacent point to create a straight line. The approximations of the differential and integral are calculated from time series data, with initial conditions $[1,5,0,0,0,0]$ with 75 measurements of each component across the reaction period of 7.5 hours, shown in Figure 4.4. The three Euler methods considered can be described by Equation 4-4. Part a shows the Forward Euler method, b the Backward Euler Method, and c the Central Euler method. For all three of these equations, the measurements are represented by y and the respective time point is represented by x, for the current measurement n.

$$\frac{dy}{dx_n} \cong \frac{y_{n+1} - y_n}{x_{n+1} - x_n}$$ Equation 4-4a

$$\frac{dy}{dx_n} \cong \frac{y_n - y_{n-1}}{x_n - x_{n-1}}$$ Equation 4-4b

$$\frac{dy}{dx_n} \cong \frac{y_{n+1} - y_n - y_{n-1}}{x_{n+1} - x_{n-1}}$$ Equation 4-4c

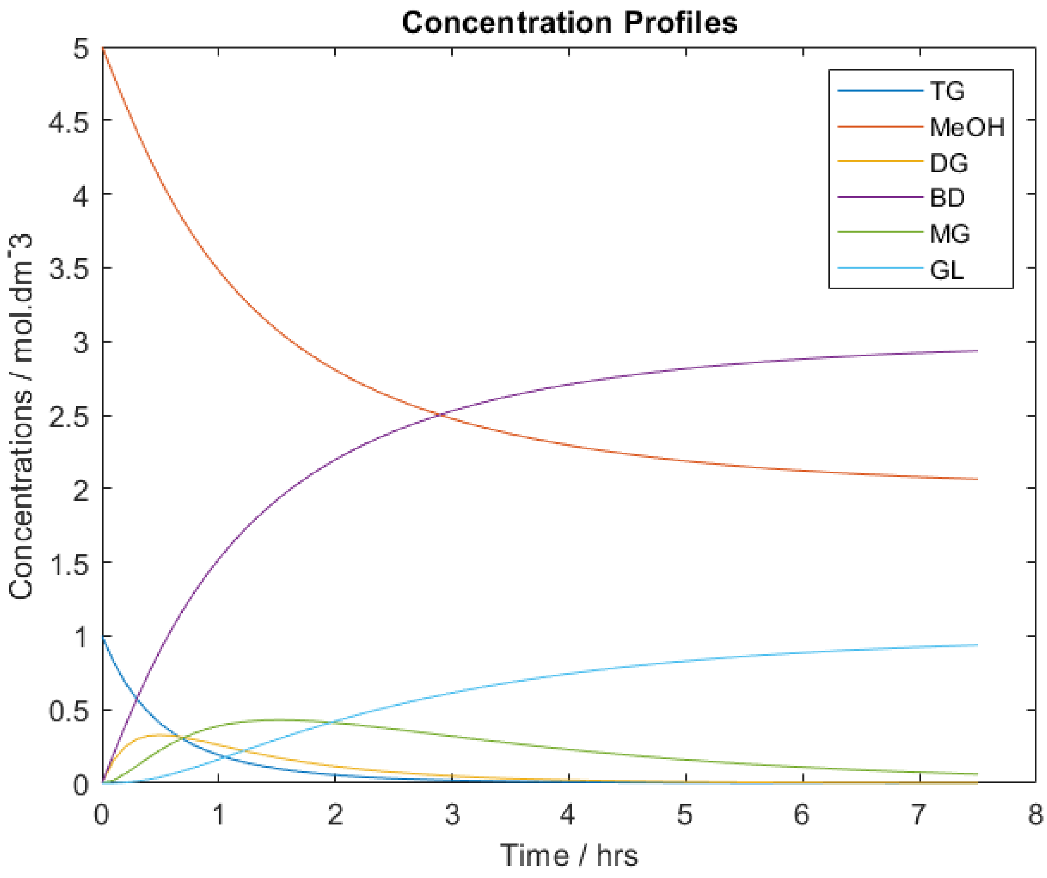


Figure 4.4 - Concentration profile (clean measurements) of the data being used to approximate the differential and integral via different methods.

3.1.9.1 Forward Euler Method

The forward Euler method has already shown to be effective in the previous examples (See section 4.4). The Forward Euler method calculates the curve using the current datapoint and the adjacent future timeseries datapoint to generate the approximation of the differential. The approximation of the differential is plotted in Figure 4.5, the identified model is displayed in

Table 4-5 and a comparison of the model identified to the validation dataset is displayed in Figure 4.6.

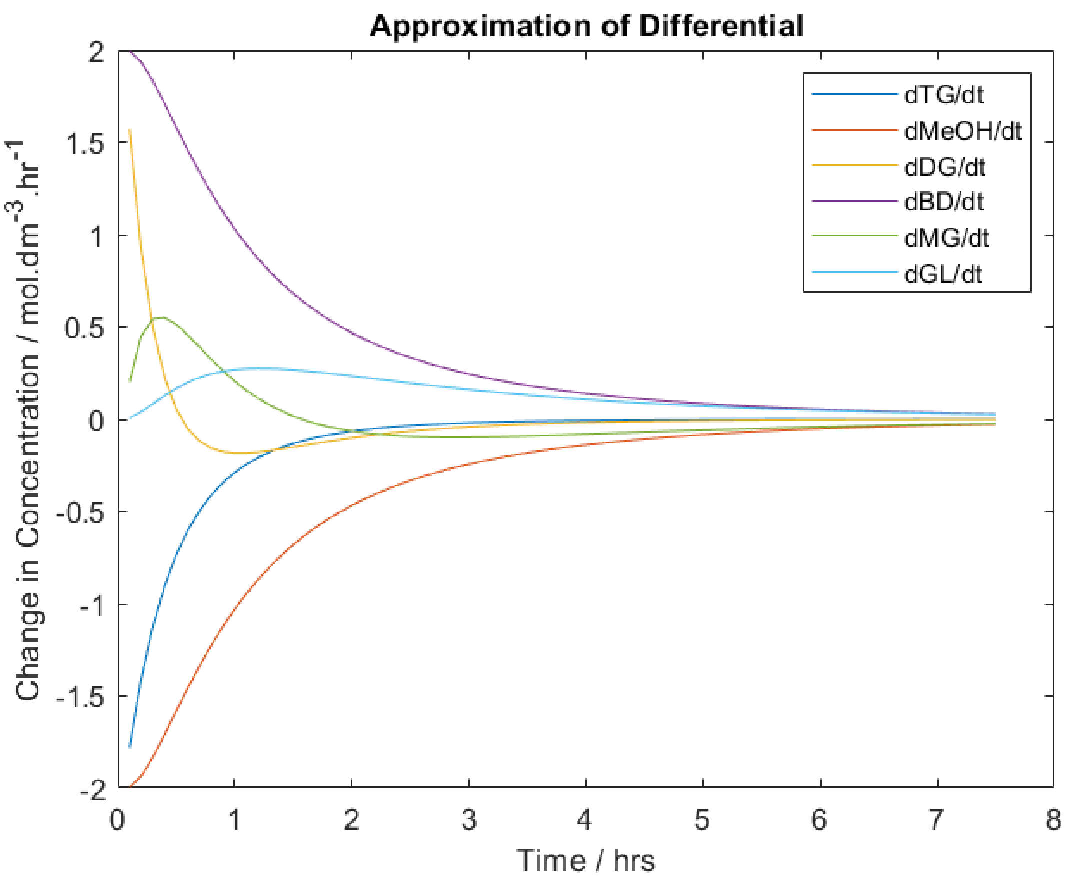


Figure 4.5 - Approximation of the differential using the Forward Euler Method.

Table 4-5 – The Model identified using the forward Euler method to identify the differential.

	$[TG].[MeOH]$		$[DG].[MeOH]$		$[MG].[MeOH]$	
	Generated Model	Differential Approach	Generated Model	Differential Approach	Generated Model	Differential Approach
$\frac{d[TG]}{dt}$	-0.4	-0.253	0	0	0	0
$\frac{d[MeOH]}{dt}$	-0.4	-0.253	-0.5	-0.277	-0.2	-0.116
$\frac{d[DG]}{dt}$	0.4	0.253	-0.5	-0.277	0	0
$\frac{d[BD]}{dt}$	0.4	0.253	0.5	0.277	0.2	0.116
$\frac{d[MG]}{dt}$	0	0	0.5	0.277	-0.2	-0.116
$\frac{d[GL]}{dt}$	0	0	0	0	0.2	0.116

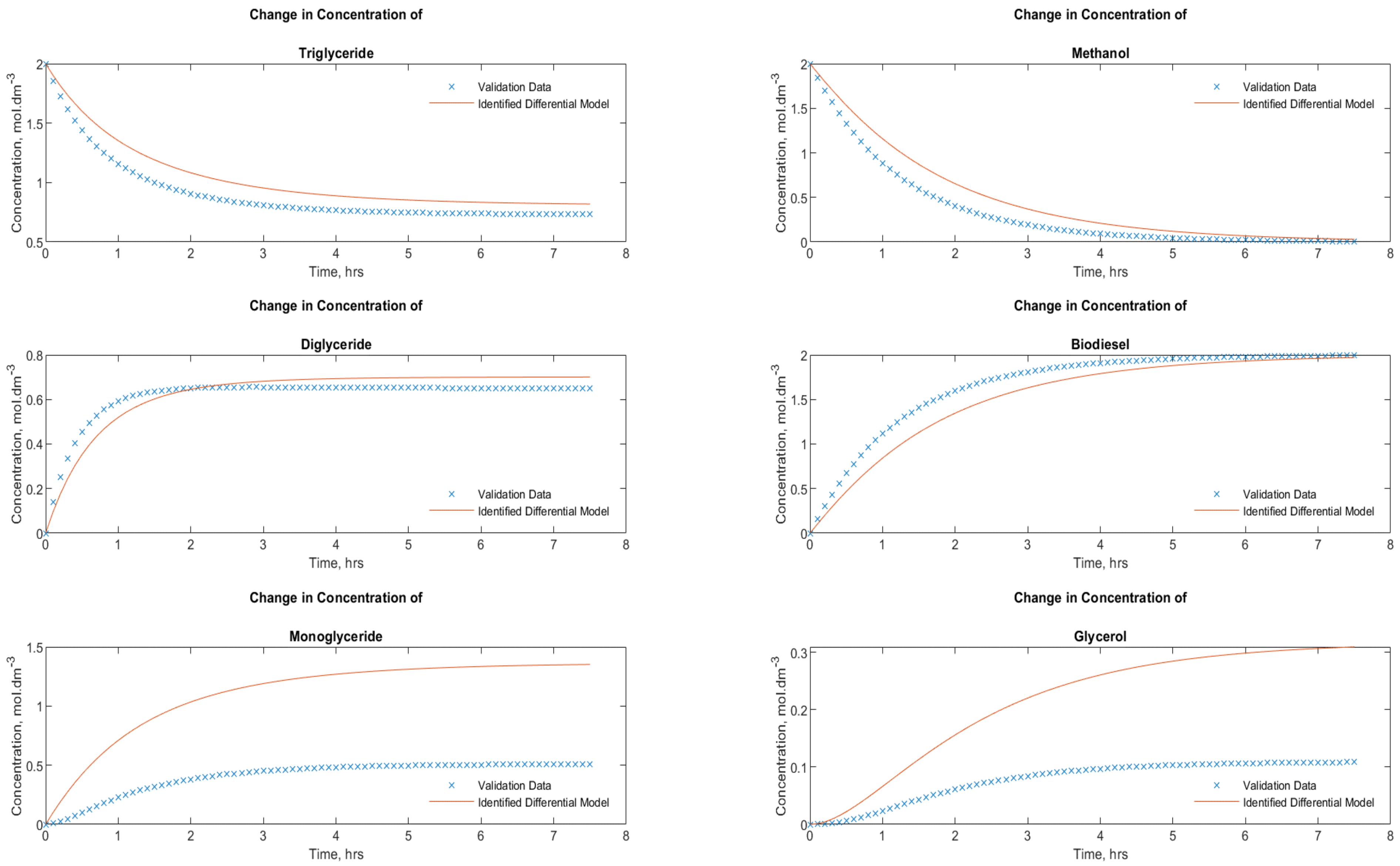


Figure 4.6 - Plotted Model identified using the Forward Euler Method approximation of the differential.

The approximation found all of reactions for each reacting component, although the identified rate constant coefficients were smaller in magnitude than the correct values in all cases. The accumulated errors from the first two reactions result in in the models of the third reaction appearing to have a large effect on the MG and GL concentration profiles, but the models appear to follow the trend of the data. This shows that the Forward Euler method has a valid approximation of the differential for the purposes of this method.

3.1.9.2 <u>Backward Euler Method</u>

The Backwards Euler method uses the current datapoint and the adjacent past timeseries datapoint to calculate an approximation of the differential. This means that the differential cannot be identified at the first measurement as there is no measurement of the concentration prior to beginning the reaction. The differential therefore has one less useable datapoint at the beginning of the reaction. Figure 4.7 - Backward Euler Method for approximating the differential term. Figure 4.7 shows an approximation of the differential calculated using the backwards Euler method and

Table 4-6 shows the kinetic model that is identified from this approximation.

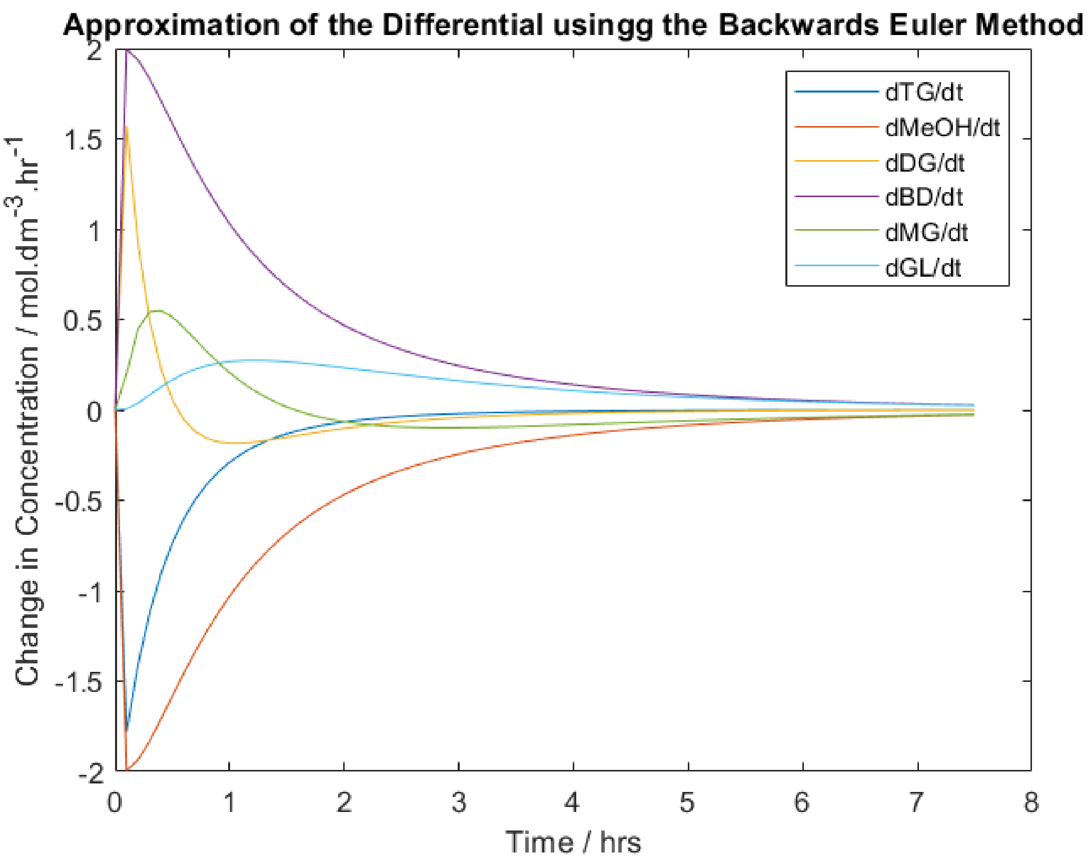


Figure 4.7 - Backward Euler Method for approximating the differential term.

Table 4-6 - Identified model when using the backward Euler method

	$[TG].[MeOH]$		$[DG].[MeOH]$		$[MG].[MeOH]$	
	Generated Model	Differential Approach	Generated Model	Differential Approach	Generated Model	Differential Approach
$\frac{d[TG]}{dt}$	-0.4	0	0	0	0	0
$\frac{d[MeOH]}{dt}$	-0.4	0	-0.5	-0.128	-0.2	0
$\frac{d[DG]}{dt}$	0.4	0	-0.5	0	0	0
$\frac{d[BD]}{dt}$	0.4	0	0.5	0.128	0.2	0
$\frac{d[MG]}{dt}$	0	0	0.5	0	-0.2	0
$\frac{d[GL]}{dt}$	0	0	0	0	0.2	0

The identified structure is dissimilar to the generated model (

Table 4-6), but the approximated differential (Figure 4.7) is very similar to the differential identified by the Forward Euler method (Figure 4.5). The difference between the approximations of the two methods is at the first datapoint where the backwards Euler method cannot approximate the differential. During the first few datapoints there are large changes in the concentration. Since no change in concentration can be observed using this approximation method this measurement is a conflicting datapoint to the rest of the dataset. Conversely, the final datapoints contained very small changes in concentration – these small changes can be appropriate to a variety of models and are therefore less useful in identifying the correct model. The identified models are not shown graphically because the model has significant and large errors.

The Backward Euler method cannot be used as it cannot extract the information from some of the important datapoints at the start of process.

3.1.9.3 *Central Euler Method*

The Central Euler Method identifies the gradient at a point by using the adjacent past timeseries datapoint, and the adjacent future timeseries datapoint to create the tangent. It would follow that the Central Euler method would have some similar problems to the Backward Euler method as it uses the adjacent past datapoint and the adjacent future datapoint to identify the gradient. To combat this, the first datapoint is calculated by the Forward Euler method, and the sequential approximations use the central Euler method. The approximation of the differential is shown in Figure 4.8Figure 4.8, the model identified is shown in Table 4-7.

The differential of this model again appears to be accurate, and like the forward Euler method. There was a breakpoint (a vertex) where the method switches between the forward Euler method and the Central Euler method – This discontinuity could provide some problems evidenced by the backwards Euler method where the first few points contain significant information.

This method worked successfully in its prediction of the differential. However, the first, and most vital datapoint must be calculated with the Forward Euler method. When compared to the Forward Euler method, the first two reactions have comparable coefficient values, however the magnitude of the coefficient for the third reaction has a more significant difference between the models. Overall, the Central Euler method does not improve the quality of the models significantly enough for it to be consider over the Forward Euler method.

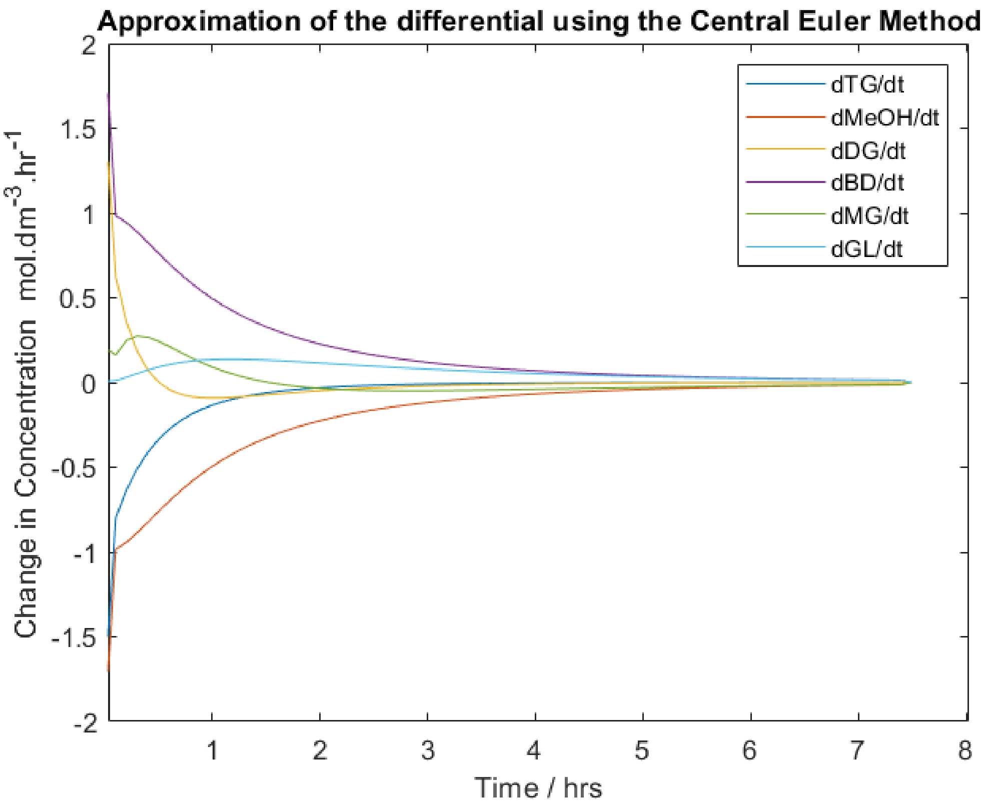


Figure 4.8 - Approximation of the Differential using the Central Euler method.

Table 4-7 - The Identified models using Central Euler Method to approximate the differential.

	$[TG].[MeOH]$		$[DG].[MeOH]$		$[MG].[MeOH]$	
	Generated Model	Differential Approach	Generated Model	Differential Approach	Generated Model	Differential Approach
$\frac{d[TG]}{dt}$	-0.4	-0.277	0	0	0	0
$\frac{d[MeOH]}{dt}$	-0.4	-0.277	-0.5	-0.274	-0.2	-0.085
$\frac{d[DG]}{dt}$	0.4	0.277	-0.5	-0.274	0	0
$\frac{d[BD]}{dt}$	0.4	0.277	0.5	0.274	0.2	0.085
$\frac{d[MG]}{dt}$	0	0	0.5	0.274	-0.2	-0.085
$\frac{d[GL]}{dt}$	0	0	0	0	0.2	0.085

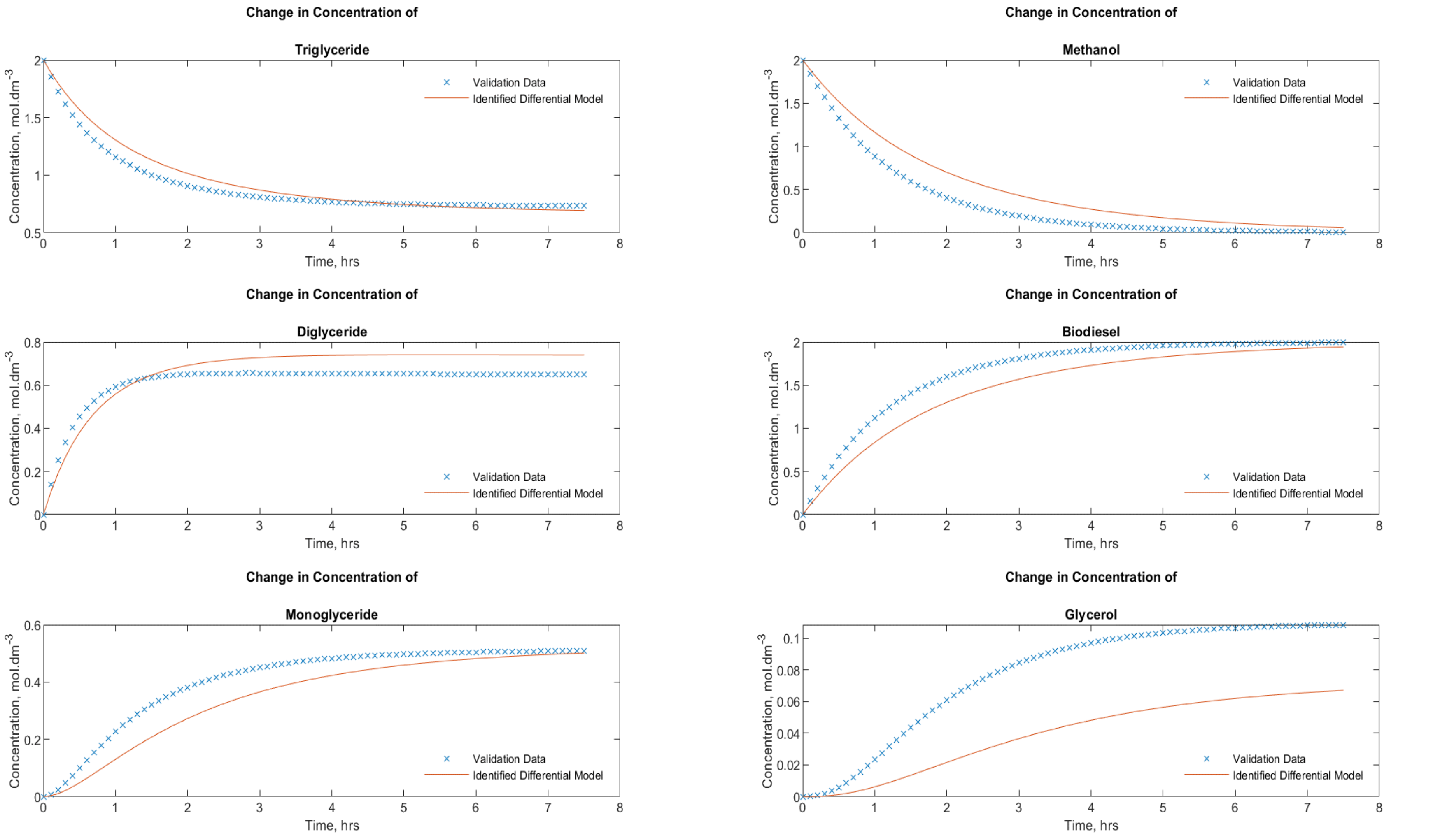


Figure 4.9 - Plotted Model identified using the Central Euler Method approximation of the differential.

