UNIVERSITÀ DEGLI STUDI DI PADOVA
DIPARTIMENTO DI INGEGNERIA INDUSTRIALE
CORSO DI LAUREA MAGISTRALE IN INGEGNERIA CHIMICA E DEI PROCESSI INDUSTRIALI

Tesi di Laurea Magistrale in
Ingegneria Chimica e dei Processi Industriali

Optimal design of experiments for kinetic model structure identification using reinforcement learning methodologies

Relatore: Prof. Fabrizio Bezzo
Correlatore: Prof. Federico Galvanin

Laureando: ANDREA FRISO

ANNO ACCADEMICO 2020 – 2021

Abstract

Mathematical models are widely used in many fields to describe physical phenomena that are pivotal for many applications. In particular, in the chemical engineering field, mathematical models are used to describe chemical reactions and processes. Kinetic models are used to describe the rate at which chemical reactions occur. However, choosing the most appropriate kinetic model for a specific application is still a challenging problem. An approach that can be used to discriminate among different models is Model-Based Design of Experiments (MBDoE). The objectives of this work are two:

- propose an alternative methodology to identify simultaneously the set of model equations and the value of kinetic model parameters.
- to couple this methodology with Reinforcement Learning. Reinforcement Learning techniques merge on the concept of learning from experience, where an agent, learns interacting with the environment.

This methodology is tested on a case study identifying the reaction kinetics in a bioreactor model. The first part of the case study identifies the model and there is the evaluation of the parameters and in the second part there is the extension of an RL technique to this case to optimize a statistic of this method.
The results obtained demonstrate that this method is capable of identifying the structure of the model and the parameters with very good precision under a wide range of conditions of measurements and noise.
In the second part of the thesis, the proposed is coupled with a reinforcement learning methodology. The aim of this pairing is to identify the optimum conditions under which to carry out the experiment.

Riassunto Esteso

Negli ultimi anni con la digitalizzazione dell'industria i modelli matematici hanno acquisito sempre più importanza. I modelli matematici hanno svariati campi di applicazione, dalla medicina all'ingegneria di processo, e vengono utilizzati per descrivere dei sistemi fisici.
Nell'ingegneria chimica in particolare questi modelli vengono utilizzati per descrivere una vasta gamma di processi. Un ruolo molto importante lo hanno i modelli cinetici, che sono i modelli che vengono utilizzati per rappresentare le reazioni chimiche. I modelli cinetici al giorno d'oggi sono fondamentali per diversi aspetti; infatti, utilizzandoli è possibile valutare la velocità di produzione di prodotti, la velocità di consumo dei reagenti e inoltre vengono utilizzati per progettare i reattori da utilizzare, per calcolare il quantitativo di catalizzatore necessario per il decorso della reazione e soprattutto per ottimizzare l'intero processo.
Il problema alla base di questi modelli è la loro selezione. Infatti il processo di selezione di un modello matematico è un processo molto complesso e solitamente molto costoso in quanto sono solitamente necessari diversi esperimenti preliminari per raccogliere un numero sufficiente di dati. Solitamente per caratterizzare un esperimento vengono proposti diversi modelli e quindi è necessario effettuare un processo di discriminazione dei modelli proposti. Nella procedura classica il primo passaggio proposto è una analisi di identificabilità dei parametri in quanto in alcuni casi i modelli sono strutturati in maniera tale che l'identificazione dei parametri al loro interno non è possibile. I modelli che non superano questo test devono essere riformulati in maniera differente. Con i modelli che superano il test di identificabilità si procede con la regressione dei parametri effettuata su dati ottenuti da misure sperimentali. Se più di un modello supera questo secondo step è necessario procedere con una ulteriore discriminazione. Per questa procedura sono state sviluppate diverse tecniche che tengono in considerazione la complessità del modello, che viene solitamente data dal numero di parametri che compaiono all'interno del modello. La procedura classica è una procedura iterativa in quanto i diversi passaggi vengono ripetuti fino a quando non rimane un solo modello adatto a descrivere il sistema. Col passare degli anni questo metodo si è evoluto in modo da ridurre il numero di esperimenti da effettuare per individuare il modello. Infatti tramite la progettazione degli esperimenti basata sul modello (model based design of experiment o MBDoE) è stato possibile pianificare gli esperimenti da condurre in maniera tale da massimizzare il quantitativo di informazioni dato dal singolo esperimento. Negli ultimi anni con il progredire della tecnologia nuovi metodi sono stati sviluppati. Ad esempio Quaglio et al. (2020) hanno proposto un approccio basato sull'utilizzo di reti neurali artificiali. Le reti neurali artificiali sono un particolare tipo di modelli matematici appartenenti al machine learning che prendono ispirazione dalle reti neurali biologiche. Questo tipo di algoritmi utilizza i dati per allenare le reti neurali effettuando esperimenti in-silico. Inoltre utilizzando le reti neurali non si ottiene un vero e proprio modello matematico ma sono le