4.5.2 Taylor Series

The Taylor series uses the (already identified) Forward Euler method to identify a second order differential so that the first order differential can be identified with more accuracy. The differential model is shown in Figure 4.10, the identified model is shown in Table 4-8, and the concentration profile of the model is shown in Figure 4.11. The Taylor series expansion is computed by MATLAB, but the forward Taylor series expansion is shown in Equation 4-5.

$$f(x_{i+1}) = f(x_i) + f'(x_i)\Delta x + \frac{f''(x_i)}{2!}(\Delta x)^2 + \cdots \qquad \text{Equation 4-5}$$

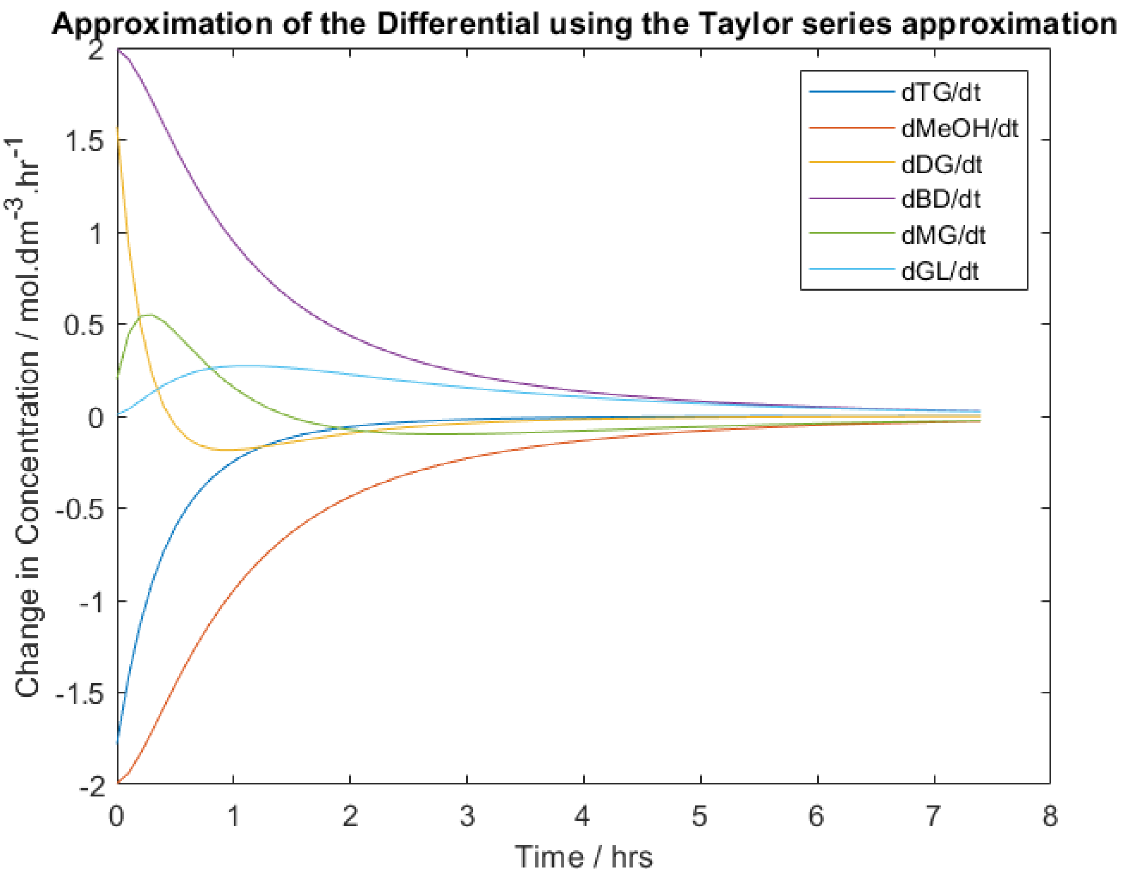


Figure 4.10 - Differential approximation using the Taylor series approximation.

Table 4-8 - Identified model using the Taylor series method.

	$[TG].[MeOH]$		$[DG].[MeOH]$		$[MG].[MeOH]$	
	Generated Model	Differential Approach	Generated Model	Differential Approach	Generated Model	Differential Approach
$\frac{d[TG]}{dt}$	-0.4	-0.282	0	0	0	0
$\frac{d[MeOH]}{dt}$	-0.4	-0.282	-0.5	-0.330	-0.2	-0.128
$\frac{d[DG]}{dt}$	0.4	0.282	-0.5	-0.330	0	0
$\frac{d[BD]}{dt}$	0.4	0.282	0.5	0.330	0.2	0.128
$\frac{d[MG]}{dt}$	0	0.282	0.5	0.330	-0.2	-0.128
$\frac{d[GL]}{dt}$	0	0	0	0	0.2	0.128

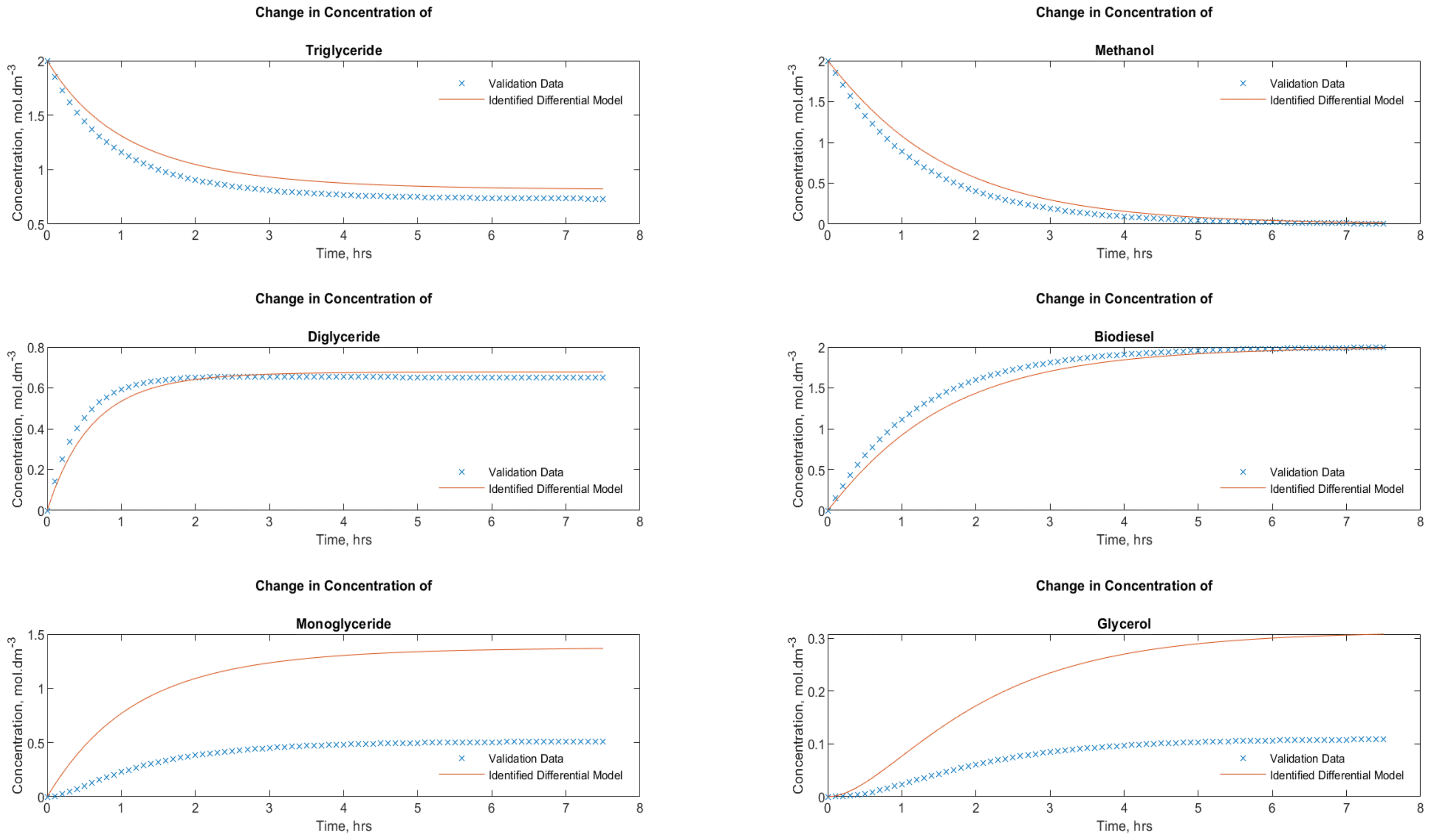


Figure 4.11 - Graphical representation of results using the Taylor series method.

The Taylor series approximation provided a good quality model with accurate numerical solutions. The numerical values of the model are like the forward Euler method, but slightly closer to the true values. This improvement is explained by the second order differential accounting for some of the error in the differential that could not be considered by the forward Euler method.

4.5.3 Differentiated Spline

The differentiated spline is a generated partial polynomial of the data provided. This is analytically differentiated using Matlab's *fnder* function to generate a differentiated partial polynomial which can be evaluated at each time point. The approximation of the differential is shown in Figure 4.12, with the identified models shown in Table 4-9 and the concentration profile of the model is displayed in Figure 4.13.

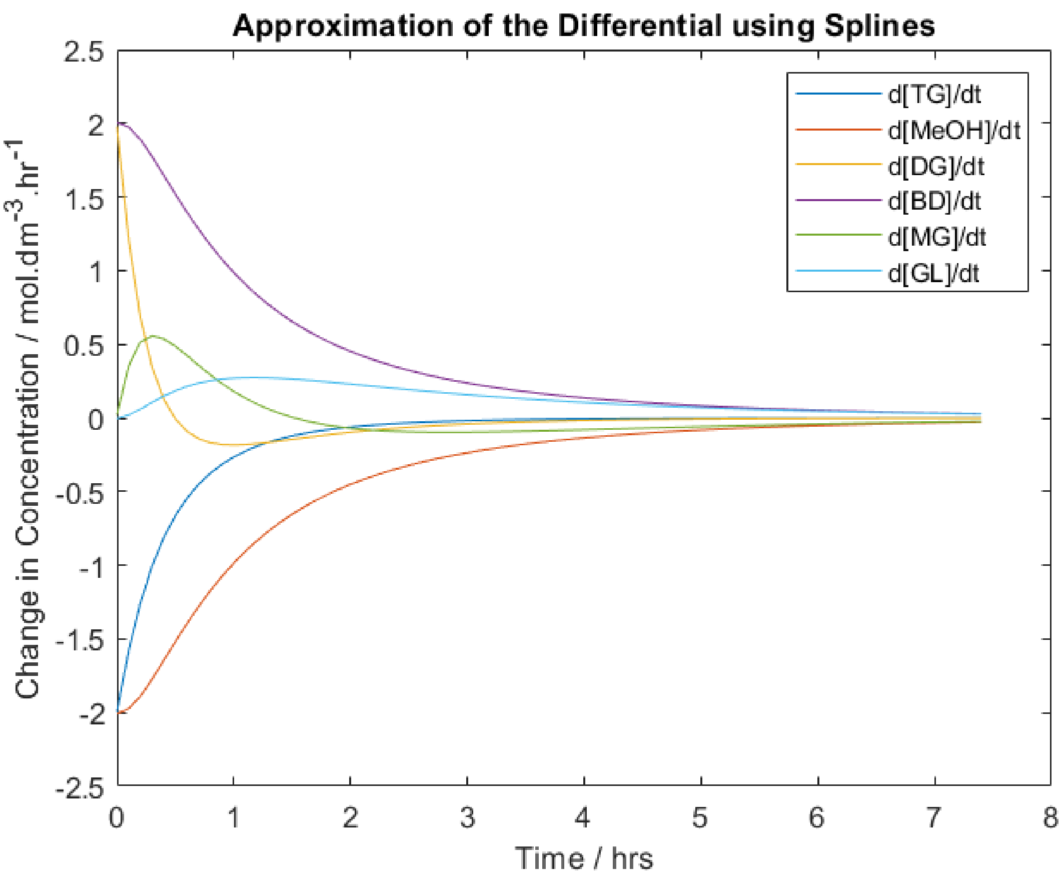


Figure 4.12 -The approximation of the differential using Differentiated Spline approach.

Table 4-9 - The Identified model found using the differentiated Spline approach.

	$[TG].[MeOH]$		$[DG].[MeOH]$		$[MG].[MeOH]$	
	Generated Model	Differential Approach	Generated Model	Differential Approach	Generated Model	Differential Approach
$\frac{d[TG]}{dt}$	-0.4	-0.399	0	0	0	0
$\frac{d[MeOH]}{dt}$	-0.4	-0.399	-0.5	-0.396	-0.2	-0.132
$\frac{d[DG]}{dt}$	0.4	0.399	-0.5	-0.396	0	0
$\frac{d[BD]}{dt}$	0.4	0.399	0.5	0.396	0.2	0.132
$\frac{d[MG]}{dt}$	0	0	0.5	0.396	-0.2	-0.132
$\frac{d[GL]}{dt}$	0	0	0	0	0.2	0.132

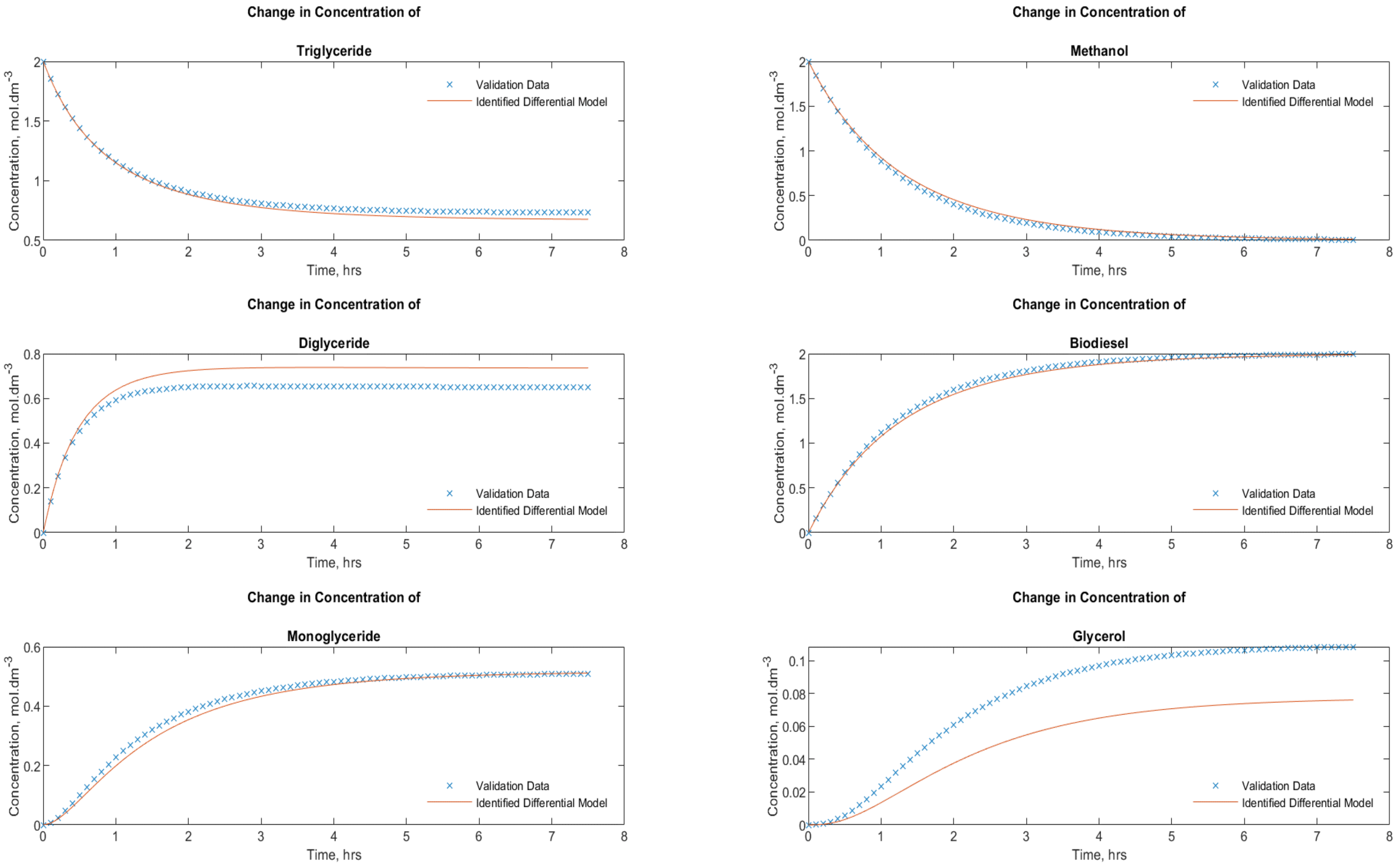


Figure 4.13 - The concentration profile of the model identified using the differentiated Spline approach compared to the validation data set.

The identified model with the differentiated spline method is the most accurate model of the differentiated methods, this can be seen in Figure 4.13 with the model having the smallest error between the validation and the model of the four differential approximations considered.

4.5.4 Differential approximation conclusions

Splining has been shown to provide the best models identified. The Backward Euler method generated the worst identified models. Of the remaining models, the Forward Euler, Central Euler, and Taylor series are all dependent upon the Forward Euler method. Therefore, the Differentiated spline approach, and the Forward Euler method will both be considered when noise is present in the measurements.

4.6 Integral approximation techniques

4.6.1 Trapezium rule

This method has been shown to be effective already in Figure 4.2 and Table 4-4. The trapezium rule uses two measurement points to estimate the area under a curve with a trapezium. The approximation of the integral is shown in Figure 4.14, the identified model is shown in Table 4-10 and the concentration profile of the identified model is in Figure 4.14.
The trapezium rule can be calculated from Equation 4-6, where a and b, refers to the limits of the integration.

$$\int f.dx \cong \frac{\Delta x}{2}\sum_{i=1}^{n}\left(f(x_{i-1}) + f(x_i)\right) \qquad \text{Equation 4-6}$$

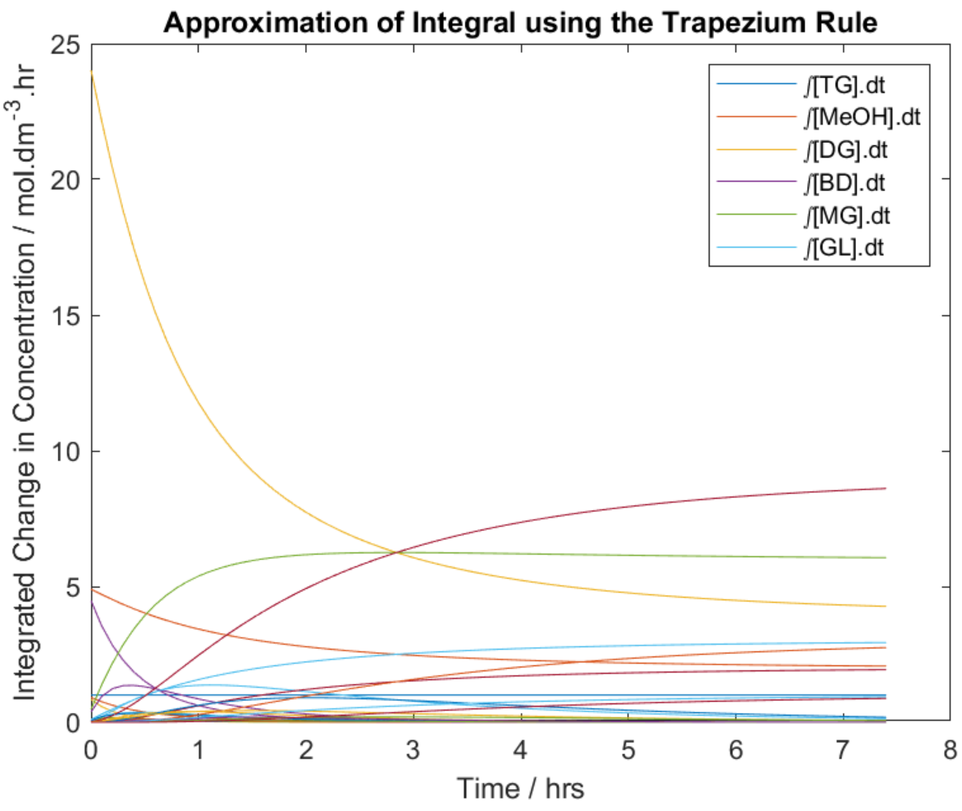


Figure 4.14 - Approximation of the integral term when using the trapezium rule.

Table 4-10 - The model identified when using the trapezium rule to approximate the integral

	$[TG].[MeOH]$		$[DG].[MeOH]$		$[MG].[MeOH]$	
	Generated Model	Differential Approach	Generated Model	Differential Approach	Generated Model	Differential Approach
$\frac{d[TG]}{dt}$	-0.4	-0.397	0	0	0	0
$\frac{d[MeOH]}{dt}$	-0.4	-0.397	-0.5	-0.394	-0.2	-0.132
$\frac{d[DG]}{dt}$	0.4	0.397	-0.5	-0.394	0	0
$\frac{d[BD]}{dt}$	0.4	0.397	0.5	0.394	0.2	0.132
$\frac{d[MG]}{dt}$	0	0	0.5	0.394	-0.2	-0.132
$\frac{d[GL]}{dt}$	0	0	0	0	0.2	0.132

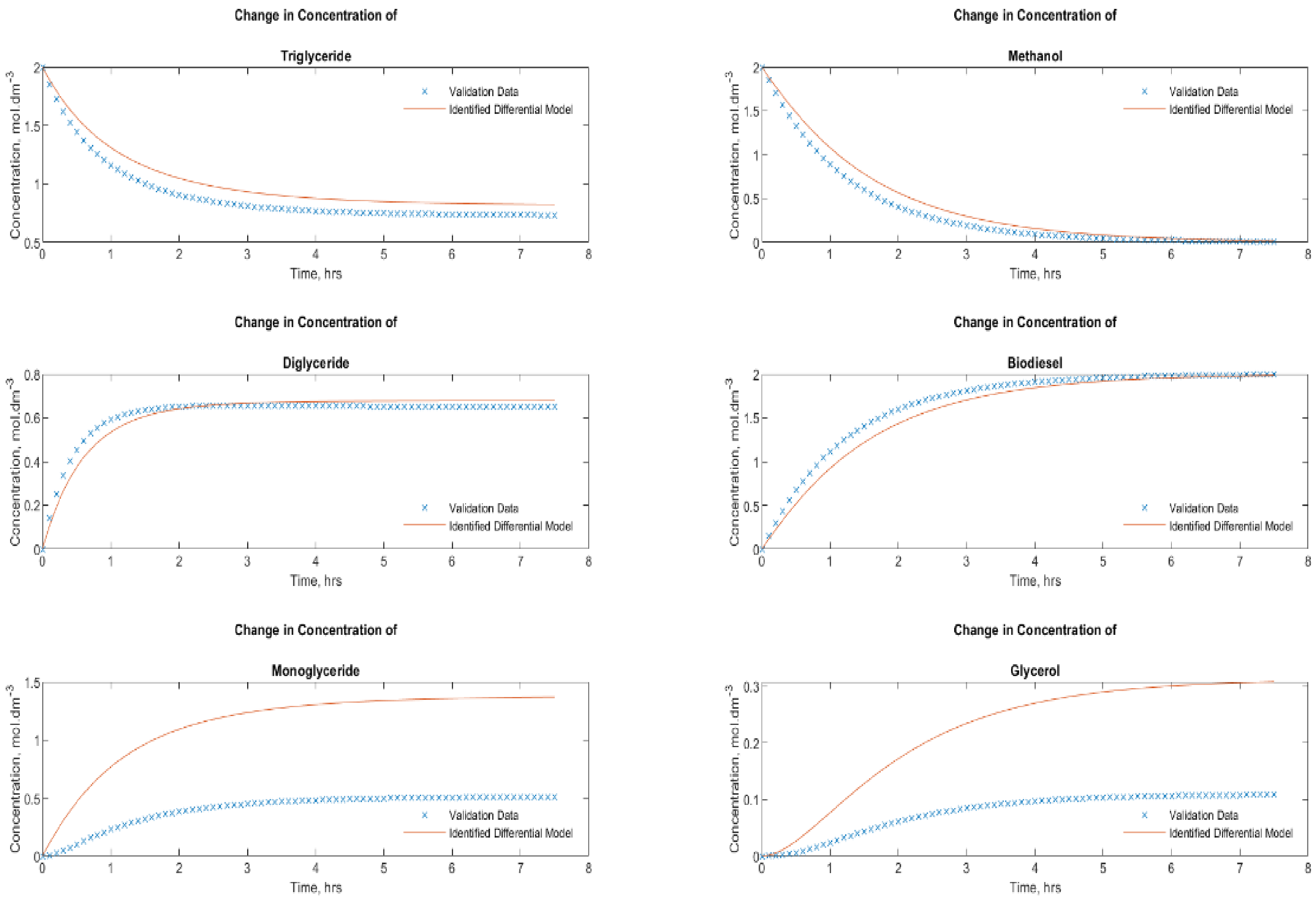


Figure 4.15 - Concentration profile of the model generated using the trapezium rule to approximate the integral.

The trapezium rule generates an accurate model – the only method that has a superior model so far is the differentiated spline.

4.6.2 Rectangle rule (Riemann Sum)

The Rectangle rule uses a similar principle to the trapezium rule but generates a rectangle to approximate the area underneath the curve. This is generally considered to be an inferior integral approximation than the trapezium rule. The approximation of the integral is shown in Figure 4.16, the identified model is shown in

Table 4-11, and the concentration profile of the model is shown in Figure 4.17. The Reiman Sum can be calculated using Equation 4-7 (Chapra, 2015):

$$\int f(x).dx = \sum_{i=1}^{n} f(x_i) * \Delta x \qquad \text{Equation 4-7}$$

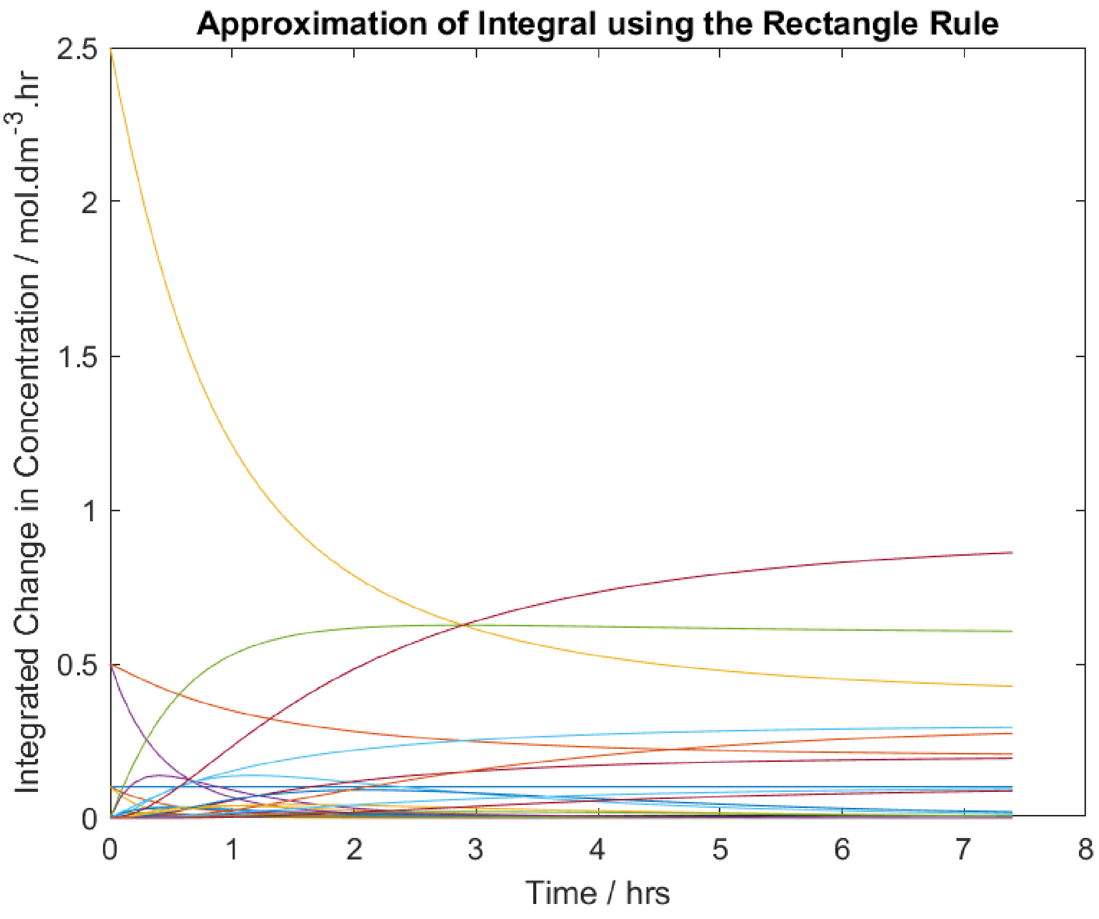


Figure 4.16 - Approximation of the Integral using the rectangle rule.

Table 4-11 - The model identified when using the rectangle rule to approximate the integral.

	$[TG].[MeOH]$		$[DG].[MeOH]$		$[MG].[MeOH]$	
	Generated Model	Differential Approach	Generated Model	Differential Approach	Generated Model	Differential Approach
$\frac{d[TG]}{dt}$	-0.4	0	0	0	0	0
$\frac{d[MeOH]}{dt}$	-0.4	0	-0.5	-0.4995	-0.2	0
$\frac{d[DG]}{dt}$	0.4	0	-0.5	0	0	0
$\frac{d[BD]}{dt}$	0.4	0	0.5	0.4995	0.2	0
$\frac{d[MG]}{dt}$	0	0	0.5	0	-0.2	0
$\frac{d[GL]}{dt}$	0	0	0	0	0.2	0

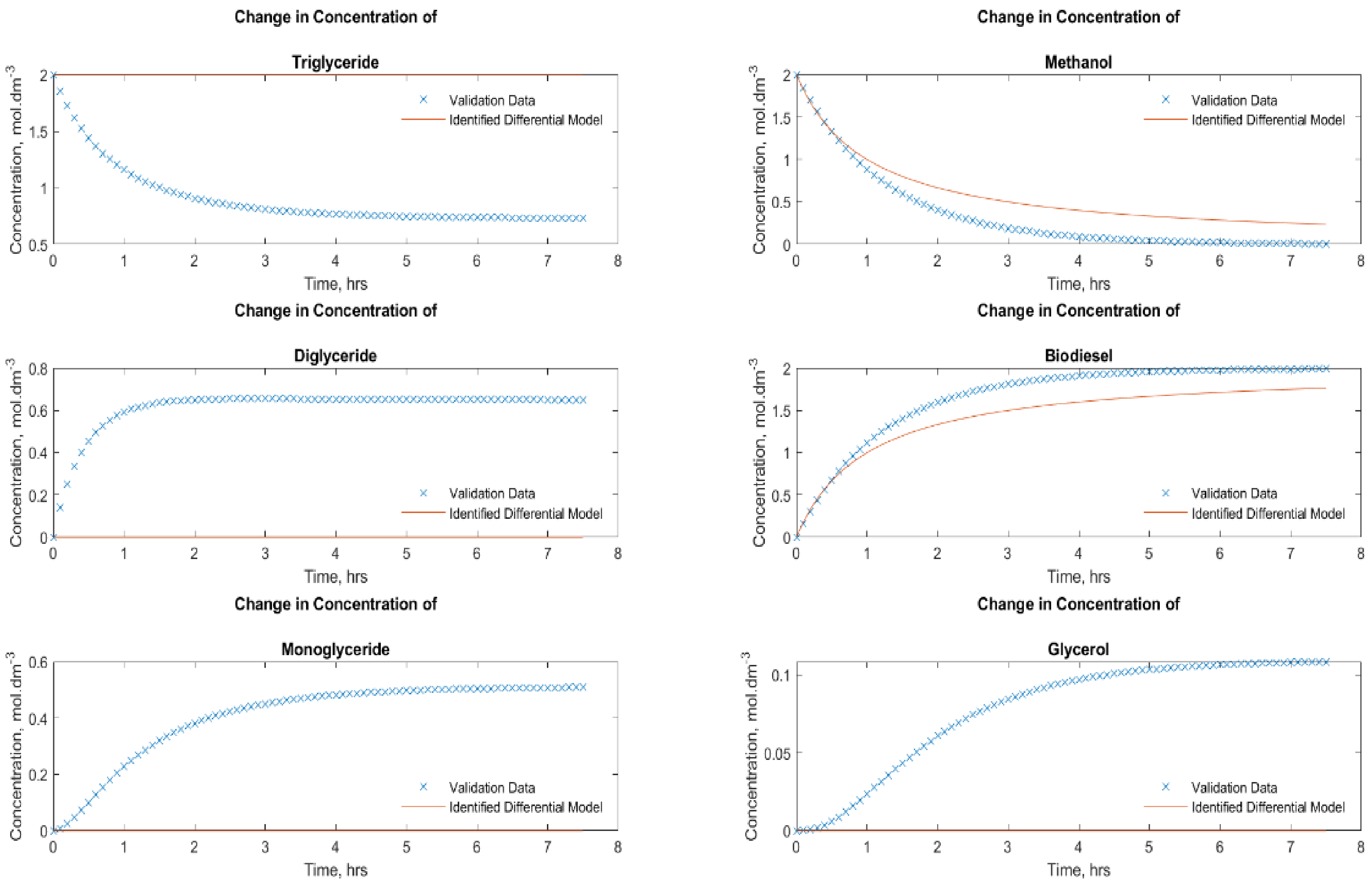


Figure 4.17 - The Concentration profile of the model identified with the Rectangle rule approximating the integral.

The rectangle rule fails to identify a model. The approximation of the integral Figure 4.16 appears to be like other approximations, however the approximations are sufficiently different for it to become impossible to identify a model.

4.6.3 Simpsons rule

There are several variations of Simpsons rule, for this approximation, Simpson's 3/8ths rule was applied as it is an advancement on the trapezium rule being a cubic version of the Newton-Coates formulae for approximating an integral. The approximation of the integral is shown in Figure 4.18,and the identified model is shown in Table 4-12. The two main Simpson's Rules, Simpson's 1/3 rule (Equation 4-8a) and Simpson's 3/8ths rule (Equation 4-8b) are shown below.

$$A = \frac{\Delta x}{3}[f(x_0) + f(x_n) + 4(f(x_1) + f(x_3) + \cdots) + 2(f(x_2 + f(x_4) + \cdots)] \qquad \text{Equation 4-8a}$$

$$A = \frac{3\Delta x}{8}[f(x_0) + f(x_n) + 3(f(x_1) + f(x_2) + f(x_4) + f(x_5) + \cdots) + (f(x_3) + f(x_6) + \cdots)] \qquad \text{Equation 4-8b}$$

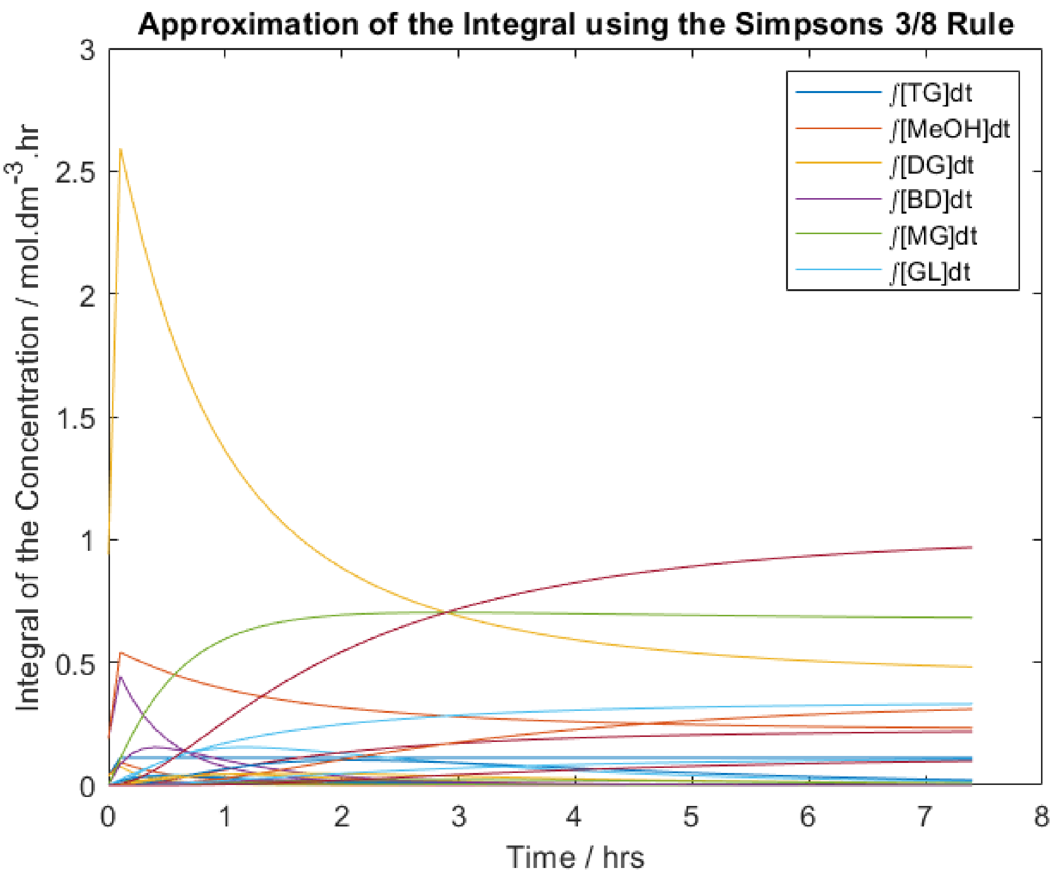


Figure 4.18 Approximation of the integral using Simpson's 3/8ths rule.

Table 4-12 - The model generated using the Simpson's 3/8ths rule to approximate the integral.

	$[TG].[MeOH]$		$[DG].[MeOH]$		$[MG].[MeOH]$	
	Generated Model	Integral Approach	Generated Model	Integral Approach	Generated Model	Integral Approach
$\frac{d[TG]}{dt}$	-0.4	0	0	0	0	0
$\frac{d[MeOH]}{dt}$	-0.4	0	-0.5	-0.4995	-0.2	0
$\frac{d[DG]}{dt}$	0.4	0	-0.5	0	0	0
$\frac{d[BD]}{dt}$	0.4	0	0.5	0.4995	0.2	0
$\frac{d[MG]}{dt}$	0	0	0.5	0	-0.2	0
$\frac{d[GL]}{dt}$	0	0	0	0	0.2	0

The approximation of the integral fails as the form of the Simpson's 3/8ths rule requires a minimum of three datapoints to generate a cubic approximation of the datapoints. This results in the first measurement being unable to generate a value for the approximation of the integral. The first datapoint has been shown in other approximation measurements to be of great importance in the identification of the model. This has resulted in the algorithm being unable to identify a kinetic model to describe the system. Approximation of the integral by the Simpson's rule cannot be considered any further.

4.6.4 Integrating Splines

As with differentiating the splines, piecewise polynomial splines are identified from the data, and can be analytically integrated by Matlab using the *fnint* function. The approximation of the integral is shown in Figure 4.19, the identified model is shown in Table 4-13, and the concentration profile of the model is displayed in Figure 4.20.

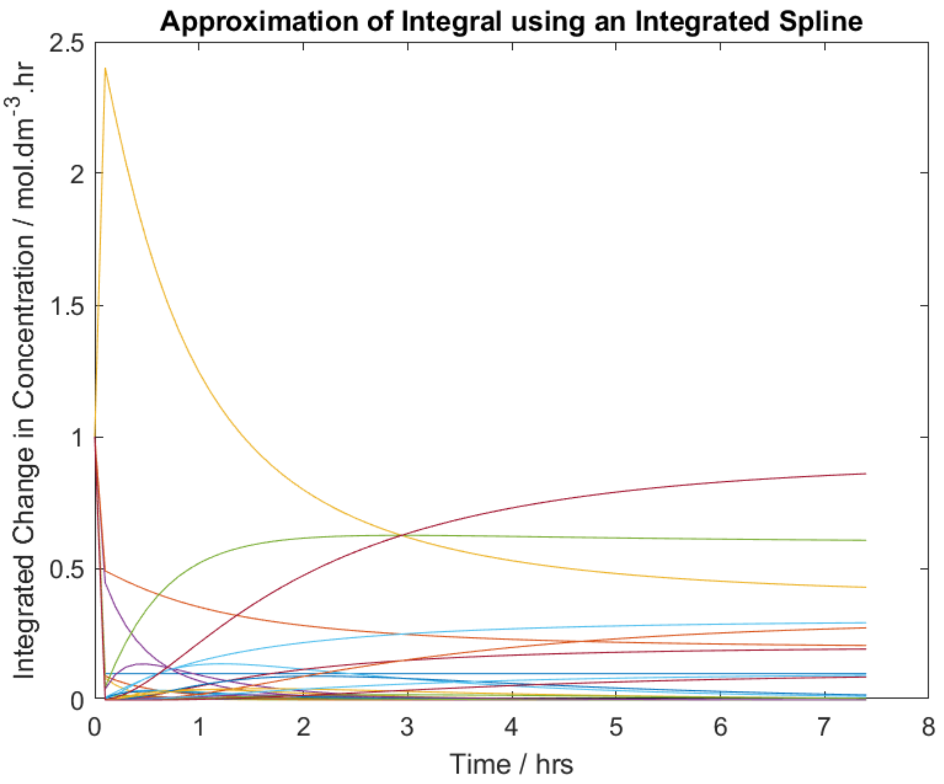


Figure 4.19 - Approximation of the Integral by analytically integrating a spline-generated piecewise polynomial.

Table 4-13 - Model Identified using the integrated spline approach.

	$[TG].[MeOH]$		$[DG].[MeOH]$		$[MG].[MeOH]$	
	Generated Model	Integral Approach	Generated Model	Integral Approach	Generated Model	Integral Approach
$\frac{d[TG]}{dt}$	-0.4	-0.573	0	0	0	0
$\frac{d[MeOH]}{dt}$	-0.4	-0.573	-0.5	-0.712	-0.2	-0.165
$\frac{d[DG]}{dt}$	0.4	0	-0.5	-0.712	0	0
$\frac{d[BD]}{dt}$	0.4	0.573	0.5	0.712	0.2	0.165
$\frac{d[MG]}{dt}$	0	0	0.5	0.712	-0.2	0
$\frac{d[GL]}{dt}$	0	0	0	0	0.2	0

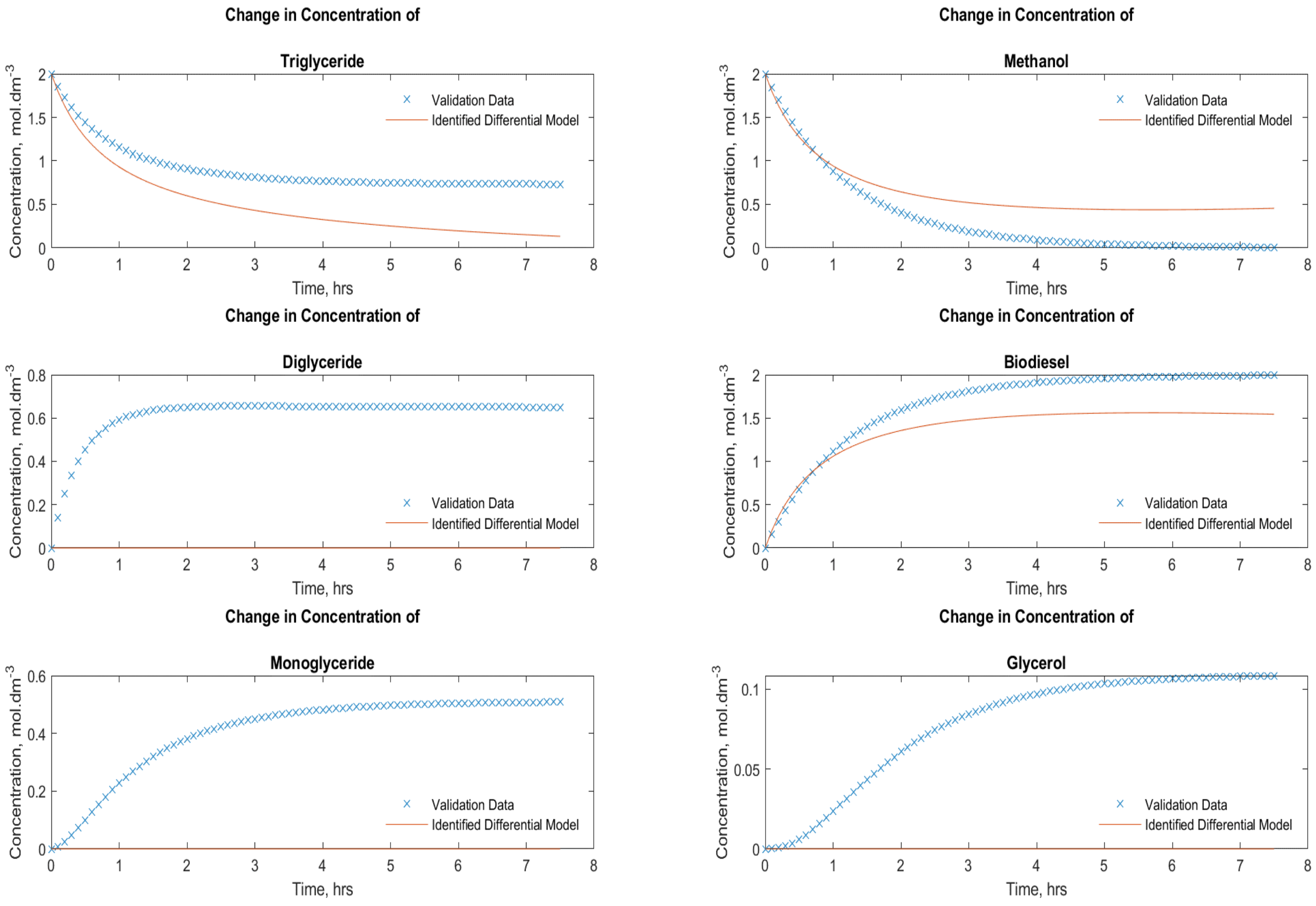


Figure 4.20 - The concentration profile of the model identified using the integrated spline approximation of the integral.

The identified approximation of the integral through analytically integrated splines (Figure 4.19) has similar problems to several of the differential approximation methods. The first datapoint cannot find the correct value, instead all values appear to have an integral value of one at time zero. Considering the errors in the identified integral, the model identified (Table 4-13) appears to be like the model used, however, components 3 and 6, (DG and GL) have not identified important terms. The importance of these terms is shown more obviously in Figure 4.19 where the importance of DG as an intermediate is shown. The partially correct identification of the change in concentration of DG resulted in there being no MG produced according to the model.

4.6.5 Integral approximation conclusions

The different integral approximation methods were less effective than the different methods of approximating the differential. The trapezium rule and the differentiated spline method gave the most accurate approximations and the best models. Since the splining method is an analytical technique, one of the numerical techniques will also be used as a comparison. The Forward Euler method is taken forward as the best numerical differentiation approach.

The most surprising result was that the integrated spline method gave such poor results when compared to the trapezium rule. It was anticipated that the integrated spline would provide superior approximations of the integral as with the differentiated spline. The importance of the approximation of the first datapoint for both integral and differential approaches appeared to be the limiting factor in many of the attempted approximation methods.

Therefore because of these results, only three approximations will be considered when noise is added to the system: The Forward Euler method, the differentiated spline, and the trapezium rule.

4.6.6 Introduction of noise to the system

To test the algorithm more thoroughly, the effect of noise of the algorithm's efficacy should be considered. Three approaches are being tested – The Forward Euler approach, a differentiated spline approach, and the trapezium rule approach. The system being modelled is once again the biodiesel production reaction network with 2% artificially added gaussian noise and continuing to use 75 measurements of the concentrations. The addition of noise also includes an element of randomness to the solutions found by the algorithm. Therefore, to produce statistically significant result there should be a minimum of 30 repeats. The goal is to show that when noise is present, the algorithm can still identify accurate models with precision with all three of the approaches being considered.

4.6.7 Differential approach with noise – Forward Euler

The thirty identified models at 2% gaussian noise with seventy-five measurements are shown in Figure 4.21. Of the 30 models that are identified, five of the models have all the reactions and terms in the correct positions. 26% of the models contain an error in the first reaction, 16% of the models contain an error in the second reaction and 83% of the models contain an error in the third reaction, and 10% of the models have identified incorrect, extra reactions that are known to be incorrect.

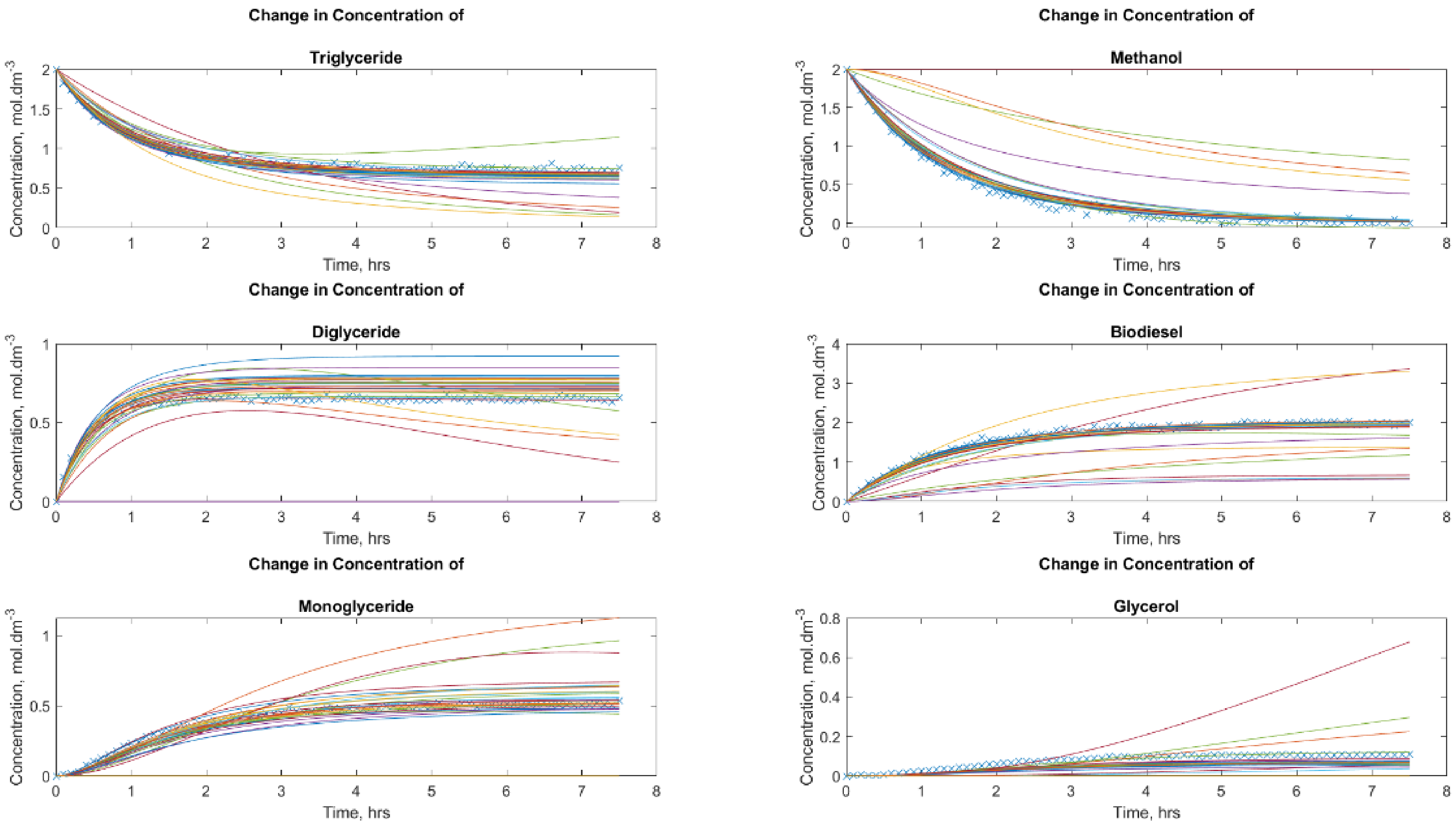


Figure 4.21 - 30 models Identified (lines) using the Forward Euler method and an example set of training data (crosses) where the training data has 75 measurements, and 2% artificial gaussian noise added.

Most of the models identified (Figure 4.21) appear to have similar shape to the data, however there are some models identified that appear to be incorrect, for example there is at least one model where the change in concentration of monoglyceride does not change. Unfortunately, none of the models can be instantly disregarded since the model of each component is strongly coupled with the other models- errors in the model of one component can have a significant impact on the model on other components.

Table 4-14 displays the percentage of the time that each possible rate constant has the correct sign. Note that many of the rate constants have a value of zero and are identified as such in all the thirty repeats. The change in concentration of TG has the correct kinetic structure identified in all the repeats. All the other kinetic descriptions of the change in concentration of the components have at least one error that has appeared.

Table 4-14 - Showing the percentage of the time each possible order is correctly identified (many of the possible orders are correctly identified as having a coefficient of zero) for the 30 models identified at 75 measurements and 2% noise. The errors have been highlighted to make them easier to identify.

	Change in concentration over time					
	$\frac{dTG}{dt}$	$\frac{dMeOH}{dt}$	$\frac{dDG}{dt}$	$\frac{dBD}{dt}$	$\frac{dMG}{dt}$	$\frac{dGL}{dt}$
[]	100	100	100	100	100	100
$'TG'$	100	100	100	100	100	100
$'TGTG'$	100	100	100	100	100	100
$'TGMeOH'$	100	90	90	86.66	100	100
$'TGDG'$	100	100	100	100	100	100
$'TGBD'$	100	100	100	100	100	100
$'TGMG'$	100	100	100	100	100	100
$'TGGL'$	100	100	100	100	100	100
$'MeOH'$	100	100	100	100	100	100
$'MeOHMeOH'$	100	96.66	100	96.66	100	100
$'MeOHDG'$	100	100	90	93.33	96.66	100
$'MeOHBD'$	100	100	100	100	100	100
$'MeOHMG'$	100	63.33	100	36.66	50	70
$'MeOHGL'$	100	100	100	100	100	100

$'DG'$	100	100	100	100	100	100
$'DGDG'$	100	100	96.66	100	96.66	100
$'DGBD'$	100	100	100	100	100	100
$'DGMG'$	100	100	100	100	100	100
$'DGGL'$	100	100	100	100	100	100
$'BD'$	100	100	100	100	100	100
$'BDBD'$	100	100	100	100	100	100
$'BDMG'$	100	100	100	100	100	100
$'BDGL'$	100	100	100	100	100	100
$'MG'$	100	96.66	100	100	96.66	100
$'MGMG'$	100	100	100	100	100	100
$'MGGL'$	100	100	100	100	100	100
$'GL'$	100	100	100	100	100	100
$'GLGL'$	100	100	100	100	100	100

There are seven positions where errors that appear only once (value of 96.66%) in the 30 repeats at this condition. Six of these seven appearances are of superfluous terms that should not be appearing within the model, and therefore are appearing in pairs – the $[MeOH].[MeOH]$ row is a reaction where $[MeOH]$ is being converted into $[BD]$. Table 4-15 shows an example of all the types of error that can occur. Reaction one, (column $[TG].[MeOH]$) has two terms in the correct places, and two terms missing. The identified superfluous term ($[MeOH]^2$) has a relatively small coefficient, therefore having a smaller impact on the overall model and there are two identified reactions that are incomplete. These false positives partially explain the reaction network, however the mechanism it identifies is not accurate.

All the other errors in the observed model occur when a value was expected to be in a position and was not present. The most common error is the change in biodiesel, associated with reaction 3 (the reaction associated with $[MeOH].[MG]$). This can be explained in two parts:

1. The rate constant of this reaction is the smallest of the three in this reaction network, and therefore has the smallest impact of the accuracy of the model identified.
2. It is the third reaction in a sequence, therefore by the time the concentration of MG is not high enough for the reaction to take place at an easily observable rate.

Table 4-15 – An example of one of the identified models using 75 measurements and 2% noise. There are several errors that occur in this model that can be compared to the generated model also shown.