reti neurali stesse l'approssimazione del modello.
Tutti i metodi proposti adottano un processo iterativo in cui prima vanno a valutare la struttura del modello e solo successivamente identificano i parametri che lo compongono. In questo lavoro di tesi l'obiettivo è quello di proporre una nuova metodologia che sia in grado di identificare simultaneamente sia la struttura del modello cinetico che i parametri che lo compongono. Inoltre in questo metodo differentemente dalle reti neurali si vuole andare a vedere la struttura del modello cinetico identificato. La tecnica sviluppata si suddivide in diversi passaggi:

- creazione di un dataset contenente i dati ottenuti da misure sperimentali (in questo caso da esperimenti in-silico);
- identificazione del modello e dei parametri cinetici;
- dopo aver individuato il modello e i parametri vengono calcolate le statistiche sia sul modello che sui parametri in modo tale da avere informazioni sulla loro adeguatezza.

Utilizzando questo metodo, quindi, è possibile andare ad individuare il modello cinetico più adatto a descrivere i dati raccolti nel primo step. L'aspetto fondamentale di questa prima parte é l'utilizzo di un metodo chiamato *supestructure method* con cui partendo da una struttura matematica molto complessa si riesce tramite un particolare tipo di ottimizzazione a creare la struttura del modello e allo stesso tempo ad identificare i parametri.
Il secondo obiettivo di questo lavoro è quello di accoppiare la metodologia proposta in precedenza con un algoritmo di Reinforcement Learning (RL). Il RL è un particolare ramo del machine learning basato sull'apprendimento automatico. L'obiettivo di un algoritmo di RL è quello di andare a generare un "agente automatico" in gradi di scegliere la migliore azione da compiere in base all'ambiente che lo circonda. Tutti questi algoritmi, infatti, si basano sullo sviluppo di una politica che l'agente deve adottare in modo tale da massimizzare una ricompensa (o funzione obiettivo) che varia a seconda dell'azione compiuta. L'accoppiamento delle tecniche di RL e la tecnica di individuazione del modello potrebbe portare a diversi vantaggi. Infatti utilizzando le tecniche di apprendimento proposte dal RL si possono modificare le condizioni degli esperimenti e/o i parametri del modello in modo tale da massimizzare delle statistiche del modello. Questo sarebbe molto utile in quanto permetterebbe di andare ad indentificare in maniera più rapida le condizioni ottimali con cui eseguire gli esperimenti in modo tale da massimizzare le informazioni ottenute. I risultati ottenuti nella prima parte della tesi sono molto promettenti; infatti, la metodologia proposta è stata in grado di identificare correttamente il modello cinetico adeguato a diverse condizioni (ad esempio con un livello di rumore nelle misure del dataset molto elevate o con un basso numero di dati nel dataset). La seconda parte invece è stata quella più complicata da sviluppare. La tecnica di RL utilizzata infatti ha apprendimento abbastanza limitato e principalmente si limita ad esplorare lo spazio sperimentale e ad indicare quali sono le condizioni migliori trovate. Questo progetto prevede molteplici sviluppi futuri. Infatti si prevede di migliorare la tecnica proposta nella prima parte aggiungendo una parte in

cui si va modificare il modificare il modello in maniera mirata in modo tale da migliorare le sue prestazioni. Inoltre la parte di accoppiamento con il RL è necessario andare a migliorare la tecnica proposta andando ad aggiornare dopo ogni iterazione il valore dei parametri stimati. .

Contents

Introduction

Chemical processes can be described with a mathematical model, usually expressed in the form of a system of Differential and Algebraic Equations (DAE). In particular, in a reaction process, where these models aim to describe perfectly what happens during the reaction, the model has to be as accurate as possible. These models, called kinetic models, describe the rate of reactions, and the reaction rate is affected by several factors such as the concentration of the reactants, the temperature of the reactor, the used catalyst and any other conditions that affect the reaction. These models are widely used in the chemical engineering world for several tasks for example designing the reactors, projecting the chemical processes and especially for the optimizations tasks. Kinetic models are additionally used, mostly in the pharmaceutical field, to understand a reaction and in this way, the experimenter can achieve a deeper knowledge of the process.
The main difficulty is that kinetic models are extremely case sensitive and can be described by very different mathematical expressions. For this reason when a chemical reaction is studied it is necessary to identify the correct model, i.e. a model that can represent adequately the data obtained during the experimental measurements. The main objective of this work is to automate the process of model identification and parameter estimation exploiting the concepts of Reinforcement Learning (Johnson et al., 2000). With these methodologies, belonging to the field of artificial intelligence, algorithms will explore different experimental conditions in the experimental design space and will determine what are the most appropriate experimental conditions to maximize some statistics of the model.

This thesis is structured in the following way:

- Chapter 1: in this first chapter an overview is given on mathematical modelling including model-based design of experiments (MBDoE) and Reinforcement Learning (RL) frameworks. Moreover, some techniques suitable for the identification of the model structure are reported.

- Chapter 2: the main topic of this section is the explanation of the proposed method for the simultaneous identification of model equations and kinetic parameters. All the mathematical expressions needed to understand how the method work are reported.

- Chapter 3:the first case study, concerning the identification of kinetic models in a bioreactor, is presented. In this chapter the experimental conditions are not optimized but are held constant. A superstructure-based approach is used to simultaneously identify the model equations and the set of model parameters. Together with the results also some considerations about the Python implementation are reported.

- Chapter 4: the new methodology (presented in Chapter 2) combining superstructure optimization and RL is applied to the case study analysed in Chapter 3.

In the last part of the thesis the conclusions are summarized and some hypotheses for some future developments to improve this method are proposed.

Chapter 1

Dynamic Model Identification and Parameter Estimation

In recent years the utilization of mathematical models to describe chemical or physical phenomena has become increasingly common. With the advent of new technologies and the digitisation of industries, the amount of data has grown exponentially and consequently also the tools used to interpret the data have increased rapidly. Mathematical models are widely used in many different fields, from natural sciences to economics and engineering applications and also for medical purposes. For example, Rosenberg (2021)Rosenberg, 2021, used mathematical models to predict some characteristics of COVID-19 (geographic spread of the disease, the temporal course of numbers of cases and fatalities, the load on health systems, and the effects of policy measures). Mathematical models have been used also to correlate the measured toxicity in the environment with the population level in that environment (Michel, 2021). Mathematical models are also widely used for applications of an economic nature, for example Karman et al. (2022) used mathematical models to create an economic benefit that does not compromise the surrounding environment, in the economic field mathematical models have also been used to describe the German economy (Andreini et al., 2021). Mathematical models are also used for social purposes, for instance Gelarch et al. (2011) used models to optimize the organization of the environment of the cities.

In process engineering, models are used to simulate systems, processes and for process optimization. Models are also used to design industrial equipment, such as reactors and units used for the separation of components. In the chemical and process industry the use of mathematical modelling to describe chemical reactions has acquired a key role in process optimization. The mathematical model needed to describe a chemical reaction are called kinetic models and are used to describe how the reaction conditions affect the speed of the chemical reaction.