	$[TG].[MeOH]$		$[MeOH]^2$		$[MeOH].[DG]$		$[MeOH].[MG]$	
	Generated Model	Identified Model	Generated Model	Identified Model	Generated Model	Identified Model	Generated Model	Identified Model
$\boldsymbol{d[TG]/dt}$	-0.4	-0.17	0	0	0	0	0	0
$\boldsymbol{d[MeOH]/dt}$	-0.4	0	0	-0.08	-0.5	-0.31	-0.2	0
$\boldsymbol{d[DG]/dt}$	0.4	0.17	0	0	-0.5	-0.31	0	0
$\boldsymbol{d[BD]/dt}$	0.4	0	0	0.08	0.5	0.31	0.2	0
$\boldsymbol{d[MG]/dt}$	0	0	0	0	0.5	0.31	-0.2	-0.06
$\boldsymbol{d[GL]/dt}$	0	0	0	0	0	0	0.2	0.06

For the other reactions (associated with $[TG].[MeOH]$ and $[MeOH].[DG]$) the lowest chance of the position being correct is 86.66% although it is not uncommon for there to be one of the errors in each identified model set, most of the identified model is correct. Table 4-14 shows that each of the positions have the correct sign, not the magnitude of the coefficient, the average value of the coefficient is shown in .

Table 4-16. .

Table 4-16 is a transposed version of the model that has been shown typically throughout this work to two decimal places. It shows that the average value of the coefficients tends to be underestimated. This could be partially due to the occurrences of missing terms that would lower the average value of the coefficient in that position, this is shown through the row associated with reaction 1 ($[[TG].[MeOH]$) where Table 4-14 showed that the coefficient associated with $\frac{d[TG]}{dt}$ was correct 100% percent of the time, and the other values were a lower percentage of being correct, and the average magnitude of the coefficients decreased as the percentage success of finding a value of the coefficient.

Table 4-16 - The average coefficient value of the thirty identified models with 2% noise, and 75 measurements when using the forward Euler method to identify models.

	Change in concentration over time					
	$\frac{dTG}{dt}$	$\frac{dMeOH}{dt}$	$\frac{dDG}{dt}$	$\frac{dBD}{dt}$	$\frac{dMG}{dt}$	$\frac{dGL}{dt}$
[]	0	0	0	0	0	0
$'TG'$	0	0	0	0	0	0
$'TGTG'$	0	0	0	0	0	0
$'TGMeOH'$	-0.35	-0.33	0.31	0.31	0	0
$'TGDG'$	0	0	0	0	0	0
$'TGBD'$	0	0	0	0	0	0
$'TGMG'$	0	0	0	0	0	0
$'TGGL'$	0	0	0	0	0	0
$'MeOH'$	0	0	0	0	0	0
$'MeOHMeOH'$	0	0	0	0	0	0
$'MeOHDG'$	0	-0.34	-0.31	0.32	0.33	0
$'MeOHBD'$	0	0	0	0	0	0
$'MeOHMG'$	0	-0.07	0	0.05	-0.05	0.07
$'MeOHGL'$	0	0	0	0	0	0
$'DG'$	0	0	0	0	0	0
$'DGDG'$	0	0	0	0	0	0
$'DGBD'$	0	0	0	0	0	0
$'DGMG'$	0	0	0	0	0	0
$'DGGL'$	0	0	0	0	0	0
$'BD'$	0	0	0	0	0	0
$'BDBD'$	0	0	0	0	0	0
$'BDMG'$	0	0	0	0	0	0
$'BDGL'$	0	0	0	0	0	0
$'MG'$	0	0	0	0	0	0
$'MGMG'$	0	0	0	0	0	0
$'MGGL'$	0	0	0	0	0	0
$'GL'$	0	0	0	0	0	0

$'GLGL'$	0	0	0	0	0	0

Another observation of the average coefficient values is that the terms that were determined to be superfluous errors that appeared only once have an average coefficient value of zero (to two decimal places). This confirms the observation from **Error! Reference source not found.** that the coefficient values of the superfluous terms are typically of a small magnitude. Overall, this indicates that the Forward Euler method is still effective with 2% noise added to the system measurements.

4.6.8 Differential Approach with Noise – Splining Method

The Splining Method uses Matlab's partial polynomial splining method (Matlab's *spline* function) to generate a model of the data, this can then be analytically differentiated (Matlab's *fndr* function) and evaluated (Matlab's *ppval* function). Thirty repeats with different artificial noise were conducted and the identified models are shown in Figure 4.22.

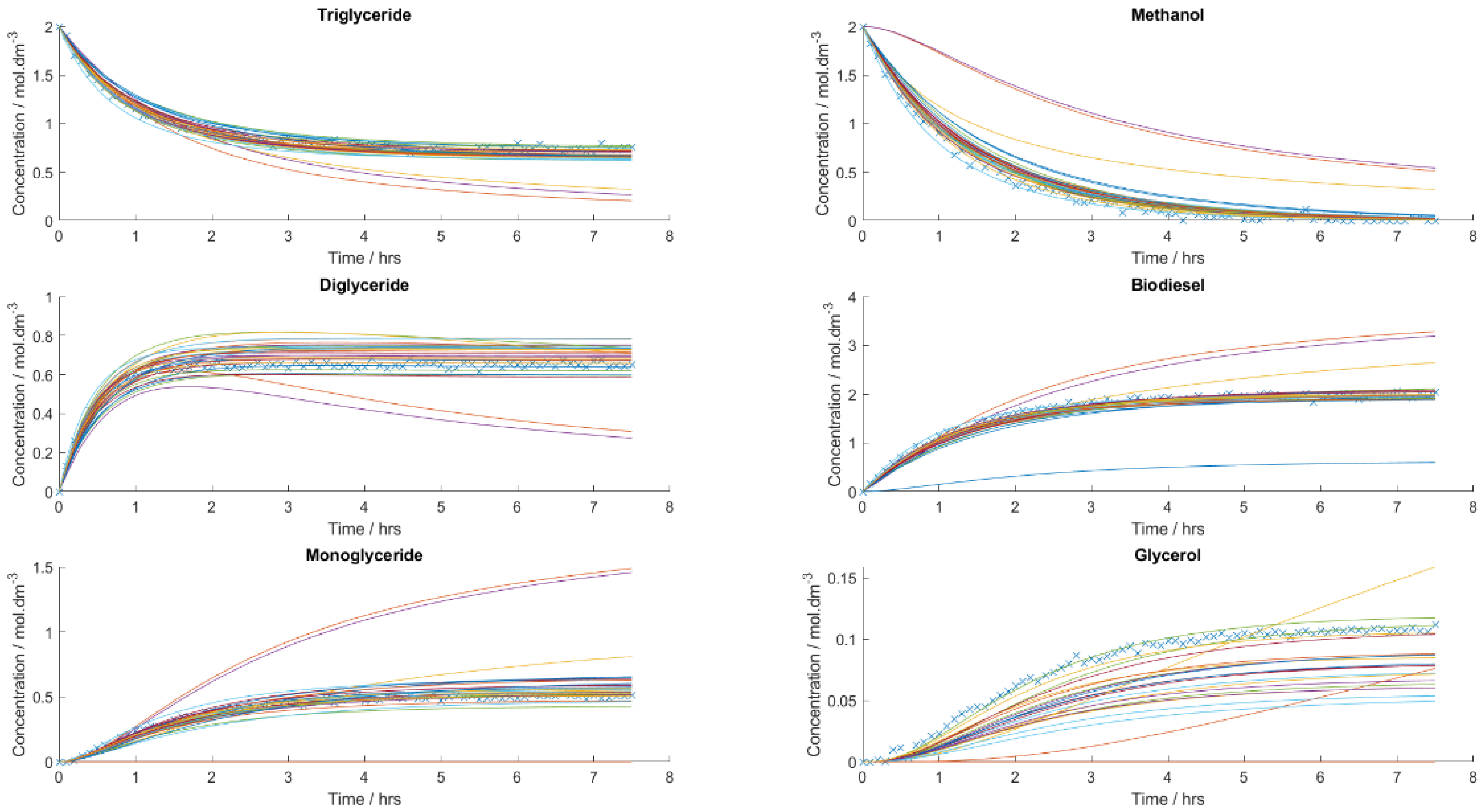


Figure 4.22 - Change in concentration of each of the components with 2% noise level and 75 measurements using differentiating Spline method. The crosses show an example of some of the measurements and the lines are the thirty generated models (30 datasets) when the algorithm splines the data and analytically differentiates the model identified.

Figure 4.22 shows that most of the models have accurate representations of the data, with a small number of the outlier models which do not follow the trend. As with the Forward Euler method, there does appear to be at least one model that does not provide a change in concentration for components MG and GL. The error in these components does not mean that the entire network model should be disregarded, however the model that describes the change in concentration of those individual components can be disregarded. Looking at the identified models, there is one instance of an error in the change in concentration of TG and DG (shown in Table 4-17 at as a single percentage at 96.67%), two instances of errors in the change in concentration of $MeOH$ (indicated by an error percentage of 96.67%) and several errors that can occur in the change in concentration of MG, BD and GL.

Table 4-17 - Percentage of times that each position is correct, with thirty repeats at 2% noise, with 75 measurements, the models were identified using the differentiated spline approach. Once again, the positions where errors have occurred in at least one instance have been highlighted.

	Change in concentration over time					
	$\frac{dTG}{dt}$	$\frac{dMeOH}{dt}$	$\frac{dDG}{dt}$	$\frac{dBD}{dt}$	$\frac{dMG}{dt}$	$\frac{dGL}{dt}$
[]	100	100	100	100	100	100
$'TG'$	100	100	100	100	100	100
$'TGTG'$	100	100	100	100	100	100
$'TGMeOH'$	100	100	100	100	100	100
$'TGDG'$	100	100	100	100	100	100
$'TGBD'$	100	100	100	100	100	100
$'TGMG'$	100	100	100	100	100	100
$'TGGL'$	100	100	100	100	100	100
$'MeOH'$	100	100	100	100	100	100
$'MeOHMeOH'$	100	100	100	100	100	100
$'MeOHDG'$	96.67	100	100	100	100	100
$'MeOHBD'$	100	100	100	100	100	100
$'MeOHMG'$	100	96.67	96.67	83.33	93.33	96.67
$'MeOHGL'$	100	100	100	100	100	100
$'DG'$	100	100	100	100	100	100

$'DGDG'$	100	100	100	100	100	100
$'DGBD'$	100	100	100	100	100	100
$'DGMG'$	100	100	100	100	100	100
$'DGGL'$	100	100	100	100	100	100
$'BD'$	100	100	100	100	100	100
$'BDBD'$	100	100	100	100	100	100
$'BDMG'$	100	100	100	100	100	100
$'BDGL'$	100	100	100	100	100	100
$'MG'$	100	100	100	100	100	100
$'MGMG'$	100	100	100	100	96.67	96.67
$'MGGL'$	100	100	100	100	100	100
$'GL'$	100	100	100	100	100	100
$'GLGL'$	100	100	100	100	100	100

The most likely errors are associated with the third reaction ($[MeOH].[MG]$), however the likelihood of these errors occurring appears to be lower than with the forward Euler method. Only one position has a percentage value lower than 90% (where $\frac{dBD}{dt}$ is affected by the third reaction $[MeOH].[MG]$). This is explained in the same way as it was for the Forward Euler method: the third reaction ($[MeOH].[MG]$) has the least influence on the change in concentration of BD due to the rate constant (the associated coefficient) being the smallest and the concentrations of MG and $MeOH$ being low throughout the reaction network.

The differentiation of splines method does however have no errors in the first reaction ($[TG].[MeOH]$) and one observed error in the second reaction ($[MeOH].[DG]$). The single error in the second reaction occurs where one model has an additional term identified associated with the change in concentration of TG.

Table 4-18 - The average coefficient value of the identified models using the differentiated spline approach at 2% noise and 75 measurements.

	Change in concentration over time					
	$\frac{dTG}{dt}$	$\frac{dMeOH}{dt}$	$\frac{dDG}{dt}$	$\frac{dBD}{dt}$	$\frac{dMG}{dt}$	$\frac{dGL}{dt}$
[]	0	0	0	0	0	0
$'TG'$	0	0	0	0	0	0
$'TGTG'$	0	0	0	0	0	0
$'TGMeOH'$	-0.39	-0.39	0.39	0.39	0	0
$'TGDG'$	0	0	0	0	0	0
$'TGBD'$	0	0	0	0	0	0
$'TGMG'$	0	0	0	0	0	0
$'TGGL'$	0	0	0	0	0	0
$'MeOH'$	0	0	0	0	0	0
$'MeOHMeOH'$	0	0	0	0	0	0
$'MeOHDG'$	0.01	-0.4	-0.4	0.4	0.4	0
$'MeOHBD'$	0	0	0	0	0	0
$'MeOHMG'$	0	-0.13	0	0.12	-0.12	0.13
$'MeOHGL'$	0	0	0	0	0	0
$'DG'$	0	0	0	0	0	0
$'DGDG'$	0	0	0	0	0	0
$'DGBD'$	0	0	0	0	0	0
$'DGMG'$	0	0	0	0	0	0
$'DGGL'$	0	0	0	0	0	0
$'BD'$	0	0	0	0	0	0
$'BDBD'$	0	0	0	0	0	0
$'BDMG'$	0	0	0	0	0	0
$'BDGL'$	0	0	0	0	0	0
$'MG'$	0	0	0	0	0	0
$'MGMG'$	0	0	0	0	0	0
$'MGGL'$	0	0	0	0	0	0
$'GL'$	0	0	0	0	0	0
$'GLGL'$	0	0	0	0	0	0

The average coefficient values (rate constant) for each position are shown in Table 4-18. The superfluous term in the second reaction (where $\frac{d[TG]}{dt}$ and $[MeOH].[DG]$ intersect) has a small value, shows that the superfluous terms tend towards zero when averaged over many repeats. The values of each reaction are significantly closer to the generated model's values, when compared against the values generated when using the Forward Euler method (.

Table 4-16). This further suggests that differentiating splines provides more accurate identified models than the Forward Euler method and the errors appear less frequently (under these conditions) suggesting the accuracy and reliability is superior using differentiated splines.

4.6.9 Integral approach with noise – Trapezium rule

The Integral approaches integrate the differential equation to provide an alternative method of identifying the kinetic structure and rate constants associated with the structure. The results are all displayed in the same form as with the differential approaches to keep the form of the results consistent. The change in molar concentration- ΔC, of each component can be calculated trivially at each time point, the integration of the structures is done using the trapezium rule. Table 4-19 show the percentage of the time that the models identified a value with the correct sign (and therefore an approximately correct value). Figure 4.23 shows the thirty models identified when 75 measurements are used with 2% artificial noise is added.

Table 4-19 - The percentage of the thirty repeats that each position has the correct sign when there at 75 measurements and 2% artificial noise added to the system.

	Change in concentration over time					
	$\frac{dTG}{dt}$	$\frac{dMeOH}{dt}$	$\frac{dDG}{dt}$	$\frac{dBD}{dt}$	$\frac{dMG}{dt}$	$\frac{dGL}{dt}$
[]	100	100	100	100	100	100
$'TG'$	100	100	100	100	100	100
$'TGTG'$	100	100	100	100	100	100
$'TGMeOH'$	96.67	46.67	96.67	56.67	100	100
$'TGDG'$	100	100	100	100	100	100
$'TGBD'$	100	100	100	100	100	100
$'TGMG'$	100	100	100	100	100	100
$'TGGL'$	100	100	100	100	100	100
$'MeOH'$	100	100	100	100	100	100
$'MeOHMeOH'$	100	96.67	100	96.67	100	100
$'MeOHDG'$	100	93.33	96.67	93.33	70	100
$'MeOHBD'$	100	100	100	100	100	100
$'MeOHMG'$	100	26.67	100	16.67	33.33	46.67
$'MeOHGL'$	100	100	100	100	100	100
$'DG'$	100	100	100	100	100	100
$'DGDG'$	96.67	100	100	100	96.67	100
$'DGBD'$	100	100	100	100	100	100
$'DGMG'$	100	100	100	100	100	100
$'DGGL'$	100	100	100	100	100	100
$'BD'$	100	100	100	100	100	100
$'BDBD'$	100	100	100	100	100	100
$'BDMG'$	100	100	100	100	100	100
$'BDGL'$	100	100	100	100	100	100
$'MG'$	100	100	100	100	100	100
$'MGMG'$	100	100	96.67	100	100	96.67
$'MGGL'$	100	100	100	100	100	100
$'GL'$	100	100	100	100	100	100
$'GLGL'$	100	100	100	100	100	100

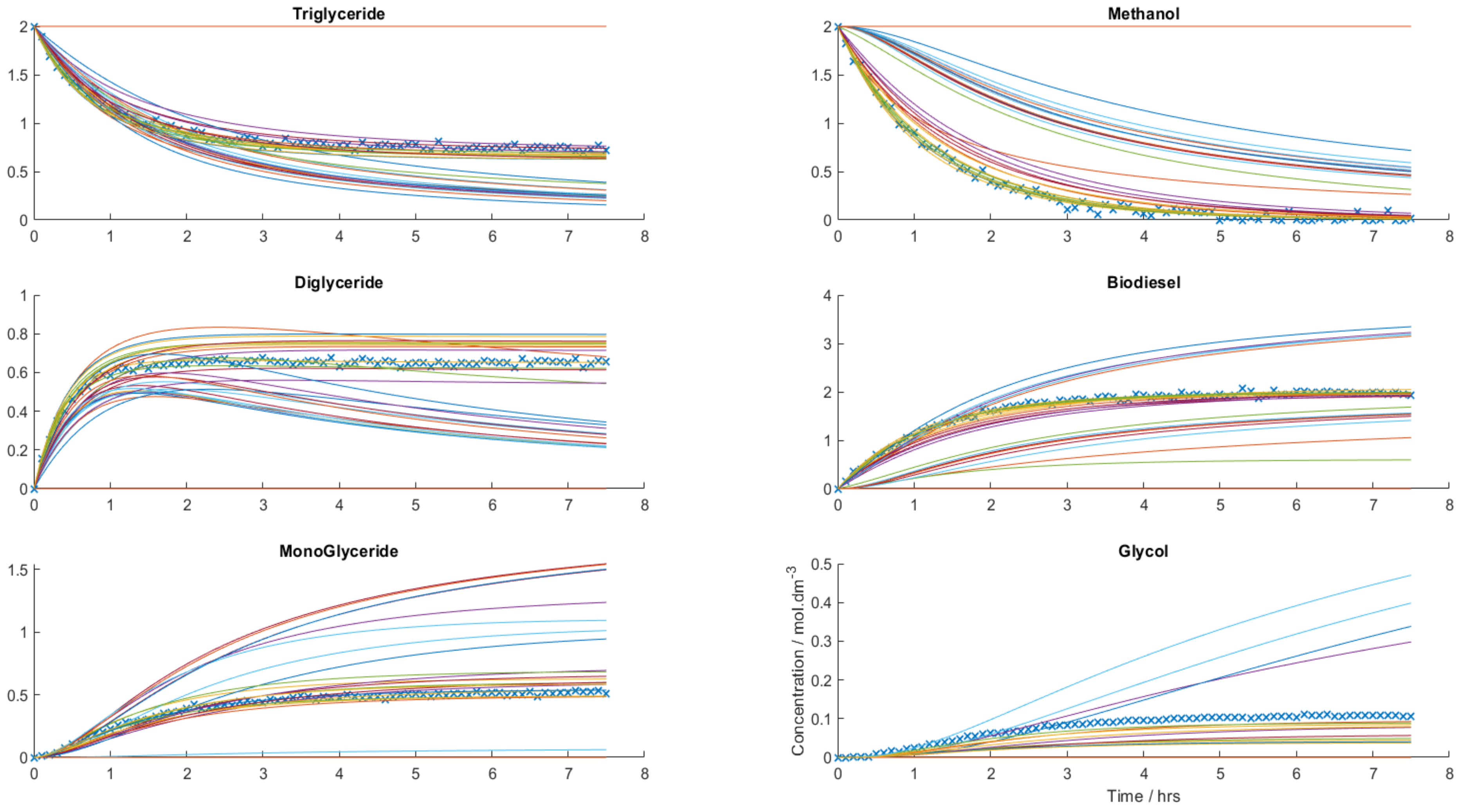


Figure 4.23 - Change in concentration of each of the components with 2% noise level and 75 measurements using Trapezium rule. The crosses show an example of some of the measurements and the lines are the thirty generated models (30 datasets) when the algorithm splines the data and analytically differentiates the model identified.

Table 4-20 - The average coefficient value identified over thirty repeats using 75 measurements at 2% noise.

	Change in concentration over time					
	$\frac{dTG}{dt}$	$\frac{dMeOH}{dt}$	$\frac{dDG}{dt}$	$\frac{dBD}{dt}$	$\frac{dMG}{dt}$	$\frac{dGL}{dt}$
[]	0	0	0	0	0	0
$'TG'$	0	0	0	0	0	0
$'TGTG'$	0	0	0	0	0	0
$'TGMeOH'$	-0.31	-0.18	0.31	0.19	0	0
$'TGDG'$	0	0	0	0	0	0
$'TGBD'$	0	0	0	0	0	0
$'TGMG'$	0	0	0	0	0	0
$'TGGL'$	0	0	0	0	0	0
$'MeOH'$	0	0	0	0	0	0
$'MeOHMeOH'$	0	0	0	0	0	0
$'MeOHDG'$	0	-0.4	-0.42	0.4	0.3	0
$'MeOHBD'$	0	0	0	0	0	0
$'MeOHMG'$	0	-0.03	0	0.02	-0.03	0.04
$'MeOHGL'$	0	0	0	0	0	0
$'DG'$	0	0	0	0	0	0
$'DGDG'$	0	0	0	0	0	0
$'DGBD'$	0	0	0	0	0	0
$'DGMG'$	0	0	0	0	0	0
$'DGGL'$	0	0	0	0	0	0
$'BD'$	0	0	0	0	0	0
$'BDBD'$	0	0	0	0	0	0
$'BDMG'$	0	0	0	0	0	0
$'BDGL'$	0	0	0	0	0	0
$'MG'$	0	0	0	0	0	0
$'MGMG'$	0	0	0	0	0	0
$'MGGL'$	0	0	0	0	0	0
$'GL'$	0	0	0	0	0	0
$'GLGL'$	0	0	0	0	0	0

The graphs of the models identified with the trapezium rule (Figure 4.23) appear to give a similar set of results when the two differential methods under these conditions (Figure 4.21 and Figure 4.22). There is at least one instance of no model being identified for the entire reaction network- shown by there being no change in concentration. Table 4-19 also suggests that there is a considerable number of errors that occur regularly within the identified models, including several superfluous terms that are incorrect – rows $[MeOH].[MeOH], [DG].[DG]$ and $[MG].[MG]$. The average coefficient terms (shown in Table 4-20) do have values in the correct positions. However, the average value for the rate constant in reactions 1 & 3 are smaller than expected. This shows the most common error that occurs at this condition – an unexpected value of zero, therefore lowering the average rate constant value.

The commonplace of these errors would suggest that the trapezium rule is not effective in the identification of kinetic parameters when noise is present, however noise was not the most influential variable when using the trapezium rule – the number of measurements used has a stronger influence on the efficacy of this method.

4.7 Effect of the number of measurements

When working with a real system, taking a measurement can be expensive – either in terms of physical and computational labour, or the measurements require specialised equipment with a monetary cost associated with it, therefore it would be ideal to minimise the number of measurements taken wherever possible. However, there is intuitively a minimum number of measurements required to provide enough information (within the data) to identify the reaction networks. It would therefore logically follow that there is an optimal number of measurements used to identify the reaction network depending on the amount of noise present in the measurements.

4.7.1 Forward Euler Method

An example of the effect of the number of measurements is shown in Figure 4.24. The immediate observations from Figure 4.24 are that at 1% measurement noise there is a minimum requirement of 10 measurements for there to be a possibility of identifying the models, although the chances of identifying models increases significantly when 20 measurements are used. When the number of measurements is less than or equal to twenty, there only model that can be identified is a null model with zero values in all positions. This is interpreted as there not being enough data provided for the algorithm. This identifies a flaw in using the Forward Euler method – and the other Euler variants, that cannot be compensated for: the space between the measurements is large, therefore causing the approximation of the differential being less accurate. This inaccuracy in the approximation of the differential results in the algorithm disqualifying the correct models from being possible solutions.

Increasing the number of measurements from 20 up to 75 appears to increase the precision of the coefficient values and therefore creating more accurate models. Whilst a more precise coefficient value is important, the precision should be weighed against the cost of collecting the required data.

The Forward Euler method uses the adjacent measurements to calculate the change in concentration, this makes the number of measurements a core variable to be considered. If the distance between measurements becomes too large, then the approximation of the differential becomes less accurate therefore making the approximation useless. Although not shown in Figure 4.24, it logically follows that when the distance between measurements is small (lots of measurements), then effect of the data corruption caused by noise will be more significant. In Section 4.8, the effect of both variables changing simultaneously will be examined.

4.7.2 Differentiation of Spline

Splining data does not have the same restrictions of the Euler approximations as it does not directly identify the differential from the data, instead there is an intermediate step of converting the data into a piecewise polynomial which in turn is differentiated analytically. Figure 4.25 shows the models identified using different numbers of measurements at 1% noise. As with the Forward Euler Method, there are no models identified once the number of measurements decreases below 10. However, when there are 10 measurements present the differentiated spline method has a higher chance of identifying a model- shown by more lines being present.

When there are 20 or more measurements used, the accuracy of the models is superior to the Forward Euler method at all scenarios. The accuracy of the identified models does not improve significantly from 20 measurements, up to 75 measurements. The effect of noise and the number of measurements used are invariably interconnected, these two variables are considered together in section 4.8.

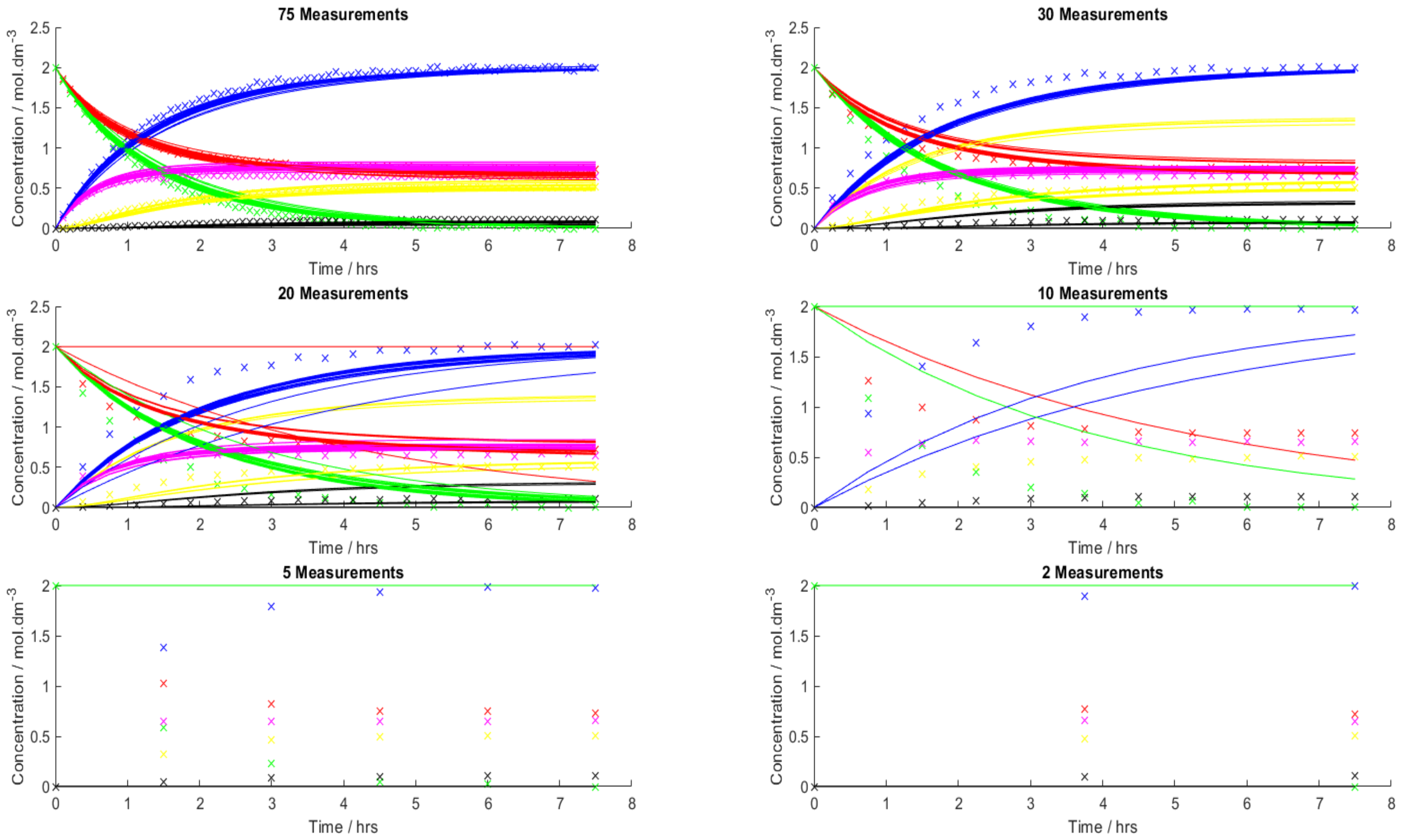


Figure 4.24 - Graphs showing the thirty identified models with the effect of the number of measurements on the quality of models identified using the Forward Euler Method with 1% artificial noise. Crosses are (example) measurements, the lines are the different models identfied. Red is Triglyceride (TG), Green is Methanol (MeOH), Magenta is Diglyceride (DG), Blue is Biodiesel (BD), Yellow is MonoGlyceride (MG), and Black is glycol (GL).

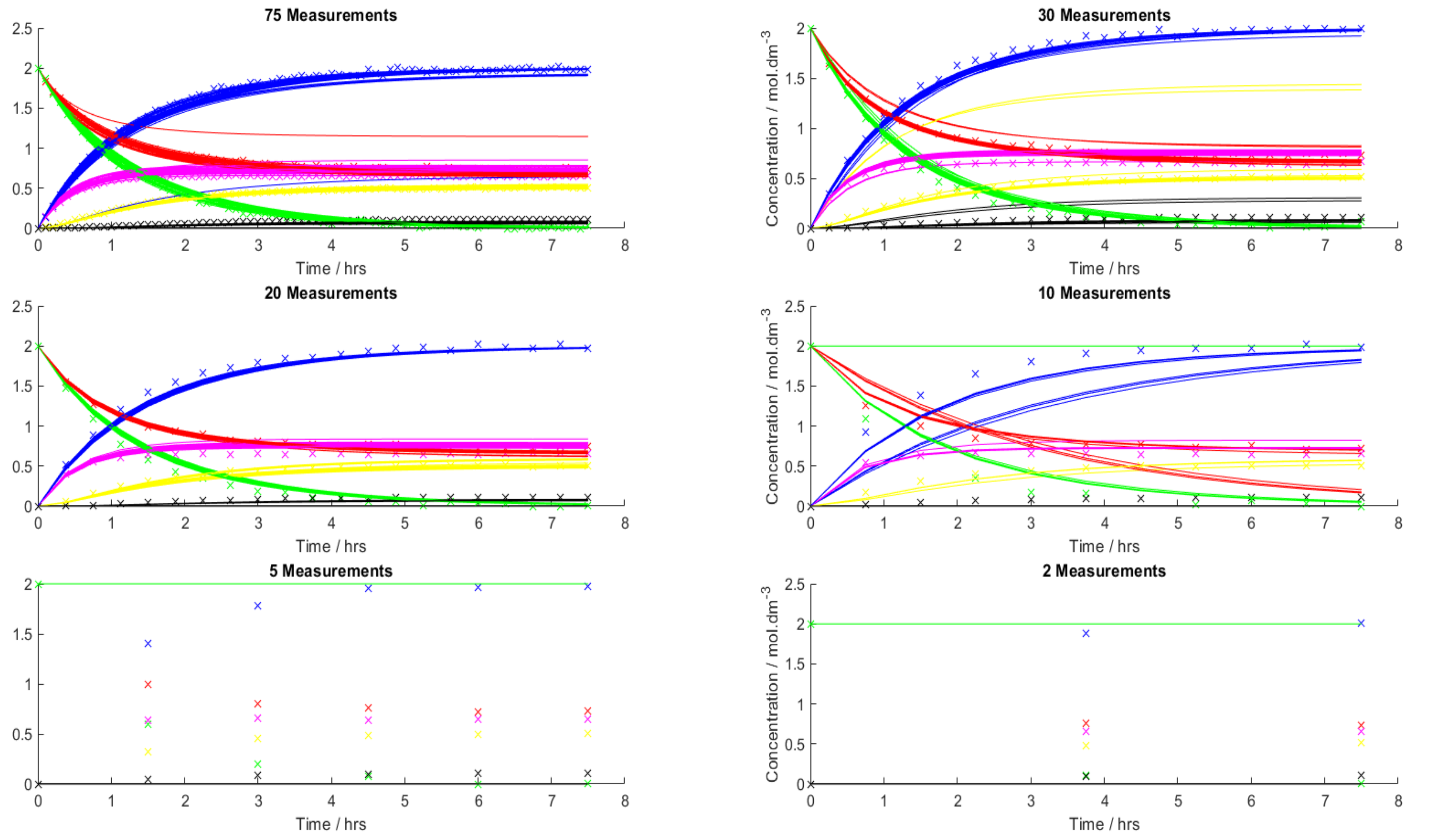


Figure 4.25 - Graphs showing the thirty identified models with the effect of the number of measurements on the quality of models identified using the Differentiating Spline method with 1% artificial noise. Crosses are (example) measurements, the lines are the different models identfied. Red is Triglyceride (TG), Green is Methanol (MeOH), Magenta is Diglyceride (DG), Blue is Biodiesel (BD), Yellow is MonoGlyceride (MG), and Black is glycol (GL).

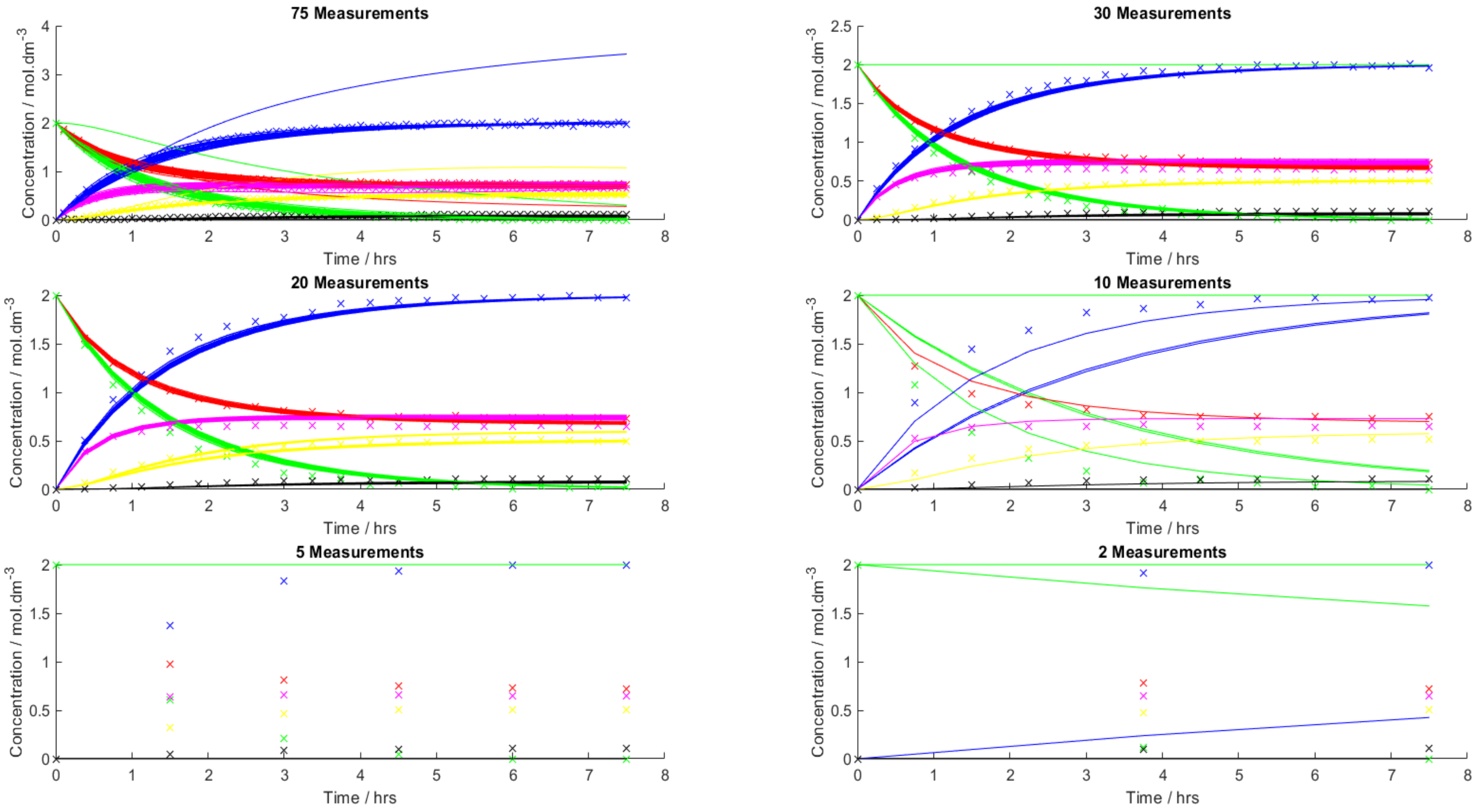


Figure 4.26 - Graphs showing the thirty identified models with the effect of the number of measurements on the quality of models identified using the Trapezium Rule with 1% artificial noise. Crosses are (example) measurements, the lines are the different models identfied. Red is Triglyceride (TG), Green is Methanol (MeOH), Magenta is Diglyceride (DG), Blue is Biodiesel (BD), Yellow is MonoGlyceride (MG), and Black is glycol (GL).

4.7.3 Trapezium rule

The effect of the number of measurements on the efficacy of the trapezium rule – integral approach, to the identification on models is shown in Figure 4.26. As with the other methods, when the number of measurements decreases below 10 measurements, there are no accurate models.

When 10 measurements are used there is a possibility of finding the reaction network, however there is also a reasonably chance of there being a model of zeros identified instead. Once the number of measurements increases to twenty or higher, the models appear to be an accurate depiction of the data with an exception observed at both 75 and 30 measurements. The models appear to have high precision as there is a smaller spread throughout the different models identified. This precision is explained through integrating the data rather than differentiating. The noise is the reason for the variation within the models at any one condition, this means that the integration of the data will reduce the impact of the noise on the algorithm.

The integration appears to be more stable than the Forward Euler method with respect to the number of measurements, however there is a point where the approximation of the integral becomes so inaccurate (due to large spaces between measurements) that the algorithm cannot find any model that is considered acceptable by the algorithm.

Of the three methods considered, the Forward Euler method appears to require the most data to give accurate models, and it has the most difficulty with fewer measurements. The Trapezium Rule has significantly more precision in the identified models and appears to transition from precise models to not finding models around the 10 measurements level. In comparison, the differentiated splines appear to have the highest reliability in the identification of models, but still cannot find models with less than 10 measurements. The analysis of this data, and the data from section 4.6.6 has shown that the number of measurements used, and the level of noise are coupled together with respect to the efficacy of each of the methods. The next section provides more analysis of the different conditions being considered and the efficacy of the models at each condition.

4.8 Comparing different conditions

To consider as many scenarios as possible a central composite design of the possible scenarios is used with thirty repeats at each condition to establish a degree of statistical confidence in the results. There are two factors that are being considered, the amount of noise present, and the number of measurements being used to identify the model. There are six levels of noise being considered:

- 0%
- 1%
- 2%
- 3%
- 5%

There are five levels of the number of measurements being used:

- 75 measurements,
- 30 measurements,
- 20 measurements,
- 10 measurements,
- 5 measurements and,
- 2 measurements.

These levels were determined using the central composite rule where there is a focus on the anticipated 'centre' of the data, with an extreme on either side of that centre. This uses considerably less computational time than a full factorial design which would have required considerably more levels for both variables being considered. The anticipated centre for the noise levels is 2%, with the extremes being 0% noise (the minimum amount of noise) and 5% noise – where the corruption makes it difficult to identify kinetic models using MILP. The results have been separated by the three different methodologies – Forward Euler, Differentiated Spline & the Trapezium Rule, into Table 4-21,

Table 4-22 & Table 4-23. The values shown in Table 4-21,

Table 4-22 and Table 4-23 are the average coefficient of the non-zero terms in the identified models, and the asterisk identifies where the method did not find the correct model for all of the repeats at that condition, it is worth noting that this method of displaying the results does not account for the likelihood of finding the value shown. There are some cases where there is a single instance of an error in the identification of a reaction and there are cases where there were a small number of models identified (lots of models with no active terms). For example – At 1% noise & 20 measurements, using the Forward Euler method, there are 2 errors that do not result in inaccurate results, and several errors that do not cause significant errors to the models shown in Figure 4.27.

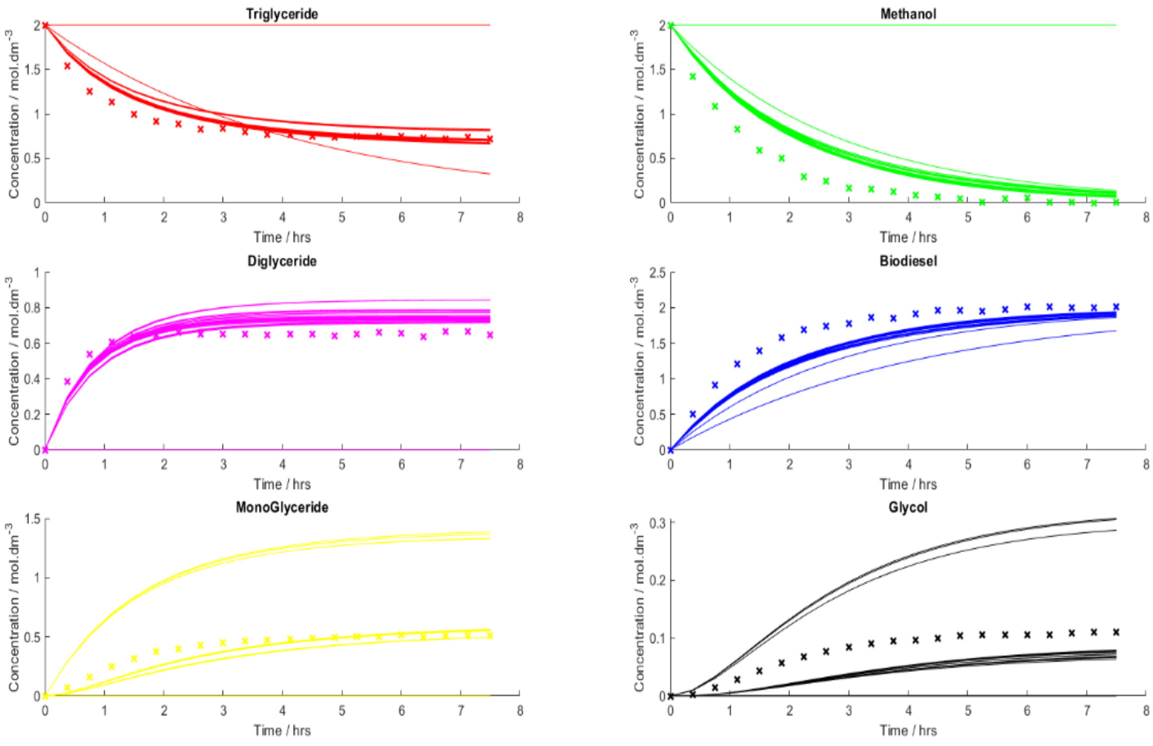


Figure 4.27 - plot of the 30 models identified at conditions 1%, 20 measurements, the crosses show an example set of measurement data and each line shows a model identified.

Figure 4.27 shows most of the models are following the trend of the measurement data, except for the errors described previously causing zero change in concentration. There are two error types that appear in the rest of the identified reaction networks: An error in the production of MG (reaction two not including MG) which occurs four times, & an error associated with reaction three. This error in reaction three occurred 24 times and did not include MG in the reaction. This explains why the monoglyceride in Figure 4.27 is overestimated in most of the identified models. The results are very consistent at this condition and the errors occur rarely (2 instances of the failure to identify a model), or the minor errors in the models identified appear to be consistent (24/30 models have the same error), this indicates that the methodology has an acceptable level of precision with some inaccuracy in the identified models at these conditions.

Table 4-21 - All conditions, Forward Euler, average coefficient of the non-zero terms identified over 30 models. An asterisk identifies where an error occurred at least once within the 30 repeats.

Forward Euler Method	**Noise**				
Measurements	**0%**	**1%**	**2%**	**3%**	**5%**
75	$r_1 = 0.25v[TG][MeOH]$ $r_2 = 0.27v[DG][MeOH]$ $r_3 = 0.12v[MG][MeOH]$	$r_1 = 0.37v[TG][MeOH]$ $r_2 = 0.35v[DG][MeOH]$ $r_3 = 0.13v[MG][MeOH] \ast$	$r_1 = 0.35v[TG][MeOH] \ast$ $r_2 = 0.34v[DG][MeOH] \ast$ $r_3 = 0.12v[MG][MeOH] \ast$	$r_1 = 0.29v[TG][MeOH] \ast$ $r_2 = 0.31v[DG][MeOH] \ast$ $r_3 = 0.09v[MG][MeOH] \ast$	$r_1 = 0.43v[TG][MeOH] \ast$ $r_2 = 0.24v[DG][MeOH]$*
30	$r_1 = 0.22v[TG][MeOH]$ $r_2 = 0.21v[DG][MeOH]$ $r_3 = 0.07v[MG][MeOH]$	$r_1 = 0.25v[TG][MeOH]$ $r_2 = 0.27v[DG][MeOH]$ $r_3 = 0.12v[MG][MeOH] \ast$	$r_1 = 0.29v[TG][MeOH]$ $r_2 = 0.26[DG][MeOH]$ $r_3 = 0.09[MG][MeOH] \ast$	$r_1 = 0.3v[TG][MeOH]$ $r_2 = 0.28[DG][MeOH]$ $r_3 = 0.10[MG][MeOH] \ast$	$r_1 = 0.29v[TG][MeOH]$* $r_2 = 0.26[DG][MeOH] \ast$ $r_3 = 0.09[MG][MeOH] \ast$
20	No Model Identified	$r_1 = 0.24v[TG][MeOH] \ast$ $r_2 = 0.23[DG][MeOH] \ast$ $r_3 = 0.08[MG][MeOH] \ast$	$r_1 = 0.25v[TG][MeOH] \ast$ $r_2 = 0.24[DG][MeOH] \ast$ $r_3 = 0.08[MG][MeOH] \ast$	$r_1 = 0.23v[TG][MeOH] \ast$ $r_2 = 0.23[DG][MeOH] \ast$ $r_3 = 0.09[MG][MeOH] \ast$	$r_1 = 0.24v[TG][MeOH] \ast$ $r_2 = 0.23[DG][MeOH] \ast$ $r_3 = 0.09[MG][MeOH] \ast$
10	No Model Identified	$r_1 = 0.10v[TG][MeOH] \ast$	No Model Identified	No Model Identified	No Model Identified
5	No Model Identified	No Model Identified	No Model Identified	No Model Identified	No Model Identified
2	No Model Identified	No Model Identified	No Model Identified	No Model Identified	No Model Identified

Table 4-22- All conditions, differentiated Spline method, average coefficient of the non-zero terms identified over 30 models. An asterisk identifies where an error occurred at least once within the 30 repeats.

Differentiated Spline	**Noise**				
Measurements	**0%**	**1%**	**2%**	**3%**	**5%**
75	$r_1 = 0.39v[TG][MeOH]$ $r_2 = 0.40v[DG][MeOH]$ $r_3 = 0.13v[MG][MeOH]$	$r_1 = 0.40v[TG][MeOH] \ast$ $r_2 = 0.40v[DG][MeOH] \ast$ $r_3 = 0.13v[MG][MeOH] \ast$	$r_1 = 0.39v[TG][MeOH]$ $r_2 = 0.40v[DG][MeOH]$ $r_3 = 0.13v[MG][MeOH] \ast$	$r_1 = 0.32v[TG][MeOH] \ast$ $r_2 = 0.39v[DG][MeOH] \ast$ $r_3 = 0.11v[MG][MeOH] \ast$	$r_1 = 0.30v[TG][MeOH] \ast$ $r_2 = 0.35v[DG][MeOH] \ast$
30	$r_1 = 0.39v[TG][MeOH]$ $r_2 = 0.37v[DG][MeOH]$ $r_3 = 0.13v[MG][MeOH]$	$r_1 = 0.38v[TG][MeOH]$ $r_2 = 0.37v[DG][MeOH] \ast$ $r_3 = 0.13v[MG][MeOH] \ast$	$r_1 = 0.39[TG][MeOH]$ $r_2 = 0.39[DG][MeOH]$ $r_3 = 0.13[MG][MeOH] \ast$	$r_1 = 0.37v[TG][MeOH] \ast$ $r_2 = 0.37[DG][MeOH] \ast$ $r_3 = 0.12[MG][MeOH] \ast$	$r_1 = 0.33[TG][MeOH] \ast$ $r_2 = 0.37[DG][MeOH] \ast$ $r_3 = 0.11[MG][MeOH] \ast$
20	$r_1 = 0.38v[TG][MeOH]$ $r_2 = 0.35v[DG][MeOH]$ $r_3 = 0.12v[MG][MeOH]$	$r_1 = 0.37[TG][MeOH]$ $r_2 = 0.35[DG][MeOH]$* $r_3 = 0.12[MG][MeOH] \ast$	$r_1 = 0.36v[TG][MeOH]$ $r_2 = 0.36[DG][MeOH]$ $r_3 = 0.12[MG][MeOH] \ast$	$r_1 = 0.35[TG][MeOH]$ $r_2 = 0.35[DG][MeOH] \ast$ $r_3 = 0.12[MG][MeOH] \ast$	$r_1 = 0.34[TG][MeOH] \ast$ $r_2 = 0.34[DG][MeOH] \ast$ $r_3 = 0.14[MG][MeOH] \ast$
10	$r_1 = 0.39v[TG][MeOH]$ $r_2 = 0.37v[DG][MeOH]$ $r_3 = 0.10v[MG][MeOH]$	$r_1 = 0.22v[TG][MeOH] \ast$ $r_2 = 0.28[DG][MeOH] \ast$ $r_3 = 0.09[MG][MeOH] \ast$	$r_1 = 0.26v[TG][MeOH] \ast$ $r_2 = 0.29[DG][MeOH] \ast$ $r_3 = 0.10[MG][MeOH] \ast$	$r_1 = 0.28v[TG][MeOH] \ast$ $r_2 = 0.28[DG][MeOH] \ast$ $r_3 = 0.10[MG][MeOH] \ast$	$r_1 = 0.26v[TG][MeOH] \ast$ $r_2 = 0.28[DG][MeOH] \ast$
5	No Model Identified	No Model Identified	$r_1 = 0.11v[TG][MeOH] \ast$ $r_3 = 0.05[MG][MeOH] \ast$	$r_1 = 0.11v[TG][MeOH] \ast$	No Model Identified
2	No Model Identified	No Model Identified	No Model Identified	No Model Identified	No Model Identified

Table 4-23 - All conditions, Trapezium rule, average coefficient of the non-zero terms identified over 30 models. An asterisk identifies where an error occurred at least once within the 30 repeats.

Trapezium rule	**Noise**				
Measurements	**0%**	**1%**	**2%**	**3%**	**5%**
75	$r_1 = 0.40v[TG][MeOH]$ $r_2 = 0.39v[DG][MeOH]$ $r_3 = 0.13v[MG][MeOH]$	$r_1 = 0.38[TG][MeOH] *$ $r_2 = 0.40v[DG][MeOH]$ $r_3 = 0.14v[MG][MeOH] *$	$r_1 = 0.32v[TG][MeOH] *$ $r_2 = 0.43v[DG][MeOH] *$ $r_3 = 0.11[MG][MeOH] *$	$r_1 = 0.24v[TG][MeOH] *$ $r_2 = 0.34v[DG][MeOH] *$ $r_3 = 0.10v[MG][MeOH] *$	$r_1 = 0.26v[TG][MeOH] *$ $r_2 = 0.40v[DG][MeOH] *$
30	$r_1 = 0.38v[TG][MeOH]$ $r_2 = 0.31v[DG][MeOH]$ $r_3 = 0.12v[MG][MeOH]$	$r_1 = 0.38v[TG][MeOH]$ $r_2 = 0.37v[DG][MeOH] *$ $r_3 = 0.13v[MG][MeOH] *$	$r_1 = 0.38[TG][MeOH]$ $r_2 = 0.37[DG][MeOH]$ $r_3 = 0.14[MG][MeOH] *$	$r_1 = 0.33v[TG][MeOH] *$ $r_2 = 0.32[DG][MeOH] *$ $r_3 = 0.12[MG][MeOH] *$	$r_1 = 0.32[TG][MeOH] *$ $r_2 = 0.39[DG][MeOH] *$ $r_3 = 0.14[MG][MeOH] *$
20	$r_1 = 0.36v[TG][MeOH]$ $r_2 = 0.35v[DG][MeOH]$ $r_3 = 0.12v[MG][MeOH]$	$r_1 = 0.35[TG][MeOH]$ $r_2 = 0.35[DG][MeOH]$ $r_3 = 0.12[MG][MeOH] *$	$r_1 = 0.35[TG][MeOH]$ $r_2 = 0.35[DG][MeOH]$ $r_3 = 0.11[MG][MeOH] *$	$r_1 = 0.32[TG][MeOH] *$ $r_2 = 0.30[DG][MeOH] *$ $r_3 = 0.10[MG][MeOH] *$	$r_1 = 0.32[TG][MeOH] *$ $r_2 = 0.33[DG][MeOH] *$ $r_3 = 0.10[MG][MeOH] *$
10	No Model Identified	$r_1 = 0.29v[TG][MeOH] *$ $r_2 = 0.30[DG][MeOH] *$ $r_3 = 0.10[MG][MeOH] *$	$r_1 = 0.29v[TG][MeOH] *$ $r_2 = 0.30[DG][MeOH] *$ $r_3 = 0.10[MG][MeOH] *$	$r_1 = 0.21v[TG][MeOH] *$ $r_2 = 0.22[DG][MeOH] *$ $r_3 = 0.06[MG][MeOH] *$	$r_1 = 0.22v[TG][MeOH] *$ $r_2 = 0.28[DG][MeOH] *$
5	No Model Identified	No Model Identified	No Model Identified	$r_1 = 0.11v[TG][MeOH] *$ $r_2 = 0.13v[DG][MeOH] *$	$r_1 = 0.12v[TG][MeOH] *$
2	No Model Identified	No Model Identified	No Model Identified	No Model Identified	No Model Identified

4.9 Discussion

The trends of the results are similar for all three methods, with the magnitude of the effects of noise and the number of measurements being slightly different for each approach.