Using the kinetic models the experimenter can describe a chemical reaction, and predict the rate of conversion of the chemical species involved in the reaction. This is very useful in the process design phase to design a reactor that maximizes the conversion of the reactant minimizing the operating costs. The structure and the parameters of the kinetic models strongly depend on the considered process and are influenced by several elements of the reactions. One challenge in the development of reaction process models is the identification of the specific reaction rate expressions. Depending on the type of reaction the model change, the catalytic reaction usually

are modelled using the Langmuir-Hinshelwood rate expression that is much more complex respect the power law that is used to describe a combustion reaction.

It is possible, from these considerations, to understand that modelling a process, phenomenon or anything else is very complex. Usually to derive a model the data obtained from a certain number of experiments are used. Looking at the experiments it is possible to hypothesize the structure of several models that are capable to fit the experimental data. According to to Srinivasan et al (2016) to model the dynamic behaviour of a system requires mainly two essential elements:

- appropriate form of the model equations
- precise estimation of the parameters

Moreover, a key aspect is to carry out as few experiments as possible in order to reduce costs because carrying out experiments is usually very expensive.
In recent years several methods that can be used to identify the appropriate dynamic model have been developed and also some methodologies to discriminate among different models have been proposed. Most of the proposed techniques propose a sequential approach to the identification of the model. So first of all they discriminate among all the proposed models and after that, they are used to evaluate the parameters that appear in the model and to maximize the precision of this evaluation. The objective of this work is to build a new method capable to identify simultaneously the structure and the parameters of the model structure and model parameters in the most precise way.

In the following part of this chapter an overview of the most important approaches used for model identification, discriminate among different models and some techniques used to automate this process are reported.

1.1 Conventional Model Building Techniques based on Design of Experiments

As mentioned in the previous section obtaining the data necessary to build the model is an expensive process, both in terms of money and in terms of time. For this reason it has been necessary to develop a systematic technique to maximize the amount of information obtained from experiments. The designed technique is called *design of experiments* (DoE).

1.1.1 Design of Experiments

This methodology is very important because it represents the connection between the experimental and the modelling world. *Design of experiments* aims to obtain the maximum amount of information of the system with the lowest number of runs. To obtain the information of the system it is necessary to perturb it and observe how the system reacts. In *Figure 1* a schematic representation, taken from Franceschini et al. (2008), of the validation procedure based on the DoE methodology is reported. In *Figure 1* the three main steps of the model validation process

Figure 1.1. *Model validation procedure (Franceschini et al., 2008)*

are reported. After one candidate model is proposed, first of all it is necessary to perform some preliminary tests to understand if it is possible to obtain information about the system from the model. Obviously, if it is not possible to obtain information the model is rejected. This is the first discrimination. The surviving models instead go under other experiments to improve the precision of the parameters.
The first idea of *design of experiments* was introduced by Fisher in 1935 (Pearson et al., 1994), this idea was based only on the input and output without taking into account the model that relates them. These methods that consider only the input and the output without considering their connection are called *Black Box methods*. More recent evolution of DoE methods is given by techniques that consider the set of differential and algebraic equations as a model. This methodology is called Model-Based Design of Experiments (MBDoE). The main points of this method, that aims to define a systematic procedure of validation of the models are:

- preliminary analysis
- application of the concepts of *DOE* to discriminate between rival models
- increase the accuracy with which parameters are determined

In *Figure 1.2* a detailed representation of all these steps of the MBDoE procedure is reported. To use this methodology it is necessary to have several proposed models to analyze, usually, the systems analyzed can be described using a system of Differential and Algebraic Equations (DAEs).

1.1.2 Step one: Structural Analysis

The first step of the model discrimination procedure consists of a structural analysis that is based on several concepts. The most important feature that the model must have to pass this first step is

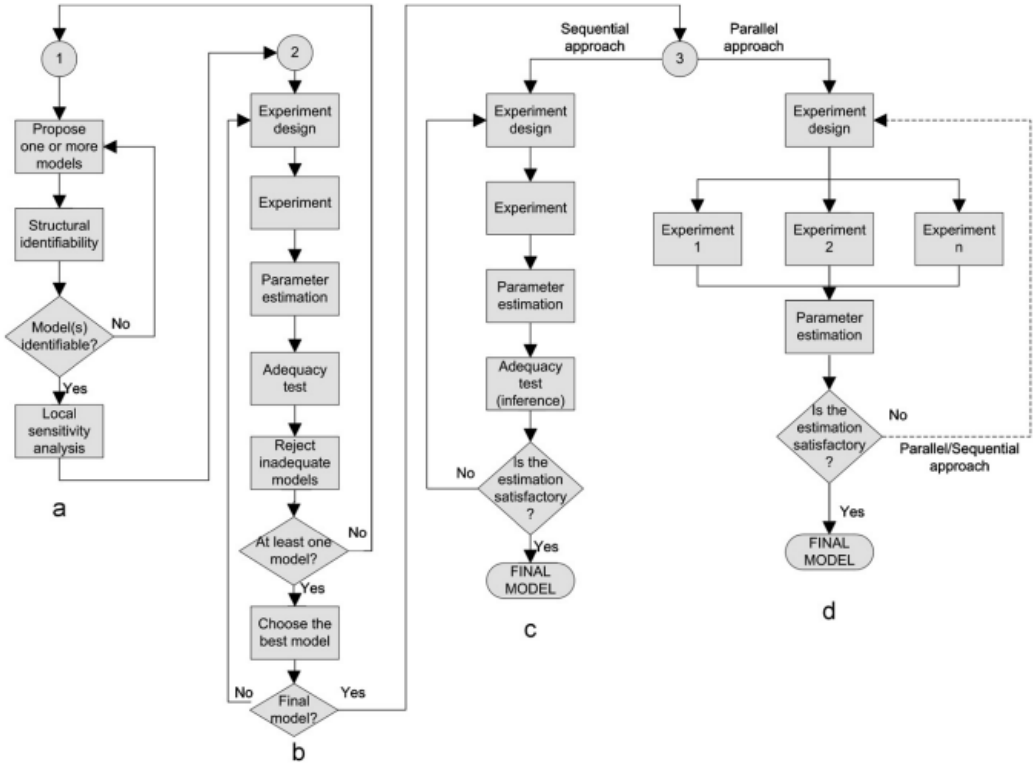

Figure 1.2. *Identification strategy (Franceschini et al., 2008).*

the *identifiability* i.e. the problem of uniquely identifying the set of model parameters. The first to introduce this concept was Bellman et al. (1970). A more accurate description of these steps is reported in *Chapter 2*. All the models that pass this first step can proceed to the following procedures.

1.1.3 Step two: Parameter estimation

After the structural analysis, it is necessary to estimate the numerical values of the parameters. This process is performed as a minimization problem. The parameters are evaluated minimizing the difference between the measurement obtained during an experiment and the predicted values obtained from the models.

The MBDoE technique is becoming more and more popular nowadays, and several variants have been developed in recent years. As reported in *Figure 1.2* the original technique used a sequential approach to perform the experiments (branch c in *Figure 1.2*). Galvanin et al. (2007) proposed a different methodology that exploits a parallel strategy. This method requires a suitable model to be carried out so it is necessary to have a model to test. It is necessary to make sure that there is at least one model available, if there is no model available it is necessary to build it. The process of building a model from zero knowledge of a system represents a very

challenging task. In the following section, an overview of several methods that can be used to build a mathematical expression starting from a dataset is reported.

1.2 Alternative Model Building Techniques

In this section, several methodologies that can be used to identify a mathematical model and an innovative technique used to improve the performance of an already existing model are presented.

1.2.1 Superstructure Method

This method is based on the fact that every mathematical model can be seen as a composition of elementary mathematical functions. This is the same for the models used to describe a system. In a very simplified way with this method, mathematical functions are seen as building blocks used to build a more complex function, the model. In the following figure (*Figure 1.3*) an example that can help to visualize how this technique work is reported.