Looking at the Forward Euler Method, Table 4-21 confirms the conclusion of section 4.7 that less than 10 measurements are not sufficient data for the algorithm to identify a model. Similarly, there is a point where the noise level becomes too high for an accurate model to be identified, shown by the condition where 75 measurements with 5% noise cannot identify the third reaction at all. It is also worth noting that this appears to be of a problem at different measurement levels indicating that the large noise has less of an influence when the number of measurements is reduced. This is explained as the noise being less impactful on the approximation of the differential when the distance between measurements is larger.

The Forward Euler method has been the least accurate of the three being considered however, Table 4-21 would indicate that this method has an acceptable range of conditions of operation. As discussed in the previous subsection, the values provided in Table 4-21 only counts the non-zero terms from the 30 repeats. Therefore, if only one of the repeats identifies the correct model structure and an accurate coefficient, and the others do not identify a model at all, then the average coefficient would appear to be accurate i.e. the precision of the methodology is not considered in Table 4-21.

As the amount of noise is increased, the precision of the models decreases, with more occurrences of missing terms and empty models being identified. By increasing the noise from zero to five percent at 75 measurements, the occurrence of errors in the identification of reaction 1 increases from 0 incidents (0% noise) to there being a single instance of an accurate approximation of reaction 1 at 5% noise. The trend follows a similar pattern for all the other two reactions, with reaction three being the most affected to the point of failing to identify any models once the noise increases to 5%. When the number of measurements is decreased, the affect of noise appears to be translated towards higher noise levels i.e. an optimal number of measurements (likely between 20 and 30 measurements) will be able to identify accurate models with more than 5% noise.

The Forward Euler method is most likely to have superfluous terms identified as part of the reaction network, these superfluous terms tend to have small coefficient values and therefore have a small impact on the change in concentration of any individual component. Most of the superfluous terms that appear when using the Forward Euler approach are using kinetic terms that are similar to the correct kinetic terms, for example, one of the repeats underestimates the coefficient associated with $[DG][MeOH]$ and also includes the additional term $[DG]$ or sometimes $[DG]^2$. The superfluous terms appear to be more common with higher number of

measurements. The frequency of superfluous terms increases as the number of measurements increases with the amount of noise being less impactful than the number of measurements, but the conditions that have the highest likelihood of finding a superfluous time is when both the number of measurements and noise is highest with eight of the repeats containing a superfluous term.
The Differentiated Spline approach has some similar errors occurring to the Forward Euler Method approach, i.e. once the number of measurements reduces below 10 measurements, there are no models identified. This is explained as there being too little information within the datapoints for the algorithm to identify. Using the differentiated spline does identify models at 10 measurements, however these models are difficult to identify and only a small number of the repeats generated models. As discussed previously, the values identified in

Table 4-22 are the average of the non-zero terms. This approach at low measurements tends to find either a model with most (or all) of the values in the correct positions or does not find any model to describe the data which is a benefit that does not occur within the Forward Euler approach. For example, for the condition 2% noise, and 10 measurement, five of the repeats generate a model that has reactions one and two with values in the correct positions and reaction three having a partially correct model. The other twenty-five repeats have no model identified at all. Superfluous terms are not common with the Differentiated Spline Approach, with them only occurring at lower noise levels & fewer measurements. The errors that occur do not appear to have any correlation at any specific condition with no repeating errors.

The Trapezium Rule approach (Table 4-23) has very similar results to the differentiated spline method (

Table 4-22, and Table 4-23), with the number of measurements having a minimum value to be effective of ten measurements, but the results at ten measurements are less likely to occur with many of the repeats not generating a model at all. The interesting difference between the Trapezium Rule approach and the Differentiated Spline approach is the stability of the coefficients throughout the different conditions. As the number of measurements decreases to 20 the average values do not change considerably, the results appear to have high precision with respect to the number of measurements. This is also apparent with the change in noise with the coefficients being the closest to the correct values of the three methods. This precision is likely due to the effects of the noise being reduced by the integration i.e. the noise will have a smaller affect on the area under the graph. This shows a benefit of the integral approaches over the differential approaches.

The number of measurements does appear to have a greater impact on the quality of result over the Differentiated Spline approach, this is likely due to the approximation of the integral becoming less accurate when the gap between measurements is too large. The number of superfluous terms increases as the number of measurements decreases. There are a variety of superfluous terms that occur and occur semi-regularly, these additional terms do not appear to have significant negative impact on the change in concentration of each of the components where these errors occur. One set of errors that appears more than once at 20 measurements, 3% noise changes the model to be (written in differential equation form):

$$\frac{d[TG]}{dt} = 0,$$

$$\frac{d[MeOH]}{dt} = -0.4[TG]^2,$$

$$\frac{d[DG]}{dt} = -0.14[DG][MeOH],$$

$$\frac{d[BD]}{dt} = 0.4[TG]^2,$$

$$\frac{d[MG]}{dt} = 0.14[DG][MeOH],$$

$$\frac{d[GL]}{dt} = 0,$$

This example of a model with superfluous terms shows that the models with superfluous terms can be a failure to identify a large portion of the reaction network, and the model that is found is a poor representation of the true model. This example model appears 5 times at this condition. The only other error that appears at that condition is a correct model with two additional reactions where the coefficient is a small number and has a small impact on the change in concentration of that component.

4.10 Conclusions

Overall, the different approaches have shown that there is a minimum amount of data required to generate an accurate model, and all the approaches appear to agree that between twenty and thirty measurements are sufficient for this reaction network. Twenty to thirty measurements appear to contain enough information about the concentration change of the components whilst not being corrupted by noise. Of the three approaches, the Differentiated Spline has the highest likelihood of identifying the correct structure, being the most likely to find a correct model at more challenging conditions and tends to produce models with the fewest superfluous terms. However, the precision of the integral approach does provide a benefit over the spline method if the coefficient of the relevant kinetic structure cannot be identified by different methods.

5 Chapter 5 Conclusions

5.1 Discussion

The stoichiometric algorithm has several limitations that were shown when the method was tested against a standard dataset of the thermal decomposition of α-pinene. The lack of reaction invariants in the α-pinene reaction network showed a limitation of the algorithm in finding the stoichiometries that this method is dependent on reaction invariants to identify stoichiometries. This can be partially remedied with *a priori* information as limitations can be placed on the identification of stoichiometries algorithm. Attempting this method on a larger reaction network, with more chemical components should be considered to test the limitations of the algorithm. The two CRNs this algorithm was applied to have five and six reactive species.

The identification of kinetic parameters was shown to be effective with several variations on the differential and integral approaches. The limitations of this approach were tested with respect to noise and the number of measurements being used in the identification process. However, a limitation that was not considered would be to use fewer datasets to identify the kinetic structure of the CRN. The datasets provided to the MILP include some datasets with no reaction taking place, whether this is necessary as a demonstration of the reaction not taking place has not been evaluated.

Both algorithms shown in this work operate on isothermal batch reactors and the reactions were irreversible, both assumptions may not be true in a real reaction network. For example, in an exothermic reaction whilst the temperature can be controlled tightly with suitable equipment, the assumptions that the algorithms are built upon are no longer valid and therefore the algorithms may fail to identify the relevant reaction properties.
If the reaction is reversible there are a variety of problems that could occur in the identification of the CRN. This has not been considered in the scope of this work, but applying this method to a reversible reaction would be an interesting application.

5.2 Conclusions

This thesis has shown that an MILP approach to the identification of chemical reaction networks stoichiometries and kinetics is possible with no *a priori* knowledge of the system in question.

Whilst *a priori* information is not essential for the identification of stoichiometries, it both can be used and is recommended to be used it to further reduce the number of feasible solutions to the stoichiometric problem. It was discovered from the α-pinene reaction network that there is a group of reaction networks that have very few invariants, and that this approach to the identification of stoichiometric modelling is not appropriate for this type of reaction network.

Similarly, the identification of kinetic models can be completed from concentration measurements only i.e. no *a priori* information, and is shown to be possible at 5% noise. The importance of the first few measurements in the identification process was discovered from the different approximation methods being considered.

The differentiated spline gave the best models among the differential approaches, with fewer errors in the identified models and the highest precision in the model coefficients of all the differential approximation methods.

The trapezium rule was the most reliable of the integral approximation techniques. Noise levels has a smaller impact on the integral approaches since the measurement errors have a smaller impact on the area under the graph.

These two methods are comparable with similar results at all conditions. However, since this is an offline approach to model identification there is no limit on the number of approaches that could be applied.

5.3 Future work

A logical progression of this work would be to identify bio-chemical kinetic models. The identification of chemical kinetics can be modified to also consider the growth mechanics of microorganisms. There are several approximations of the biological processes depending on how the microorganism is behaving. For this study the Monod, Haldane, AND and OR models of are being considered as possible structures for the biological kinetics. The growth rates of each of these models is shown in Equation 5-1:

Monod Model
$$\mu = \frac{\mu_{max}[S]}{[S] + K_s} \qquad \text{Equation 5-1a}$$

Haldane Model
$$\mu = \frac{\mu_{max}[S]}{[S] + K_s + \left(\frac{1}{75}[S]^2\right)} \qquad \text{Equation 5-1b}$$

AND Model
$$\mu = \frac{\mu_{max,1}[S_1]}{[S_1] + K_{s1}} * \frac{\mu_{max,2}[S_2]}{[S_2] + K_{s2}} \qquad \text{Equation 5-1c}$$

OR Model
$$\mu = \frac{\mu_{max,1}[S_1]}{[S_1] + K_{s1}} + \frac{\mu_{max,2}[S_2]}{[S_2] + K_{s2}} \qquad \text{Equation 5-1d}$$

Each of these growth rates (μ) need to be combined with the differential equation before it can be identified. With an accurate growth rate model, the rate of change in concentration of biomass is the current concentration of biomass multiplied by the growth rate.

Another possible avenue of further research would be to consider non-isothermal batch reactors, this would change the problem to a non-linear problem which could be solved using Mixed Integer Non-Linear Programming (MINLP). Within the Chemical Engineering space, MINLP has been used for design of plant layout (Ye et al., 2018). Attempts have been made to optimise multicomponent distillation operations, with some success. However due to the complexity of the problem, it has proven difficult with MINLP (Tumbalam Gooty et al., 2019). A review of MINLP's uses in process systems engineering is summarised by Trespalacios and Grossmann (2014). This limited literature review shows that there is an interest in process industries to implement optimisation techniques such as MINLP in further work. Typically an MINLP formulation would appear very similar to the generic MILP formulation shown below in Equation 5-2 where either the constraints (shown as equality and inequality constraints in terms of $\boldsymbol{A}$ and $\boldsymbol{b}$), or the cost function J (or both) would be non-linear depending on the type of non-linear optimisation taking place.

$$J = \min(\boldsymbol{Cx}) \qquad \text{Equation 5-2}$$

Subject to:

$$\boldsymbol{A_{eq}x} = \boldsymbol{b_{eq}}$$

$$\boldsymbol{Ax} \leq \boldsymbol{b}$$

$$lb \leq x \leq ub$$

6 Appendices

6.1 Appendix 1. Matlab code for Identification of Stoichiometries

Contents

```
clc
clear
close all

InitCond = [1,3,0,0,0,0];

Kinetics1 = [-0.4,0,0;...
            -0.4,-0.5,-0.2;...
            0.4,-0.5,0;...
            0.4,0.5,0.2;...
            0,0.5,-0.2;...
            0,0,0.2];
Useful1 = [4,11,13];


tspan = [0:0.3:7.5];
PercentNoise = [0.02];
APriori = 1;% IF AProri==1, use A Priori information, else, no priori information
% ADDING NOISE
[MeasurementsTime,Measurements(:,:)] = ode45(@(tspan,InitCond)biodieselratesKin(tspan,InitCond,Kinetics1,Useful1),tspan,InitCond);

Percent = PercentNoise; % This is for when running with different noise levels

for i = 1:size(InitCond,1)
sumoferror = 0;
for c = 1:size(InitCond,2)
    RangeC = (max(Measurements(:,c)-min(Measurements(:,c))))*Percent;
    awgnNoise = randn(size(Measurements(:,c)))*RangeC;
    sumoferroror = sumoferror+RangeC;

    YNoise(:,c) = Measurements(:,c) + awgnNoise;
    YNoise(1,c) = Measurements(1,c);
end
YNoise(YNoise(:,:) < 0) =0;
NMeasurements(:,:) = YNoise(:,:);


Dataset(:,:) = [MeasurementsTime,Measurements(:,:),NMeasurements(:,:)];
end



FullData1 = [MeasurementsTime,NMeasurements(:,1:6),293*ones(size(MeasurementsTime,1),1),ones(size(MeasurementsTime,1),1)];
```

Data set needs to be in the form:

```
%Columns: time, Concentrations, Temperature, Volume, FeedFlowrate
%Rows are the different time points
% System must be a batch process, feed flowrate must be entirely zeros.
% this setup is a remnant of when i was trying to get it work for fed batch
% processes.
```

```
NoComponents = 6;
NoInvariants = 20;
bounds = 3;
tries = 30;
NormData = [];

FullData2 = [FullData1(:,2:7)];
```

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

The following section is set up to check if t he data set is either noisy or clean

This will not be needed for real data sets, as it will be assumed to be noisy data

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

Normalise Data

```
for  j = 1:length(FullData2(:,1))
    for i = 1:length(FullData2(1,1:end))

        NormData(j,i) = (FullData2(j,i) - min(FullData2(:,i))) / (max(FullData2(:,i)) - min(FullData2(:,i)));

    end
end
```

```
Tolerance = 1;
%{
figure(1)
plot(FullData(:,1),NormData(:,1:end))
Button = questdlg('Does the data appear to be noisy?','Noise','Noisy','Clean','Unknown','Unknown');

switch Button
    case 'Noisy'
        % Noisy Data Transformations
        noise = 0.8;

    case 'Clean'
        % Clean data transformations
        noise = 0.001;

    case 'Unknown'
        % Mistakes have been made.
        msgbox({'Data set should be in matrix where columns are:','time, concentrations, temperature','Reconfigure data'})
        stop
end

close(1)
%}
```

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

The following section converts the raw data into a form suitable for 'GetInvariants6' code.

This includes a step for converting the data from conc. into moles This allows more functionality RE volume changes - this will potentially be a stepping stone for making this code suitable for fed batch processes

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

RelData = [FullData1(:,1:NoComponents+1),FullData1(:,NoComponents+3),FullData1(:,NoComponents+4:end)];


[Invariants,FalsePositives] = GetInvariants6(RelData,NoComponents,Tolerance,bounds);

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

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

    AtomicMatrix = [];



Invariants = Invariants(:,1:NoComponents);

Invariants = [Invariants;AtomicMatrix];
Invariants = unique(Invariants,'rows');
NoInvariants = size(Invariants,1);

Stoichs = GetStoichs(Invariants,NoComponents,NoInvariants,bounds,tries,APriori);

MoreStoichs = sort(Stoichs,2);
FullStoichs = unique(Stoichs,'rows');
UniqueStoichsWithoutA = unique(Stoichs(:,1:NoComponents),'rows');


clear Stoichs MoreStoichs FullStoichs
```

```
AtomicMatrix = [3,5,6,3;...
        1,4,1,0;...
        3,6,5,2;...
        1,3,2,1;...
        3,7,4,1;...
        3,8,3,0]';
 Invariants = [Invariants;AtomicMatrix];
 NoInvariants = size(Invariants,1);

Stoichs = GetStoichs(Invariants,NoComponents,NoInvariants,bounds,tries,APriori);

MoreStoichs = sort(Stoichs,2);
FullStoichs = unique(Stoichs,'rows');
UniqueStoichsWithA = unique(Stoichs(:,1:NoComponents),'rows');
%{
figure(1); plot(FullData(:,1),FullData(:,8:13))
title('Noisy data');

figure(2);plot(FullData(:,1),FilterData(:,1:6))
title('filtered data');

figure(3);plot(FullData(:,1),FullData(:,2:7))
title('clean data')
%}
```

Published with MATLAB® R2020a

Contents

```matlab
 function [UsefulInvars,FalsePositives]=GetInvariants6(FullData,NoComponents,noise,bounds)

k = 1;
Aeq = [];

for t = 1:length(FullData(:,1))
    for i = 1: NoComponents
        if FullData(t,1) == 0
            k = t;
        end
        Value(t,i)=(FullData(t,i+1)-FullData(k,i+1))/FullData(t,NoComponents+2);

    end
end



for i = 1:NoComponents

    for t = 2:length(FullData(:,1))
        if FullData(t,1) == 0
            k = t;
            continue
        end
        int(t,i) = trapz(FullData(t-1:t,1),Value(t-1:t,i));

    end
    %{
    figure(1)
    hold all
    plot(FullData(:,1),[Value(:,i),int(:,i)])
    legend('Raw Data','Integrated Data')
    close(1)
    %}
end

Aeq = [Aeq,int];
Aeq = [Aeq;Value];
%{
letters = {'TG','MeOH','DG','BD','MG','GL','Volume'};

for i = 1:NoComponents+1
   subplot(3,3,i),plot(FullData(:,1),Aeq(1:101,i))
   hold all
   subplot(2,3,i),plot(FullData(:,1),Aeq(102:202,i))

   xlabel('time /s')
   ylabel('Concentration / Moldm-3')
   legend('Integrated \Delta Concentration','\Delta Concentration')
   title(letters(i))

end
%}

UB = bounds;
LB = -bounds;
tol = 0.001;


% Sigma = sigma' + sigma'' -1
Afeq = [ zeros(NoComponents),-eye(NoComponents),-eye(NoComponents),eye(NoComponents),zeros(NoComponents,1),zeros(NoComponents,1);...
    zeros(1,NoComponents),zeros(1,NoComponents),zeros(1,NoComponents),-ones(1,NoComponents),ones(1,1),zeros(1,1)];...];

Bfeq = [-ones(NoComponents,1);...
    0];


for i = 1:(NoComponents)
MVector(i,1) = (10^i)+0.01;
end

MVector = sort(-MVector);
MVector = -MVector;


% Vars, slack, sigma', sigma'', sigma, sumofsigma , noise


% Slacks
A = [eye(NoComponents),zeros(NoComponents,3*(NoComponents)+1),-eye(NoComponents,1);...
    zeros(NoComponents),zeros(NoComponents,3*(NoComponents)+1),-eye(NoComponents,1);...
    ... % Sigma'
    eye(NoComponents),UB*eye(NoComponents),zeros(NoComponents),zeros(NoComponents),zeros(NoComponents,1),zeros(NoComponents,1);...
    -eye(NoComponents),(LB+tol)*eye(NoComponents),zeros(NoComponents),zeros(NoComponents),zeros(NoComponents,1),zeros(NoComponents,1);...
    ... % Sigma''
    -eye(NoComponents),zeros(NoComponents),(-LB)*eye(NoComponents),zeros(NoComponents),zeros(NoComponents,1),zeros(NoComponents,1);...
    eye(NoComponents),zeros(NoComponents),(LB+tol)*eye(NoComponents),zeros(NoComponents),zeros(NoComponents,1),zeros(NoComponents,1);...
    ...
    ...% Data
    Aeq(:,1:NoComponents), zeros(length(Aeq),NoComponents),zeros(length(Aeq),NoComponents),zeros(length(Aeq),NoComponents),zeros(length(Aeq),1),-ones(length(Ae
    -Aeq(:,1:NoComponents), zeros(length(Aeq),NoComponents),zeros(length(Aeq),NoComponents),zeros(length(Aeq),NoComponents),zeros(length(Aeq),1),-ones(length(A


B = [(10+ tol)*ones(NoComponents,1);...
     tol*ones(NoComponents,1);...
    ...
    (UB)*ones(NoComponents,1);...
    -tol*ones(NoComponents,1);...
    ...
    -LB*ones(NoComponents,1);...
    -tol*ones(NoComponents,1);...
    ...
    tol*ones(length(Aeq),1);...
    -tol*ones(length(Aeq),1)];


lb = [LB*ones(NoComponents,1);zeros(3*(NoComponents),1);0;tol];
ub = [UB*ones(NoComponents,1);ones(3*(NoComponents),1);inf;noise];

f = [zeros(NoComponents,1);zeros(3*NoComponents,1);1;1];

intcon = [1:length(f)-1];
Invariants = [];
UsefulInvars = [];
FalsePositives = [];
Unsure = [];
AllPolynomials = [];
UsePolys = [];
AllMean = [];
UseMean = [];

it= 1;
options = optimoptions('intlinprog','display','off');

while   size(UsefulInvars,1) < 100 && it <= 20
```

```
[Invariant,fval,exitflagint] = intlinprog(f,intcon,A,B,Afeq,Bfeq,lb,ub,options);

if exitflagint == -2
    if size(Unsure,1) > 0
    msgbox('Unsure Matrix is not empty.');
    disp(Unsure)
    end
    it = it+1;
break
end

%{
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Testing solutions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
errors = Aeq(:,7:12)*Invariant(1:6);
   clean = Aeq(:,1:6)*Invariant(1:6);

 P = polyfit([FullData(:,1);FullData(:,1)],errors(:),1);
   M = mean(errors);

Mplot = M*ones(2*length(FullData(:,1)),1);
   y = P(1)*[FullData(:,1);FullData(:,1)]+P(2);


plot([FullData(:,1);FullData(:,1)],[errors,clean,y,Mplot])
   legend('errors','clean','y','mean')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%}

% Store Invariants
Invariants = [Invariants;round(Invariant(1:NoComponents))'];

% Integer cut
Indexzeros = find(Invariant(NoComponents+1:end-1)==0);
Indexones = find(Invariant(NoComponents+1:end-1));
Numberzeros = length(Indexzeros);
Numberones = length(Indexones);

temp = zeros(4*NoComponents+2,1)';
%temp = [round(Invariant(1:NoComponents)'),0];

temp(Indexones+NoComponents) = 1;
temp(Indexzeros+NoComponents) = -1;
temp(end-1) = -1;

tempb = 3*NoComponents - 1 - Numberzeros;

A = [A;temp];
B = [B;tempb];
```

Checking Invars (automatically)

```
   [NoInvars] = size(Invariants);
   errors = Aeq(:,1:NoComponents)*Invariant(1:NoComponents);
```

```
    c ean = Aeq(:,1:NoComponents)* nvariant(1:NoComponents);

    P = polyfit([FullData(:,1);FullData(:,1)],errors(:),1);
    M = mean(errors);

    AllPolynomials = [AllPolynomials;P];
    AllMean = [AllMean;M];
    Mplot = M*ones(2*length(FullData(:,1)),1);
    y = P(1)*[FullData(:,1);FullData(:,1)]+P(2);
```

Checking Invars (Manually)

```
    if P(1)<=1e-2 && P(1)>=-1e-2 && P(2)<=5e-2 && P(2) >= -5e-2 && M <= 0.1 && M >= -0.1

            figure
    plot([FullData(:,1);FullData(:,1)],[errors,clean,y,Mplot])
    legend('errors','clean','y','mean')

    button = questdlg('Does this Data set appear to be about zero','question','yes','no','unsure','unsure');

    switch button
    case 'yes'
         UsefulInvars = [UsefulInvars;Invariants(end,1:NoComponents)];

    case 'no'
         FalsePositives = [FalsePositives;Invariants(end,1:NoComponents)];

    case 'unsure'
          Unsure = [Unsure;Invariants(end,1:NoComponents)];
    end

     close all

     UsePolys = [UsePolys;P];
     UseMean = [UseMean;M];
 %}
    end

    %P = polyfit([FullData(:,1),FullData(:,1)],errors(:),1);
    %}
```

While Loop requirements

```
   it = it+1;
```

```
end
```

```
Not enough input arguments.

Error in GetInvariants6 (line 6)
for t = 1:length(FullData(:,1))
```

Published with MATLAB® R2020a

Contents

```
function [Storage] = GetStoichs(Invariants,NoComponents,NoInvariants,bounds,tries,APriori)
```

```
tol = 0.01;


Identity = -eye(NoInvariants,NoComponents);
```

```
Not enough input arguments.