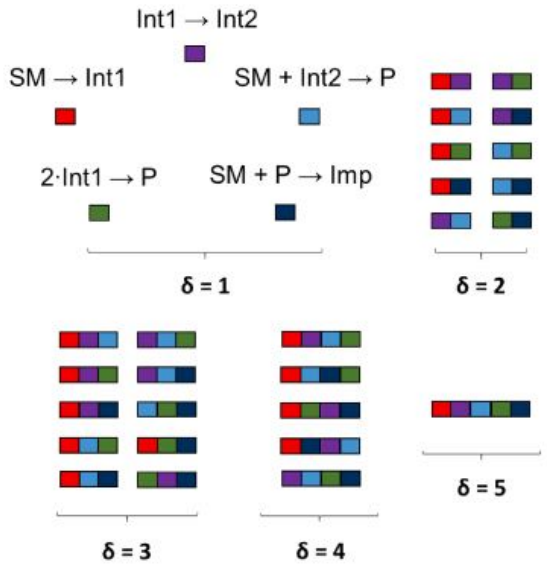

Figure 1.3. *Model Generation Step (Taylor et al., 2021)*

In *Figure 1.3* there are 5 different elementary functions (SM, 2Int1, SM+P, SM+Int2, Int1) that can be assembled to form a more complex model. As it is possible to see there are many ways to assemble the model. In fact, starting from 5 operations it is possible to obtain 31 different models depending on the number of operations you are going to use (in *Figure 1.3* there are 5 possible models for $\delta = 1$, 10 for $\delta = 2$ and 3, 5 for $\delta = 4$ and 1 for $\delta = 5$). In the process of building models, the complexity of the model is unknown and consequently, also the number of operations that make up the model is unknown. This means that there is an additional degree of freedom that is precisely the number of operations that constitute the model. This degree of

freedom is very important because by modifying it, it is possible to decide whether to obtain a very complex model that perfectly describes the data obtained or a simpler model that describes the data a little more roughly. However, the fitting performance of the model, i.e. its ability to represent the experimental observation, is not necessarily proportional to its complexity.
To obtain the structure of the model using a superstructure a particular type of optimization is required. It required an optimization that can identify the operations that make up the mathematical model and at the same time estimate the value of the parameters.

1.2.2 Sparse identification of Non-linear Dynamics (SINDy)

Another methodology that can be used to identify the structure of the model is called *Sparse Identification of Non-Linear Dynamics*, usually abbreviated to *SINDy* (Brunton et al., 2016). This method in some ways is similar to the *superstructure method* (see **Section 1.2.1**) and essentially exploits the fact that in most systems there are only a few relevant terms that define the dynamics of the analyzed system. To apply this method is necessary to have a library of all possible elementary models that can be used to build complex models. Starting from a library it is possible to build a model using a sparse vector, so a vector whose majority of elements is equal to zero. This vector specifies which operations compose the model. In *Figure 1.4* is reported how this method work. In *Figure 1.4* $\dot{X}$ represent the dynamic model, ($\Theta(X)$) is the

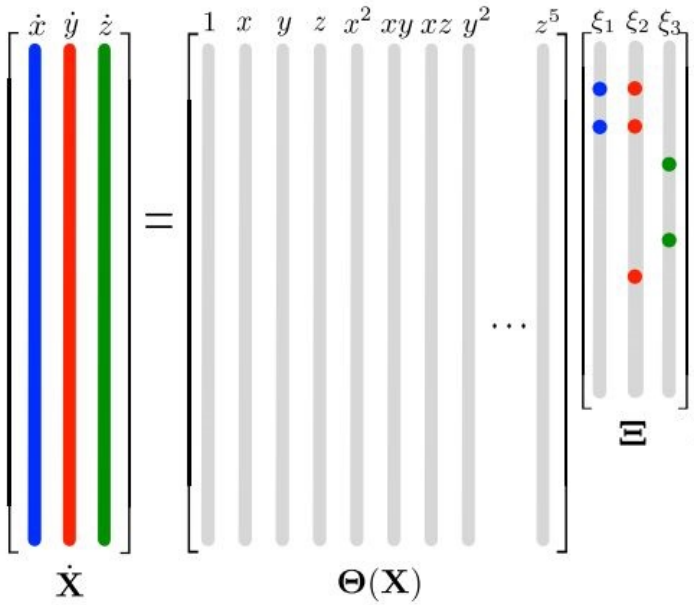

Figure 1.4. *SINDy (Brunton et al., 2016)*

library and (Ξ) is the sparse vector.
Several variants have been developed from this method, for example, a similar methodology, called *reactive SINDy* has been proposed (Hoffmann et al., 2018), which purpose is to identify the mechanism of the chemical reactions and consequently the mathematical models. The ma-

jor disadvantage of these methods is that they rely on a library. It is important to underline this fact because it is impossible to have a library that includes every single mathematical operation within it. So if an operation that is not included in the library is needed, this would be a problem and would not be possible to find the real model.

1.2.3 Artificial Neural Networks

In recent years an innovative method belonging to the class of machine learning techniques has seen its popularity increase. This technique is the *Artificial Neural Network* (*ANNs*) model (Raissi et al., 2018, Zhao et al., 2020). This method had great success mainly because it is a very flexible technique and also because a neural network is essentially a universal approximator.
ANNs are inspired by the biological neural network. Like in the brain of humans (or animals) in the ANNs, there is a large number of "neurons" that respond to inputs and propagate a signal downstream the network.
In biology, Neural Networks are very complex interconnections of neurons communicating through electrical signals. In a brain, the neurons are linked together through a very dense network of synapses and this makes this way of transmitting information very effective and fast. An Artificial Neural Network (ANN) exploits the same information transfer system. Also in the ANN neurons are present neurons (in this case are called nodes) communicating with each other using their connections in the same way as synapses.
The most basic form of Artificial Neural Network was developed by Frank Rosenblatt in 1962 (Van Der Malsburg, 1986) and is called perceptron. A representation of the structure of a single layer perceptron is reported in *Figure 1.5* The perceptron is composed by different components:

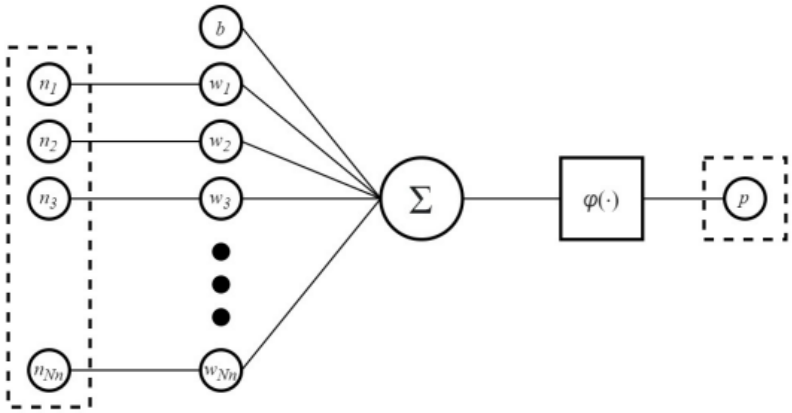

Figure 1.5. *Representation of a perceptron (Quaglio et al., 2020b).*

- an input vector (n) that consists in N_n elements;
- an output, p;
- a vector of weights, w;

- a vector of biases, b.

In *Figure 1.5* the structure of a single layer perceptron is reported but usually, an Artificial Neural Network is made of the combination of several perceptrons organized in different layers. The number of layers and the number of neurons in each layer are parameters that are fixed by the experimenter. The choice of the number of layers is a very important choice because the right value of layers improves the performance of the neural network. In *Figure 1.6* is reported the structure of an Artificial Neural Network.