Error in GetStoichs (line 8)
Identity = -eye(NoInvariants,NoComponents);
```

build the A matrix and b vector for the optimiser

```
APrioriKnowledge = [];
APrioriKnowledgeB = [];
if APriori == 1
APrioriKnowledge = [1,zeros(1,NoComponents-1),zeros(1,(NoComponents*4)+1);...
                    0,1,zeros(1,NoComponents-2),zeros(1,(NoComponents*4)+1);...
                    0,0,0,-1,0,0,zeros(1,(NoComponents*4)+1);...
                    0,0,0,0,0,-1,zeros(1,(NoComponents*4)+1);...
zeros(1,NoComponents),zeros(1,NoComponents),ones(1,NoComponents),zeros(1,NoComponents),-ones(1,NoComponents),0];

                APrioriKnowledgeB = [0;0;0;0;2];
end
 %}
%{
For Biodiesel Reaction:
                    1,zeros(1,NoComponents-1),zeros(1,(NoComponents*4)+1);...
                    0,1,zeros(1,NoComponents-2),zeros(1,(NoComponents*4)+1);...
                    0,0,0,-1,0,0,zeros(1,(NoComponents*4)+1);...
                    0,0,0,0,0,-1,zeros(1,(NoComponents*4)+1);...
zeros(1,NoComponents),zeros(1,NoComponents),ones(1,NoComponents),zeros(1,NoComponents),-ones(1,NoComponents),0
B = [0;0;0;0;2]

For VdV
                    1,0,0,0,zeros(1,NoComponents*4+1);...
                    0,-1,0,0,zeros(1,NoComponents*4+1);...
                    0,0,0,-1,zeros(1,NoComponents*4+1);...
                    zeros(1,NoComponents),zeros(1,NoComponents),ones(1,NoComponents),zeros(1,NoComponents),-ones(1,NoComponents),0];

                B = [0;0;0;2];
For NaCl
                1,0,0,0,zeros(1,NoComponents*4+1);...
                    0,1,0,0,zeros(1,NoComponents*4+1);...
                    0,0,0,-1,zeros(1,NoComponents*4+1);...
                    0,0,-1,-1,zeros(1,NoComponents*4+1);...
                    zeros(1,NoComponents),zeros(1,NoComponents),ones(1,NoComponents),zeros(1,NoComponents),-ones(1,NoComponents),0];
                B = [0;0;0;0;2];
%}


StoredStoichiometry = [];
Storage = [];
BigM = 10;



Mvector = [];

for i = 1:3*NoComponents
   Mvector(i,1) = 10^(i+1);
end
```

```
A = [Invariants,Identity,zeros(NoInvariants,3*NoComponents+1);...
    -Invariants,Identity,zeros(NoInvariants,3*NoComponents+1);...
    ...
    eye(NoComponents),-eye(NoComponents),zeros(NoComponents,3*NoComponents+1);...
    zeros(NoComponents),eye(NoComponents),zeros(NoComponents,3*NoComponents+1);...
    ...Binaries
    eye(NoComponents),zeros(NoComponents),bounds*eye(NoComponents),zeros(NoComponents),zeros(NoComponents),zeros(NoComponents,1);...
    -eye(NoComponents),zeros(NoComponents),(-bounds+tol)*eye(NoComponents),zeros(NoComponents),zeros(NoComponents),zeros(NoComponents,1);...
    zeros(1,NoComponents),zeros(1,NoComponents),ones(1,NoComponents),zeros(1,NoComponents),-ones(1,NoComponents),zeros(1,1);...
    ...
    -eye(NoComponents),zeros(NoComponents),zeros(NoComponents),bounds*eye(NoComponents),zeros(NoComponents),zeros(NoComponents,1);...
    eye(NoComponents),zeros(NoComponents),zeros(NoComponents),-(bounds+tol)*eye(NoComponents),zeros(NoComponents),zeros(NoComponents,1);...
    ...
    APrioriKnowledge];

B = [zeros(NoInvariants,1);...
    zeros(NoInvariants,1);...
     ...
    (BigM+tol)*ones(NoComponents,1);...
    tol*ones(NoComponents,1);...
    ...
    bounds*ones(NoComponents,1);...
    -tol*ones(NoComponents,1);...
    2;...
    ...
    bounds*ones(NoComponents,1);...
    -tol*ones(NoComponents,1);...
    ...
    APrioriKnowledgeB];

Aeq = [zeros(NoComponents),zeros(NoComponents),-eye(NoComponents),-eye(NoComponents),eye(NoComponents),zeros(NoComponents,1);...
      zeros(1,NoComponents),zeros(1,NoComponents),zeros(1,NoComponents),zeros(1,NoComponents),-ones(1,NoComponents),ones(1,1)];

Beq = [-ones(NoComponents,1);...
    0];
```

```
f = [zeros(NoComponents,1);-ones(NoComponents,1);Mvector;0];

lb = [-bounds*ones(NoComponents,1);-inf(NoComponents,1);zeros(3*NoComponents,1);0];
ub = [bounds*ones(NoComponents,1);inf(NoComponents,1);ones(3*NoComponents,1);inf];
options = optimoptions('intlinprog','display','off');
```

```
for ii = 1:tries
```

```
 [Stoichiometry,error,exitflag] = intlinprog(f,[1:length(f)],A,B,Aeq,Beq,lb,ub,options);

 if exitflag == -2
     break
 end
```

Save the results of the solver

```
Stoichiometry = round(Stoichiometry);
Storage = [Storage;transpose(Stoichiometry)];
```

Integer cut

```
Indexzeros = find(Stoichiometry(2*NoComponents+1:end)==0);
Indexones = find(Stoichiometry(2*NoComponents+1:end));
Numberzeros = length(Indexzeros);
Numberones = length(Indexones);

temp = zeros(1,5*NoComponents+1);

temp(Indexones+2*NoComponents) = 1;
temp(Indexzeros+2*NoComponents) = -1;

tempb = 3*NoComponents - 1 - Numberzeros;

A = [A;temp];
B = [B;tempb];

%}
```

```
end
```

Published with MATLAB® R2020a

6.2 Appendix 2. Matlab code for Identification of Kinetics

Contents

Simplifying code for Identifying kinetics using MILP

```
clc
clear
global NoComponents Bounds tol
%global Keep
NoComponents = 6;
NoCombos = 28;
qqq=1;

        i1 = 1;
        i2 = 5;
        i3 = 3;
k=1;
for j = 1:6
j1 = i2;
InitCond2(k,:) = zeros(1,6);
InitCond2(k,j) = j1;
k = k+1;

    for i = 1:6
        if i==j
            continue
        end
        InitCond2(k,:) = zeros(1,6);

InitCond2(k,j) = j1;
InitCond2(k,i) = i1;
k = k+1;
InitCond2(k,:) = zeros(1,6);
InitCond2(k,i) = i2;
InitCond2(k,j) = j1;
k=k+1;
InitCond2(k,:) = zeros(1,6);
InitCond2(k,i)=i3;
InitCond2(k,j)= i3;
k=k+1;
    end
end

InitCond2 = unique(InitCond2,'rows');
InitCond = [];%[3,1,0,0,0,0];
InitCond = [InitCond;InitCond2];

version = 'Int1DataNoise5';
filename = 'Int1Noise5';
pathname = 'C:\Users\cjdix\Documents\MATLAB\Plots for K Identification\Int1NoiseVar';
%tvar = [0.0375, 0.05, 0.075, 0.1, 0.15, 0.3];

%PercentNoise = [0, 0.01, 0.02,  0.03, 0.05];
Alltvars = [75, 30, 20, 10, 5, 2];

PercentNoise = [0.05];%:0.005:0.05];
ErrBounds = 50;
Bounds = [1];
tol = 1e-3;
oo = 5; % The top oo models are used
CC = 3; % Number of Cost function attempts
atts = 3; % number of repeats
repeats = 30;
tend = 7.5;
tvar = [tend./Alltvars];
```

```
sumoferror = 0;
Kinetics1 = [-0.4,0,0;...
            -0.4,-0.5,-0.2;...
            0.4,-0.5,0;...
            0.4,0.5,0.2;...
            0,0.5,-0.2;...
            0,0,0.2];
Useful1 = [4,11,13];

Condition = input('Conditions? Clean (1) or Noisy Data (2)');
Method = input('Differential (1) or Integral (2)');
if Method ==1
dCmethod = input('1.Fwd Euler, 2.Central Euler, 3.Taylor, 4.Spline');
elseif Method ==2
dCmethod = input('1.trapz, 2.Rectangle, 3. Simpsons3/8, 4.Spline? ');
end

Model = zeros(NoComponents,NoCombos,repeats);
MC = cell(size(Model,3),1);
```

```
Error using input
Cannot call INPUT from EVALC.

Error in simpleMILPforK12 (line 77)
Condition = input('Conditions? Clean (1) or Noisy Data (2)');
```

```
%InitCond = InitCond(32,:);
for tt = 1:size(tvar,2)
    clear tspan Measurements NMeasurements AICvals2 tID2 CxID2 CT3 AICvals3 TVal2 Val
    tspan = [0:tvar(tt):tend];

%{
%% Reaction Info:
TG + MeOH -> DG + BD
DG + MeOH -> MG + BD
MG + MeOH -> GL + BD

TG = C1
MeOH = C2
DG = C3
BD = C4
MG = C5
GL = C6

%}
for i = 1:size(InitCond,1)
    Inity = InitCond(i,:);
[MeasurementsTime,Measurements(:,:,i)] = ode45(@(tspan,Inity)biodieselratesKin(tspan,Inity,Kinetics1,Useful1),tspan,Inity);
end

for jjj = 1:length(PercentNoise(1,:))
 clear Percent
Percent = PercentNoise(jjj);

for rrr = 1:repeats
```

```
clear  YNoise AICvals2 AICvals3 KKKStore CxID2 YNoise2  Dataset dC_dt1 dC_dt2 combo1 combo2 DeltaC1 DeltaC2 trapzcombos1 trapzcombos2 Dataset2 Dataset22 Datas

    for i = 1:size(InitCond,1)

for c = 1:size(InitCond,2)
    RangeC = (max(Measurements(:,c,i)-min(Measurements(:,c,i))))*Percent;
    awgnNoise = randn(size(Measurements(:,c,i)))*RangeC;
    sumoferroror = sumoferror+RangeC;

    YNoise(:,c,i) = Measurements(:,c,i) + awgnNoise;
    YNoise(1,c,i) = Measurements(1,c,i);
end
YNoise(YNoise(:,:,i) < 0) =0;
NMeasurements(:,:,i) = YNoise(:,:,i);


Dataset(:,:,i) = [MeasurementsTime,Measurements(:,:,i),NMeasurements(:,:,i)];
    end

KStore2 = zeros(oo,NoCombos,NoComponents);

for ii = 1:NoComponents
```

```
    clear CxID AICvals
 %{
switch ii
       case 1 % TG
           ABinAdd =    [zeros(1,NoCombos),zeros(1,1), zeros(1,1), zeros(1,NoCombos), zeros(1,NoCombos),  -ones(1,1)];
           BBinAdd =    [-1];


       case 2 % MeOH, sum -ve >= 1 & sum+ve ==0 (i.e. Reactant only)
           ABinAdd =    [zeros(1,NoCombos),zeros(1,1), zeros(1,1), zeros(1,NoCombos), zeros(1,NoCombos),  -ones(1,1)];
           BBinAdd =    [-1];
```

```
        case 3 % DG, sum -ve>=1  &  sum +ve <= 1 (i.e. Intermediate)
            ABinAdd =    [zeros(1,NoCombos),zeros(1,1), zeros(1,1), -ones(1,NoCombos), zeros(1,NoCombos),  -ones(1,1)];
            BBinAdd =    [-1];


        case 4 % BD, sum +ve >= 1  &  sum -ve ==0, (i.e. Product)
            ABinAdd = [zeros(1,NoCombos),zeros(1,1), zeros(1,1), zeros(1,NoCombos),zeros(1,NoCombos),    -ones(1,1)];
            BBinAdd =    [-1];


        case 5 % MG, sum -ve>=1  &  sum +ve <= 1 (i.e. Intermediate)
            ABinAdd = [zeros(1,NoCombos),zeros(1,1), zeros(1,1), zeros(1,NoCombos), zeros(1,NoCombos),  -ones(1,1)];
            BBinAdd =    [-1];


        case 6 % GL, sum +ve >= 1  &  sum -ve ==0, (i.e. Product)
            ABinAdd = [zeros(1,NoCombos),zeros(1,1), zeros(1,1), zeros(1,NoCombos),zeros(1,NoCombos),   -ones(1,1)];
            BBinAdd =    [-1];


    end
    %AAdd = [];
    %BAdd = [];
     %}
    ABinAdd = [];
    BBinAdd = [];

itert = 0;
clear RelAIC All

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
while itert < 3
```

```
    clear A B errors1 SqErr

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
f = [zeros(1,NoCombos), 0,  1,    zeros(1,2*NoCombos), 1 ]; % Cost function
% cost function -> only the error term and the number of non-zero terms are
% minimised (both currently set to value of 1).
KineticsGuess = zeros(1,length(f));

switch ii % We know the number of reactions (jj) that each component takes place in.
     case 1
         jj = 1; % Number of Reactions that take place for component ii
         itert = itert+1;
         KineticsGuess(4) = -0.4;
     case 2
         jj = 3;
         itert = itert+1;
         KineticsGuess(4) = -0.4;
         KineticsGuess(11) = -0.5;
         KineticsGuess(13)= -0.2;
     case 3
         jj = 2;
        itert = itert+1;
        KineticsGuess(4) = 0.4;
         KineticsGuess(11) = -0.5;
     case 4
         jj = 3;
        itert = itert+1;
        KineticsGuess(4) = 0.4;
         KineticsGuess(11) = 0.5;
         KineticsGuess(13)= 0.2;
     case 5
         jj = 2;
         itert = itert+1;
         KineticsGuess(11) = 0.5;
         KineticsGuess(13)= -0.2;
     case 6
         jj = 1;
         itert = itert+1;
         KineticsGuess(13)= 0.2;
    otherwise
        disp('There are 6 components that can be identified. Pick a number between 1 and 6');
 end

switch Condition
    case 1
        a = 2:7;
        itert = itert + 1;
    case 2
        a = 8:13;
        itert = itert + 1;
    otherwise
        a = 1:6;
        disp('To use Clean data set choose 1, to use Noisy data set choose 2');
        itert = 0;
end
componentname = {'TG','MeOH','DG','BD','MG','GL'};
Rowsoftime = 0;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i = 1:size(Dataset,3)
Concentrations(:,:,i) = Dataset(:,a,i);

Time(:,i) = Dataset(:,1,i);

Rowsoftime =Rowsoftime + length(Time(:,i))-1;

if i==66
    pause (2)
end

[dC_dt1(:,:,i), combo1(:,:,i), DeltaC1(:,:,i), trapzcombos1(:,:,i), comboname, Timeint] = ...
                    datatransform(Concentrations(:,:,i), Time(:,i),componentname, dCmethod);
end



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

Plotting to find errors in the data transform function

```
%{

%figure(1);plot(Time(1:end-1,1),dC_dt1(:,:,50));title('Approximation of Differential');xlabel('Time / hrs')
%figure(9);plot(Time(:,1),combo1(:,:,50));title('combos');xlabel('Time / hrs')
%figure(3);plot(Time(1:end-1,1),DeltaC1(:,:,50));title('DeltaC');xlabel('Time / hrs')
%%%%%
figure(10);plot(Time(1:end-1,1),trapzcombos1(1:end-1,:,50));title('Approximation of Integral using an Integrated Spline');
%%%%%
xlabel('Time / hrs'); ylabel('Integrated Change in Concentration / mol.dm^-^3.hr')
%legend('\int[TG].dt','\int[MeOH].dt','\int[DG].dt','\int[BD].dt','\int[MG].dt','\int[GL].dt')
%%%%%
%saveas(figure(10),fullfile(fname,[filename,'.fig']));saveas(figure(10),fullfile(fname,[filename,'.bmp']))
%%%%%


%figure(8);plot(Time(:,1),Concentrations(:,:,50));title('Concentration Profiles');xlabel('Time / hrs')
%ylabel('Concentrations / mol.dm^-3')
%xlabel('Time / hrs')
%legend('TG','MeOH','DG','BD','MG','GL')
%%
%{
for i = 1:5
    figure(i);clf
end
title('Approximation of the differential using the Taylor Series')
legend('dTG/dt','dMeOH/dt','dDG/dt','dBD/dt','dMG/dt','dGL/dt')
ylabel('Change in Concentration  mol.dm^-^3.hr^-^1')
%}
for i = 1:5
    figure(i);clf
end
%}
dC_dt = [];
combos = [];
DeltaC= [];
trapzcombos=[];
switch Method
    case 1
        for i = 1:size(Dataset,3)
            dC_dt = [dC_dt;dC_dt1(1:end,:,i)];
            combos = [combos;combo1(1:end-1,:,i)];
        end
        Inputs1 = [dC_dt(:,:,:)];
        Inputs2 = [combos];
        itert = itert+1;
    case 2
        for i = 1:size(Dataset,3)
           DeltaC = [DeltaC;DeltaC1(:,:,i)];
           trapzcombos = [trapzcombos;trapzcombos1(:,:,i)];
        end


        Inputs1 = [DeltaC];
        Inputs2 =[trapzcombos];
        itert = itert+1;

    otherwise
        disp('To use Differential method use 1,or use Integral method use 2')
        itert = 0;

end

NoCombos = size(combo1,2);
```

```
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%{
Testing Inputs accuracy
Sol = [-0.3,0,0;...
       -0.3,-0.3,-0.1;...
```

```
        0.3,-0.3,0;...
         0.3,0.3,0.1;...
         0,0.3,-0.1;...
         0,0,0.1];
% Usefull is the potions of variables.



%}
CFvalues = [];
for cc = 1:CC
    CFvalues(cc) = cc-1;
    CFvalues(CC+cc)=(cc*0.1);
end
CFvalues(end+1) = 10;

CFvalues = sort(CFvalues);
CFvalues = unique(CFvalues);

KineticStore = [];
for b = 1:size(Bounds,2)
for e = 1:size(ErrBounds,2)



for cc = 1:length(CFvalues)
```

```
f = [0,  zeros(1,NoCombos),  1,    zeros(1,2*NoCombos), CFvalues(cc) ]; % Cost function revamp
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Playing with this section at the moment to get exact derivatives

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

Data

Referred to as inputs as depending on the 'method' (above) they can be dC_dt & combos or DeltaC & trapzcombos 1. 0 = -y + mx - error 2. 0 = y - mx - error

```
AData = [-Inputs1(:,ii), Inputs2(:,:), -ones(Rowsoftime,1), zeros(Rowsoftime,NoCombos*2+1);...
          Inputs1(:,ii), -Inputs2(:,:), -ones(Rowsoftime,1), zeros(Rowsoftime,NoCombos*2+1)];

BData = [zeros(Rowsoftime,1);...
         zeros(Rowsoftime,1)];
```

Binary variable constraints

i,e, these 4 equations convert the nonzero terms into a 1 in either

```
 ABin = [zeros(NoCombos,1),   eye(NoCombos),    zeros(NoCombos,1),    (Bounds(b)-tol)*eye(NoCombos), zeros(NoCombos),              zeros(NoCombos,1);...
        zeros(NoCombos,1),  -eye(NoCombos),   zeros(NoCombos,1),   -Bounds(b)*eye(NoCombos),      zeros(NoCombos),              zeros(NoCombos,1);...
        ...
        zeros(NoCombos,1), -eye(NoCombos),    zeros(NoCombos,1),    zeros(NoCombos),             (Bounds(b)-tol)*eye(NoCombos),   zeros(NoCombos,1);...
        zeros(NoCombos,1),  eye(NoCombos),    zeros(NoCombos,1),    zeros(NoCombos),            (-Bounds(b)-tol)*eye(NoCombos),   zeros(NoCombos,1)];


 BBin=[(Bounds(b)-tol)*ones(NoCombos,1);...
        zeros(NoCombos,1);...
        ...
        (Bounds(b)-tol)*ones(NoCombos,1);...
        zeros(NoCombos,1)];

 Aalt = [zeros(1,1),       zeros(1,NoCombos),    zeros(1,1),            zeros(1,NoCombos),          zeros(1,NoCombos),          ones(1,1);...
         zeros(1,1),       zeros(1,NoCombos),    zeros(1,1),            zeros(1,NoCombos),          zeros(1,NoCombos),          -ones(1,1)];

 Balt = [NoCombos;...
        -1];

  A = [ABin;Aalt;AData;ABinAdd];
  B = [BBin;Balt;BData;BBinAdd];
```

equality constraints

```
 Aeq = [zeros(1,1),       zeros(1,NoCombos),     zeros(1,1),            -ones(1,NoCombos),     -ones(1,NoCombos),    1;...
        ABinAdd];

 Beq = [0;...
        BBinAdd];
```

Other terms

```
% Integer Constraint on dC_dt (must be 1) & all the binary variables must be either 1 or 0
intcon = [1, (length(f)-2*NoCombos):length(f)];
% Lower and upper bounds \-> the upper bound of the error variable has been
% set at 3 as an arbitrary upper bound of 3, this effects the r
%
%for e = 1:size(ErrBounds,1)
```

```
LB = [0.99;     -Bounds(b)*ones(NoCombos,1);             0;     zeros(2*NoCombos,1);               0];
UB = [1.01;      Bounds(b)*ones(NoCombos,1);      ErrBounds(e);     ones(2*NoCombos,1);         NoCombos];

%options = optimoptions('intlinprog','Display','off');
```

Identifying models

Finds 5 models that best approximate the system

```
    for iter = 1:atts
```

```
  options = optimoptions('intlinprog','display','off');
[Kinetics, fval, exitflag] = intlinprog(f,intcon,A,B,Aeq,Beq,LB,UB,[],options);

    if  isempty(Kinetics) || fval == -2% i.e. if no feasible point found

       Kinetics = [zeros(NoCombos,1);1;100;ones(2*NoCombos+1,1)];
    end
KineticStore = [KineticStore;Kinetics(2:29)'];
```

Integer Cut to find multiple solutions

```
IndexOnes = find(Kinetics(NoCombos+3:end-1));
IndexZeros = find(Kinetics(NoCombos+3:end-1)==0);
NumberOnes = length(IndexOnes);
NumberZeros = length(IndexZeros);

FalsePositive = zeros(1,length(f));

FalsePositive(NoCombos+2+IndexZeros) = -1;
FalsePositive(NoCombos+2+ IndexOnes) = 1;

FalseB = 2*NoCombos - 1 - NumberZeros;
A = [A;FalsePositive];
B = [B;FalseB];
```

```
    end
    if isempty(KineticStore)
     continue
    end
```

```
end

end
end
 KineticStore = round(KineticStore(:,:),5);
 KineticStore = unique(KineticStore(:,:),'rows');
```

Cleaning up the kinetic store - A lot of the solutions contain 1e-15 magnitude zeros that

```
 for r = 1:size(KineticStore(:,1),1)
     for c = 1:size(KineticStore(1,:),2)

        if  KineticStore(r,c)<1e-5 && KineticStore(r,c)>-1e-5
            KineticStore(r,c) = 0;
        end
     end
 end

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

Testing of the models

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

% Random Initial Conditions
Useful1 = [4,11,13];
InitTest = [1,5,0,0,0,0;...
            3,4,0,0,0,0;...
            5,5,0,0,0,0;...
            1,5,3,0,0,0;...
            1,1,1,1,0,0;...
            5,1,0,0,0,0;...
            ... 0,5,1,0,0,0;...
            ...0,5,0,0,1,0;
            1,1,0,1,0,1];

for i = 1:size(InitTest,1)
```

Test (Comparison) Data

```
    C0 = InitTest(i,:);
     [tT,CxT(:,:,i)] = ode45(@(tspan,C0)biodieselratesKin(tspan,C0,Kinetics1,Useful1),tspan,C0);
```

```
        for c = 1:size( nit est,2)

            RangeC = (max(CxT(:,c,i))*Percent);
            sumoferroro` = sumoferror+RangeC;

            YNoise2(:,c) = CxT(:,c,i) + awgnNoise;
            YNoise2(1,c) = CxT(1,c,i);
        end
        YNoise2(YNoise2(:,:) < 0) =0;
        NMeasurements(:,:,i) = YNoise2(:,:);
% Recording: DatasetTime & Dataset2

CID = C0(1,ii)*ones(1,size(KineticStore,1));
dC_dtID = zeros(size(NMeasurements,1),1);
NUseful = [];
```

Time step errors (1 step error)

```
    for l = 1:size(KineticStore,1)
        for r = 1:size(KineticStore,2)
            Q = sum(abs(KineticStore(:,r)));
            if Q>=0.01
                NUseful = [NUseful,l];
            end
        end
        for t2 = 2:size(NMeasurements,1)
            k=1;
            combok(k) = [1];
            k = k+1;
            C0 = NMeasurements(t2,:);
                for b = 1:size(NMeasurements,2)
                combok(k) = C0(b);
                k=k+1;
                    for bb = b:size(NMeasurements,2)
                        combok(k) = C0(b)*C0(bb);
                        k = k+1;
                    end
                end
            dC_dtID(t2,l) = [0];

                for k = 1:size(combok,2)
                    dC_dtID(t2,l) = dC_dtID(t2,l)+ KineticStore(l,k)*combok(k)*Timeint;
                end
            CID(t2,l) = CID(t2-1,l)+dC_dtID(t2,l);
         end
    end

 CxID(:,:,i) = CID;
```

```
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Test Error

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
I3=[];
for init = 1:size(InitTest,1)
    y=0;
   [AICvals(init,:)] = AICerrors(CxT(:,ii,init),CxID(:,:,init),KineticStore(:,:));
   for init2 = 1:size(AICvals,2)
        if AICvals(init,init2)==min(AICvals(init,:)) && y == 0
            y=1;
        elseif AICvals(init,init2)==min(AICvals(init,:)) && y~=0
            AICvals(init,init2) = 10000;
        end
   end
    minerror(init) = find(AICvals(init,:)==min(AICvals(init,:)));
    minvalue(init) = AICvals(init,minerror(init));

    for  c = 1:size(AICvals,2)
        RelAIC(init,c) = exp(minvalue(init)-AICvals(init,c));
    end

end
%{

 Alternative method of choosing which Kinetics to choose


k=1;
All=[]; All2=[];
for i = 1:size(RelAIC,2)
All(i)= sum(RelAIC(:,i));
All2(i) = mean(AICvals(:,i));
    if All(i) > 0.5
    KStore2(k,:,ii) = KineticStore(i,:);
    k=k+1;
    end
end
%}
for m = 1:size(AICvals,2)
All(1,m) = sum(AICvals(:,m));
```

```
end
I3 = []; I4=[];
[I3,I4] = mink(All,oo);

I1=[];I2=[];
        for p = 1:size(AICvals,1)

            [I1(p,:),I2(p,:)] = mink(AICvals(p,:),oo);

        end
%I3 = [I3;I2];
%I3=unique(I3);
KStore2(1:size(I4,2),:,ii) = KineticStore(I4,:);
if ii == 2 || ii == 4
   LL = 1;
end
```

```
end
%disp(ii);


KCell = cell(NoComponents,1);

for c = 1:NoComponents

k = 1;
    for cc = 1:size(KStore2,1)
        if sum(abs(KStore2(cc,:,c)))>0
            KCell{c,1}(k,:)= KStore2(cc,:,c);
            k=k+1;
        end
    end
end


[KStoreNew2] = AllPossibleKinetics(KCell);
       for iters = 1:size(KStoreNew2,1)
           KStoreNew3(:,:,iters) = KStoreNew2{iters}(:,:);
       end

CT2 = zeros(1,NoComponents,size(KStoreNew3,3));

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

Removes the 'superflous' terms

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

Record = [];
for n = 1:size(KStoreNew3,3)
    pp=1;
        for c = 1:size(KStoreNew3,2)
            ColPos = 0; ColNeg = 0; ColZero = 0;
                for r = 1:size(KStoreNew3,1)
                        if KStoreNew3(r,c,n)>0
                            ColPos = ColPos+1;
                        elseif KStoreNew3(r,c,n)<0
                            ColNeg = ColNeg+1;
                        else
                            ColZero = ColZero+1;
                        end
                end

             if ColPos > 0 && ColNeg >0
                Record(n,pp) = c;
                pp = pp+1;
             elseif ColZero == NoComponents

             else
                   KStoreNew3(:,c,n)= zeros(NoComponents,1);
             end

        end

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Averaging the coefficients? I guess?

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for n = 1:size(Record,1)
    for r = 1:size(Record,2)
    if Record(n,r) ==0
       continue
    end
      Column = KStoreNew3(:,Record(n,r),n);

      for c = 1:size(Column,1)
            if Column(c,1) < 0
                Kins(c,1) = -1;
            elseif Column(c,1) > 0
                Kins(c,1) = 1;
            else
```

```
                Kins(c,1) = 0;
            end

      end
        [L,~,LL] = find(Column);
        out = accumarray(L,LL,[],@mean);

        Now = mean(abs(out));

NewColumn = ones(NoComponents,1)*Now;%.*Kins;
for c = 1:size(NewColumn,1)
   NewColumn(c,1) = NewColumn(c,1)*Kins(c,1);
end
      KStoreNew3(:,Record(n,r),n) = NewColumn;
    end
end



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

Time step errors (consecutive)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
I1=[];
I2=[];

for init = 1:size(InitTest,1)

   C0 = InitTest(init,:);

   for k = 1:size(KStoreNew3,3)



       CT2(1,:,k) = InitTest(init,:);
       C0 = InitTest(init,:);

        for t2 = 2:size(NMeasurements,1)
              combo2(1) = 1;j=2;
            for c = 1:NoComponents
                   combo2(j)= C0(c);
                    j=j+1;
                for cc = c:NoComponents
                   combo2(j) = C0(c)*C0(cc);
                   j=j+1;
                end
            end

                    for c = 1:NoComponents
                        dC_dt2(t2,c,k) = KStoreNew2{k}(c,:)*tvar(tt)*combo2';
                        CT2(t2,c,k) = CT2(t2-1,c,k)+dC_dt2(t2,c,k);
                    end
            C0 = CT2(t2,:,k);
        end



   end
   [AICvals2(init,:)] = AICerrorsFull(CxT(:,:,init),CT2(:,:,:),KStoreNew3(:,:,:));

   [I1,I2(init,:)] = mink(AICvals2(init,:),100);

   KKKStore = KStoreNew3(:,:,unique(I2(init,:)));
end

%{
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Remove models that final point exceeds 100 (or is NaN)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Rows = [];

k=1;
    for c = 1:size(AICvals2,2)
         Q = 0;
         if c==421
             P=[];
         end
            for c2 = 1:size(AICvals2,1)
                if isnan(AICvals2(c2,c)) == 0 || CT2(end,1,c2) > 100 %|| CT2(end,2,c2)>100 || CT2(end,3,c2)>100 || CT2(end,4,c2)>100 || CT2(end,5,c2)>100 || C
                    Q=Q+1;
                end
            end

            if Q == size(InitTest,1)
                    Rows = [Rows;c];
                    k=k+1;
            end
    end
Rows = unique(Rows);

KStoreNew = KStoreNew3(:,:,Rows);
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Plot remaining Kinetics to remove models that exceed 10 and/or

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   for init = 1:size(InitTest,1)
       %init=1;

        for k = 1:size(KKKStore,3)

            CT3(1,:,k) = InitTest(init,:);
            C0 = InitTest(init,:);

                for t2 = 2:size(NMeasurements,1)
                    combo2(1) = 1;j=2;
                        for c = 1:NoComponents
                            combo2(j)= C0(c);
                                j=j+1;
                            for cc = c:NoComponents
                                combo2(j) = C0(c)*C0(cc);
                                j=j+1;
                            end
                        end

                            for c = 1:NoComponents
                                dC_dt3(t2,c,k) = KKKStore(c,:,k)*tvar(tt)*combo2';
                                CT3(t2,c,k) = CT3(t2-1,c,k)+dC_dt3(t2,c,k);
                            end
                    C0 = CT3(t2,:,k);
                end
                    for c = 1:NoComponents

                            if CT3(t2,c,k) >= 10 || CT3(t2,c,k) <= -0.5 || isnan(CT3(t2,c,k))
                                CT3(:,:,k) = zeros(t2,NoComponents);
                                KKKStore(:,:,k)=zeros(NoComponents,NoCombos);

                            end

                    end


                %   figure()
                %   plot(CT3(:,:,k))


        end

   end
   %{
   for c = 1:NoComponents
            figure(c)
           plot(CxT(:,c,1))
           hold on


   for k = 1:size(KStoreNew,3)
   plot(CT3(:,c,k))
   end
      end
   %}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Models with ODE45

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

minerror = [];
minvalue=[];

K2New=[];
K22 = 1;

for i3 = 1:size(KKKStore,3)
    Q = 0;
    for i2 = 1:size(KKKStore,1)
            if nnz(KKKStore(i2,:,i3))<= NoComponents
               Q=Q+1;
             end
    end

    if Q == NoComponents
        K2New(:,:,K22) = KKKStore(:,:,i3);
                K22 = K22+1;
    end
end


KKKStore = K2New;

for i = 1:size(InitTest,1)
```

```
 C0 = InitTest(i,:);

    for k = 1:size(KKKStore,3)

        NUseful = [];KID = [];
        for m = 1:size(KKKStore,2)
             Q = sum(abs(KKKStore(:,m,k)));
            if Q>=0.01
                NUseful =[NUseful,m];
                KID =[KID,KKKStore(:,m,k)];
            end

            if nnz(KKKStore(:,m,k)) == 1 && isempty(KID)==1
               KID = [];
            elseif nnz(KKKStore(:,m,k)) == 1
                KID(:,size(KID,2))= zeros(NoComponents,1);
            end

        end

          if isempty(KID) || size(NUseful,2) >= NoComponents
                NUseful = [1];
                KID = zeros(NoComponents,1);
          end


    [tID2,CxID2(:,:,k)] = ode45(@(tspan,C0)biodieselratesKin(tspan,C0,KID,NUseful),tspan,C0);

    for c = 1:NoComponents
        if CxID2(end,c,k) < -0.2
            CxID2(:,:,k) = zeros(size(tspan,2),NoComponents);
        end
    end

    end



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

Relative AIC to find the best full model

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    [AICvals3(i,:)] = AICerrorsFull(CxT(:,:,i),CxID2(:,:,:),KKKStore);
```

```
end


TotalAIC = [];
for i = 1:size(AICvals3,2)
    TotalAIC(1,i) = sum(AICvals3(:,i));
end

minerror = find(TotalAIC(1,:) == min(TotalAIC(1,:)));


minvalue = TotalAIC(1,minerror);

 if numel(minvalue) > 1
    minvalue = minvalue(1);
 end

for i = 1:size(AICvals3)
RelAIC2(1,i) = exp(minvalue-TotalAIC(1,i));
end

         [Possiblex,Possibley] = find(RelAIC2(1,:)>0.01);
         Possibley = unique(Possibley);



Best = find(RelAIC2==max(RelAIC2));

KIDed = KKKStore(:,:,Best);

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

Validation of models

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
InitVal = [2,2,0,0,0,0];

for i = 1:size(InitVal,1)
    InitV = InitVal(i,:);

[TVal,DatasetV(:,:,i)] = ode45(@(tspan,InitV)biodieselratesKin(tspan,InitV,Kinetics1,Useful1),tspan,InitV);


for c = 1:size(InitTest,2)
    RangeC = (max(DatasetV(:,c,i)-min(DatasetV(:,c,i))))*Percent;
    awgnNoise = randn(size(DatasetV(:,c,i)))*RangeC;
    sumoferroror = sumoferror+RangeC;

    YNoise(:,c) = DatasetV(:,c,i) + awgnNoise;
    YNoise(1,c) = DatasetV(1,c,i);
end
YNoise(YNoise(:,:) < 0) =0;
NMeasurementsV(:,:) = YNoise(:,:,1);
```

```
DatasetV2(:,:,i) = [TVal,NMeasurementsV(:,:,i)];
end


%[KReOptim] = ReOptimFun(DatasetV2(:,:,1),KIDed);

for c = 1:size(Possibley,1)
Used = [];
KIDd = [];
        for c2 = 1:size(KIDed,2)
    Q = sum(abs(KIDed(:,c2)));
            if Q > 0.01
                Used = [Used,c2];
                KIDd = [KIDd,KIDed(:,c2)];
            end

        end

    for c2 = 1:size(KIDd,2)
                if nnz(KIDd(:,c2)) < 2 && isempty(KIDd) == 0
                    KIDd(:,c2)= zeros(NoComponents,1);
                end
    end
    if isempty(KIDd) && isempty(Used)
       KIDd = zeros(NoComponents,1);
       Used = 1;
    end
    [TVal2,Val(:,:,c)] = ode45(@(tspan,InitV)biodieselratesKin(tspan,InitV,KIDd,Used),tspan,InitV);

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%




for c = 1:NoComponents
   figure(c)
   plot(TVal,DatasetV2(:,c+1,1),'x')
    hold on
    plot(TVal2,Val(:,c,1))
    legend('Validation Data','ID Model')
    xlabel('Time, hrs')
    ylabel('Concentration, mol.dm^-^3')

    switch c
        case 1
            title('Change in Concentration of Triglyceride')
         %   annotation('textbox',[0.4 0.5 0.3 0.3],'String','d[TG]/dt = -0.354[TG][MeOH]','FitBoxToText','on')
        case 2
            title('Change in Concentration of Methanol')
          %  annotation('textbox',[0.3 0.5 0.3 0.3],'String','d[MeOH]/dt = -0.406[TG][MeOH]-0.511[DG][MeOH]-0.247[MG][MeOH]','FitBoxToText','on')
        case 3
            title('Change in Concentration of Diglyceride')
           % annotation('textbox',[0.4 0.5 0.3 0.3],'String','d[MeOH]/dt = -0.406[TG][MeOH]-0.511[DG][MeOH]-0.247[MG][MeOH]','FitBoxToText','on')
        case 4
            title('Change in Concentration of BioDiesel')
          %  annotation('textbox',[0.25 0.125 0.3 0.3],'String','d[MeOH]/dt = -0.406[TG][MeOH]-0.511[DG][MeOH]-0.247[MG][MeOH]','FitBoxToText','on')

        case 5
             title('Change in Concentration of Monoglyceride')
           % annotation('textbox',[0.25 0.125 0.3 0.3],'String','d[MeOH]/dt = -0.406[TG][MeOH]-0.511[DG][MeOH]-0.247[MG][MeOH]','FitBoxToText','on')

        case 6
            title('Change in Concentration of Glycol')
           % annotation('textbox',[0.4 0.1 0.3 0.3],'String','d[MeOH]/dt = -0.406[TG][MeOH]-0.511[DG][MeOH]-0.247[MG][MeOH]','FitBoxToText','on')
    end
    %}
end

%{
%[KReOptim] = ReOptimFun(DatasetV2,Storage);
KReDone = zeros(size(Storage));


for u = 1:length(Useful)
KReDone(:,Useful(u)) = KReOptim(:,u);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Error of Identified, reoptimised  model.
C0 = Measurements(1,1:6);

Kin2 = KReDone(:,[Useful]);

[tVal,CVal] = ode45(@(tspan,InitVal)biodieselratesKin(tspan,InitVal,Kin2,Useful),tspan,InitVal);

SqErr = [];
if length(CVal(:,1))~= length(Measurements(:,1))
    SqErr = 1000*ones(length(Measurements(:,1)),NoComponents);
   AbsErr = 1000*ones(length(Measurements(:,1)),NoComponents);

else
    for i = 1:size(CVal,2)
%{
    figure(i)
plot(MeasurementsTime(:,1),Measurements(:,i))
hold on
plot(MeasurementsTime(:,1),Cxk(:,i))
legend('Measurements','Identified')
%}
        for t = 1:size(MeasurementsTime,1)
        SqErr(t,i) = (Measurements(t,i)-CVal(t,i))^2;
        AbsErr(t,i) = (abs(Measurements(t,i)-CVal(t,i)));
        end
    end
end
%}


    Model= KIDed(:,:,1);
    for timed = 1:size(TVal2,1)
        for c = 1:size(Val,2)
        AbsError(timed,c) = abs(Val(timed,c)-DatasetV2(timed,c+1));
        end
    end
%SumAbsErr(m,1:NoComponents) = sum(AbsErr,1);
%SumSqErr(m,1:NoComponents) = sum(SqErr,1);

 %  if tt == 1
   MC{rrr,tt,jjj}.Model = Model;
   %MC{i,jjj}.AbsErr = SumAbsErr(:,i);
   %MC{i,jjj}.SqErr = SumSqErr(:,i);
   MC{rrr,tt,jjj}.t = size(tspan,2);
   MC{rrr,tt,jjj}.Noise = PercentNoise(jjj);
   MC{rrr,tt,jjj}.AbsError = [TVal2,AbsError];
   MC{rrr,tt,jjj}.ValData = [DatasetV2];
   MC{rrr,tt,jjj}.ModeledData = [TVal2,Val];
  %{
else
   MC{i+(tt-1)*NoComponents,jjj}.Model = Model(:,:,i);
   %MC{i+(tt-1)*NoComponents,jjj}.AbsErr = SumAbsErr(:,i);
   %MC{i+(tt-1)*NoComponents,jjj}.SqErr = SumSqErr(:,i);
   MC{i+(tt-1)*NoComponents,jjj}.t = size(tspan,2);
   MC{i+(tt-1)*NoComponents,jjj}.Noise = PercentNoise(jjj);
   end
   %}

   Denom = repeats*size(tvar,2);
   NUMBER = sprintf('%d / %d completed',qqq, Denom);
   disp(NUMBER)
   qqq=qqq+1;

%}
```

```
end
end
end

save(fullfile(pathname,version),'MC')
% Repeat settings and store models & error for 100 iterations to see effect of noise & limited data points.
%{
IDModels = cell(length(tvar),length(PercentNoise));
for t = 1:length(tvar)
   for n = 1:length(PercentNoise)
        for c = 1:NoComponents
       IDModels{t,n}.Models(c,:) = MC{t+c-1,n}.Model;
       IDModels{t,n}.AbsErr(c,:) = MC{t+c-1,n}.AbsErr;
       IDModels{t,n}.SqErr(c,:) = MC{t+c-1,n}.SqErr;
        end
        IDModels{t,n}.TimePoints(c,:) = MC{t,n}.t;
        IDModels{t,n}.NoisePercent(c,:) = MC{t,n}.Noise;
   end
end






 %{
figure(6)
title({'Change in Concentration of Glycol','26 Data points, 0% Noise'})
figure(5)
title({'Change in Concentration of MonoGlyceride','26 Data points, 0% Noise'})
figure(4)
title({'Change in Concentration of BioDiesel','26 Data points, 0% Noise'})
figure(3)
title({'Change in Concentration of DiGlyceride','26 Data points, 0% Noise'})
figure(2)
title({'Change in Concentration of Methanol','26 Data points, 0% Noise'})
figure(1)
title({'Change in Concentration of TriGlyceride','26 Data points, 0% Noise'})
%}
%}
```

Contents

```
function [dC_dt, combo, DeltaC, Integral, comboname, Timeint]=datatransform(Concentrations, Time, componentname, dCmethod)
```

```
global NoComponents

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

Timeint

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Timeint = Time(2) - Time(1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
Not enough input arguments.

Error in datatransform (line 10)
Timeint = Time(2) - Time(1);
```

Combos

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
combo(:,1) = ones(length(Concentrations(:,1)),1);
k=2;
for i = 1:NoComponents
    combo(:,k)=Concentrations(:,i);
    comboname(k) = componentname(i);
    k=k+1;
    for j = i:NoComponents
        combo(:,k) = Concentrations(:,i).*Concentrations(:,j);
        comboname(k) = strcat(componentname(i),componentname(j));
        k = k+1;
    end
end

NoCombos = length(comboname);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

TrapzCombos & DeltaC

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
switch dCmethod
    case 1 % trapz

 for t = 1:length(Concentrations(:,1))-1
           for c = 1:NoComponents
               DeltaC(t,c) = (Concentrations(t+1,c)-Concentrations(t,c))/Timeint;
           end
       for c = 1:NoCombos
       Integral(t,c) =  trapz(combo(t:t+1,c));
       end
 end

    case 2 % Rectangle Rule (Reimann sum)
       % Timeint = (Time(2)-Time(1))/2;
   for t = 1:length(Concentrations(:,1))-1
       for c = 1:NoComponents
               DeltaC(t,c) = (Concentrations(t+1,c)-Concentrations(t,c))/Timeint;

       end
```

```
        for c = 1:NoCombos
        Integral(t,c) = Timeint*(combo(t,c));
        end

    end

       case 3 % Simpson's 3/8 rule
          % Timeint = (Time(2)-Time(1))/3;
        for t = 1:length(Concentrations(:,1))-1

            for c = 1:NoComponents
            DeltaC(t,c) = (Concentrations(t+1,c)-Concentrations(t,c))/Timeint;
            end

            for c = 1:NoCombos
            if t==1  || t==length(combo(:,1)-1)
                n = 1;
            elseif mod(t,2)==1
                n = 3;
            elseif mod(t,2)==0
                n=3;
            end

        Integral(t,c) = (3*Timeint/8)*n*combo(t,c);


            end
        end
     case 4 % Spline
        % Timeint = (Time(2)-Time(1))/4;

        for c = 1:NoCombos

            pp = spline(Time(:,1),combo(:,c));
            p_int = fnint(pp,1);


                 Integral1(:,c) = fnval(p_int,Time);

         for t = 1:length(Time)-1
            Integral2(t,c) = ppval(p_int,Time(t));
         end


%for t = 1:length(Time)-1
      %  dC_dt(t,c) = ppval(p_der,Time(t));
      %  end
        end

        Integral = Integral1;

    for t = 2:length(Concentrations(:,1))-1
        for c = 1:NoComponents
                DeltaC(t,c) = (Concentrations(t+1,c)-Concentrations(t,c))/Timeint;

        end
    end


end

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

dC_dt

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Below is Euler's method, a standard method and is used as a staple in the
% rest of the
     for t = 1:length(Time)-1
         for c = 1:NoComponents
             dCEuler(t,c) = (Concentrations(t+1,c) - Concentrations(t,c))/Timeint;
         end
     end
```

```
%  figure(1);plot(Concentrations(:,1),Concentrations(:,2));title('Concentrations')
%  figure(2);plot(Concentrations(1:end-1,1),dCEuler(:,1));title('dC_dt')

  switch dCmethod

      case 1 % Forward Euler Method
  dC_dt = dCEuler;

      case 2 % Central Euler
          dC_dt(1,1:NoComponents) = dCEuler(1,:);
  for t = 2:length(Time)-1

      for c = 1:NoComponents

          dC_dt(t,c) = (0.5*Concentrations(t+1,c)-0.5*Concentrations(t-1,c))/(2*Timeint);

      end
  end
          dC_dt(t+1,1:NoComponents) = zeros(1,NoComponents);

  case 3 % Taylor Series
     for t = 1:length(Time)-1
        for c = 1:NoComponents
           if t==1 ||t==t(end)
               dC_dt(t,c)=dCEuler(t,c);
               continue
           end
            dC2(t,c) = (dCEuler(t+1,c)-2*dCEuler(t,c)-dCEuler(t-1,c))/(Timeint^2);

            dC_dt(t,c) = dCEuler(t,c)+((Timeint^2)/2)*dC2(t,c);
        end

     end


  case 4 % Spline based
      for c = 1:NoComponents

      pp = spline(Time,Concentrations(:,c));
      p_der = fnder(pp,1);
      for t = 1:length(Time)-1
      dC_dt(t,c) = ppval(p_der,Time(t));
      end
      end
  end
```

```
end
```

Published with MATLAB® R2020a

```
function [AICvalues] = AICerrorsFull(CxT, CxID, KineticStore)
```

This Version is to find a single AIC value for the whole model... problems.

```
Rowsoftime = size(CxID,1);

for r = 1:size(CxID,3)
    k = nnz(KineticStore(:,:,r));
    for c = 1:size(CxT,2)



error(:,c) = abs(CxT(:,c) - CxID(:,c,r));

sumerror(r,c) = sum(error(:,c));

    end
ssumerror(r) = sum(sumerror(r,:));
AICvalues (r) = Rowsoftime*log(ssumerror(r)/Rowsoftime)+2*k;


end
```

```
Not enough input arguments.

Error in AICerrorsFull (line 7)
Rowsoftime = size(CxID,1);
```

```
end
```

Published with MATLAB® R2020a

References

ADAMSON, R., HOBBS, M., SILCOCK, A. & WILLIS, M. J. 2017. Steady-state optimisation of a multiple cryogenic air separation unit and compressor plant. *Applied Energy,* 189**,** 221-232.

AMRHEIN, M., SRINIVASAN, B. & BONVIN, D. 1999. Target factor analysis of reaction data: use of data pre-treatment and reaction-invariant relationships. *Chemical Engineering Science,* 54**,** 579-591.

ARIS, R. & MAH, R. H. S. 1963. Independence of Chemical Reactions. *Industrial & Engineering Chemistry Fundamentals,* 2**,** 90-94.

BENICHOU, M., GAUTHIER, J. M., GIRODET, P., HENTGES, G., RIBIERE, G. & VINCENT, O. 1971. Experiments in mixed-integer linear programming. *Mathematical Programming,* 1**,** 76-94.

BLAU, G., MEHTA, B., BOSE, S., PEKNY, J., SINCLAIR, G., KEUNKER, K. & BUNCH, P. 2000. Risk management in the development of new products in highly regulated industries. *Computers and Chemical Engineering,* 24**,** 659-664.

BONVIN, D. & RIPPIN, D. W. T. 1990. Target factor analysis for the identification of stoichiometric models. *Chemical Engineering Science,* 45**,** 3417-3426.

BRENDEL, M., BONVIN, D. & MARQUARDT, W. 2006. Incremental identification of kinetic models for homogeneous reaction systems. *Chemical Engineering Science,* 61**,** 5404-5420.

BURNHAM, S. C., SEARSON, D. P., WILLIS, M. J. & WRIGHT, A. R. 2008. Inference of chemical reaction networks. *Chemical Engineering Science,* 63**,** 862-873.

CAI, A., TSAY, R. S. & CHEN, R. 2009. Variable selection in linear regression with many predictors. *Journal of computational and graphical statistics,* 18**,** 573-591.

CHAPRA, S. C., CANALE, R.P. 2015. *Numerical Methods for Engineers,* New York, USA, McGraw-Hill Education.

FILIPPI, C., GREFFE, J. L., BORDET, J., VILLERMAUX, J., BARNAY, J. L., BONTE, P. & GEORGAKIS, C. 1986. Tendency modeling of semibatch reactors for optimization and control. *Chemical Engineering Science,* 41**,** 913-920.

FLOUDAS, C. A. & LIN, X. 2005. Mixed Integer Linear Programming in Process Scheduling: Modeling, Algorithms, and Applications. *Annals of Operations Research,* 139**,** 131-162.

FOGLER, H. S. 1999. *Elements of chemical reaction engineering,* Upper Saddle River, NJ, Upper Saddle River, NJ : Prentice Hall PTR.

FOTOPOULOS, J., GEORGAKIS, C. & STENGER, H. G. Structured target factor analysis for the stoichiometric modeling of batch reactors. Proceedings of 1994 American Control Conference - ACC '94, 29 June-1 July 1994 1994. 495-499 vol.1.

FOTOPOULOS, J., GEORGAKIS, C. & STENGER, H. G. 1995. Effect of Model Uncertainty on the Tendency Modeling, Optimization and Control of Batch Reactors. *IFAC Proceedings Volumes,* 28**,** 425-431.

FOTOPOULOS, J. E. A. 1998. Use of tendency models and their uncertainty in the design of estimators for bathc reactors. *chemical engineering and processing: process intensification,* 37**,** 545-558.

FUGUITT, R. E. & HAWKINS, J. E. 1945. The Liquid Phase Thermal Isomerization of α-Pinene1a, 1b. *Journal of the American Chemical Society,* 67**,** 242-245.

FUGUITT, R. E. & HAWKINS, J. E. 1947. Rate of the thermal isomerization of α-Pinene in the liquid phase1. *Journal of the American Chemical Society,* 69**,** 319-322.

GONÇALVES, I. & SILVA, S. 2013. Balancing Learning and Overfitting in Genetic Programming with Interleaved Sampling of Training Data. *In:* KRAWIEC, K., MORAGLIO, A., HU, T., ETANER-UYAR, A. Ş. & HU, B. (eds.) *Genetic Programming: 16th European Conference, EuroGP 2013, Vienna, Austria, April 3-5, 2013. Proceedings.* Berlin, Heidelberg: Springer Berlin Heidelberg.

GROSSMANN, I. E. & SANTIBANEZ, J. 1980. Applications of mixed-integer linear programming in process synthesis. *Computers & Chemical Engineering,* 4**,** 205-214.

HAIKARAINEN, C., PETTERSSON F., SAXEN H., 2013. An MILP Model for Distributed Energy System Optimisation. *Energy System Optimization, Chemical Engineering Transactions, 35, 295-300.*

HII, C. J. K., WRIGHT, A. R. & WILLIS, M. J. 2014. Utilizing a genetic algorithm to elucidate chemical reaction networks: An experimental case study. *International Journal of Chemical Engineering and Applications,* 5**,** 516.

HOFF, J. H. V. T. 1884. *Etudes de dynamique chimique,* Amsterdam, Frederik Muller & Co.

HRISTAKEVA, M. & SHRESTHA, D. 2004. Solving the 0-1 Knapsack Problem with Genetic Algorithms.

KOZA, J. R. 1992. *Genetic programming : on the programming of computers by means of natural selection,* Cambridge, Mass., Cambridge, Mass. : MIT Press.

LAING, H., O'MALLEY, C., BROWNE, A., RUTHERFORD, T., BAINES, T. & WILLIS, M. J. 2020. Development of a biogas distribution model for a wastewater treatment plant: a mixed integer linear programming approach. *Water Science and Technology,* 82**,** 2761-2775.

LANGARY, D. & NIKOLOSKI, Z. 2019. Inference of chemical reaction networks based on concentration profiles using an optimization framework. *Chaos: An Interdisciplinary Journal of Nonlinear Science,* 29**,** 113121.

LANGDON, W. B. 2011. Minimising testing in genetic programming. *Research Note,* 11**,** 1.

LAUINGER, D., CALIANDRO, P., VAN HERLE, J. & KUHN, D. 2016. A linear programming approach to the optimization of residential energy systems. *Journal of Energy Storage,* 7**,** 24-37.

LIMA, R. M. & GROSSMANN, I. E. 2011. Computational advances in solving Mixed Integer Linear Programming problems.

LUBIN, M., YAMANGIL, E., BENT, R. & VIELMA, J. P. 2018. Polyhedral approximation in mixed-integer convex optimization. *Mathematical Programming,* 172**,** 139-168.

MAIA, M. H. & GALVÃO, R. K. H. 2009. On the use of mixed-integer linear programming for predictive control with avoidance constraints. *International Journal of Robust and Nonlinear Control,* 19**,** 822-828.

MARIA, G. 2004. A Review of Algorithms and Trends in Kinetic Model Identification for Chemical and Biochemical systems. *Chemical and Biochemical Engineering Quarterly,* 18**,** 195-222.

MCKAY, B., WILLIS, M. & BARTON, G. 1997. Steady-state modelling of chemical process systems using genetic programming. *Computers & Chemical Engineering,* 21**,** 981-996.

MITCHELL, M. 1995. Genetic Algorithms: An Overview. *Complexity,* 1.

NASH, J. C. 2000. The (Dantzig) simplex method for linear programming. *Computing in Science & Engineering,* 2**,** 29-31.

NEUMANN, P., CAO, L., RUSSO, D., VASSILIADIS, V. & LAPKIN, A. 2019. A new formulation for symbolic regression to identify physico-chemical laws from experimental data. *Chemical Engineering Journal,* 387.

OPFERMANN, J. 2000. Kinetic Analysis Using Multivariate Non-linear Regression. I. Basic concepts. *Journal of Thermal Analysis and Calorimetry,* 60**,** 641-658.

PATSIATZIS, D. I., KNIGHT, G. & PAPAGEORGIOU, L. G. 2004. An MILP Approach to Safe Process Plant Layout. *Chemical Engineering Research and Design,* 82**,** 579-586.

POTVIN, J.-Y. 1996. Genetic algorithms for the traveling salesman problem. *Annals of Operations Research,* 63**,** 337-370.

PRINZ, O. & BONVIN, D. 1994. Monitoring Chemical Reaction Systems Using Incremental Target Factor Analysis. *IFAC Proceedings Volumes,* 27**,** 339-344.

RADMANESH, M., KUMAR, M., NEMATI, A. & SARIM, M. Solution of Traveling Salesman Problem with Hotel Selection in the framework of MILP-tropical optimization. 2016 American Control Conference (ACC), 6-8 July 2016 2016. 5593-5598.

RAMAN, R. & GROSSMANN, I. E. 1991. Relation between MILP modelling and logical inference for chemical process synthesis. *Computers & Chemical Engineering,* 15**,** 73-84.

RAVI, R., VINU, R. & GUMMADI, S. N. 2017. *Coulson and Richardson's chemical engineering. Volume 3A, Chemical and biochemical reactors and reaction engineering*, Kidlington, Oxford, United Kingdom : Butterworth-Heinemann is an imprint of Elsevier.

RICHARDS, A. & HOW, J. P. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301), 8-10 May 2002 2002. 1936-1941 vol.3.

WALLER, K. V. & MAKILA, P. M. 1981. Chemical reaction invariants and variants and their use in reactor modeling, simulation, and control. *Industrial & Engineering Chemistry Process Design and Development,* 20**,** 1-11.

WILLIS, M. J. & STOSCH, M. V. 2016. Inference of chemical reaction networks using mixed integer linear programming. *Computers & Chemical Engineering,* 90**,** 31-43.

YANG, Q., SING-LONG, C. A. & REED, E. J. 2020. Rapid data-driven model reduction of nonlinear dynamical systems including chemical reaction networks using ℓ1-regularization. *Chaos: An Interdisciplinary Journal of Nonlinear Science,* 30**,** 053122.

RAMAN, R. & GROSSMANN, I. E. 1991. Relation between MILP modelling and logical inference for chemical process synthesis. *Computers & Chemical Engineering,* 15**,** 73-84.

TRESPALACIOS, F. & GROSSMANN, I. 2014. Review of Mixed-Integer Nonlinear and Generalized Disjunctive Programming Methods. *Chemie Ingenieur Technik,* 86.

TUMBALAM GOOTY, R., AGRAWAL, R. & TAWARMALANI, M. 2019. An MINLP Formulation for the Optimization of Multicomponent Distillation Configurations. *Computers & Chemical Engineering,* 125.

YE, Y., GROSSMANN, I. E. & PINTO, J. M. 2018. Mixed-integer nonlinear programming models for optimal design of reliable chemical plants. *Computers & Chemical Engineering,* 116**,** 3-16.