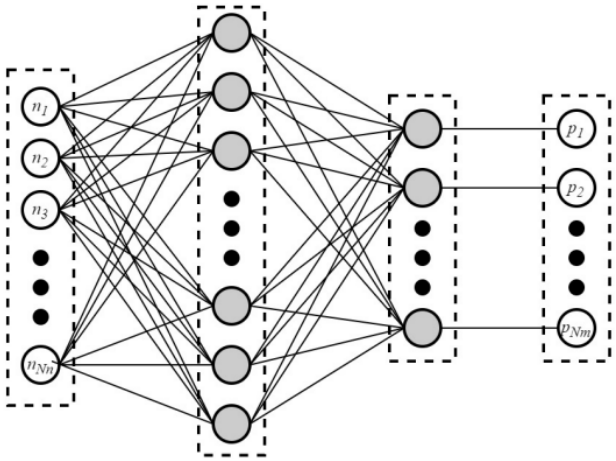

Figure 1.6. *Representation of an Artificial Neural Network (Quaglio et al., 2020b)*

Thanks to their flexibility Neural Networks can be used for several purposes. The first uses of ANN date back to the 1990s when they were used to approximate any non-linear function (Baldi et al., 1989). Nowadays Artificial Neural Networks are widely used to solve automatically classification problems (Heo et al., 2018), for non-linear systems identification (Petsagkourakis et al., 2019) and process control.
Quaglio et al (2020b) proposed a new approach to recognize the kinetic models from experimental data using an Artificial Neural Network. For this application, the ANN is trained for recognising the appropriate kinetic model structure given an initial dataset obtained from experiments. This method is different from the other methods because it does not require evaluating the parameters, and it works very well when there is a high number of candidate models able to describe the phenomenon. A schematic representation of the technique proposed by Quaglio et al., 2020b is reported in *Figure 1.7*. This procedure starts from the definition of a library of suitable models. With the experiment, at some predefined conditions, it is possible to obtain some samples. The process of identification is divided into two steps:

- Generation of the in-silico data. In this step, the data are generated in-silico using the proposed models

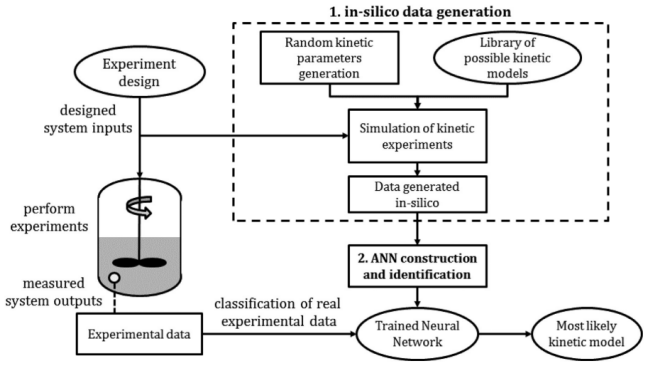

Figure 1.7. *Proposed ANN-based approach for kinetic model recognition, (Quaglio et al., 2020b)*

- The generated data are used to build and train the ANN. In this step the ANN *learns* about the system using the generated data.

After these two stages, a neural network is obtained a neural network that is capable to approximate the system. So this method does not produce a true mathematical model but a neural network that approximates the model. It is worth at this point to introduce a particular type of neural network the Physic Informed Neural Networks, called also PINNs (Raissi et al., 2019). This technique is capable to solve the tasks respecting any given law described by some non-linear equation. This type of neural network is particularly suitable for complex physical, biological and engineering problems. Differently from a standard neural network, the PINN is divided into two parts:

- The first part is a neural network that is used to approximate the behaviour of the system;
- In the second part the physics law are used to determine the solution of the model. This part is called the 'physic part' because the physics knowledge is exploited to obtain the right solution.

1.2.4 Genetic Programming

Genetic Programming (GP) techniques belong to a particular class of probabilistic search procedure called *Evolutionary Algorithms*. These methods mimic the evolutionary processes that happen in nature and have become very used because are extremely effective at searching non-linear models. *Genetic programming* is not really a new methodology, in fact the first documents on genetic programming date back to 1992 with the paper of Koza.
In *Figure 1.8* a schematic representation of the functioning of this method is reported. In *Figure 1.8* is reported the process used to identify a model starting from data. In this case the output of the GP algorithm is a model or a set of validated models.
In *Figure 1.9* the block diagram description of a non-linear system is represented. It is composed of several nodes that represent the structure of the model, they can have variable lengths.

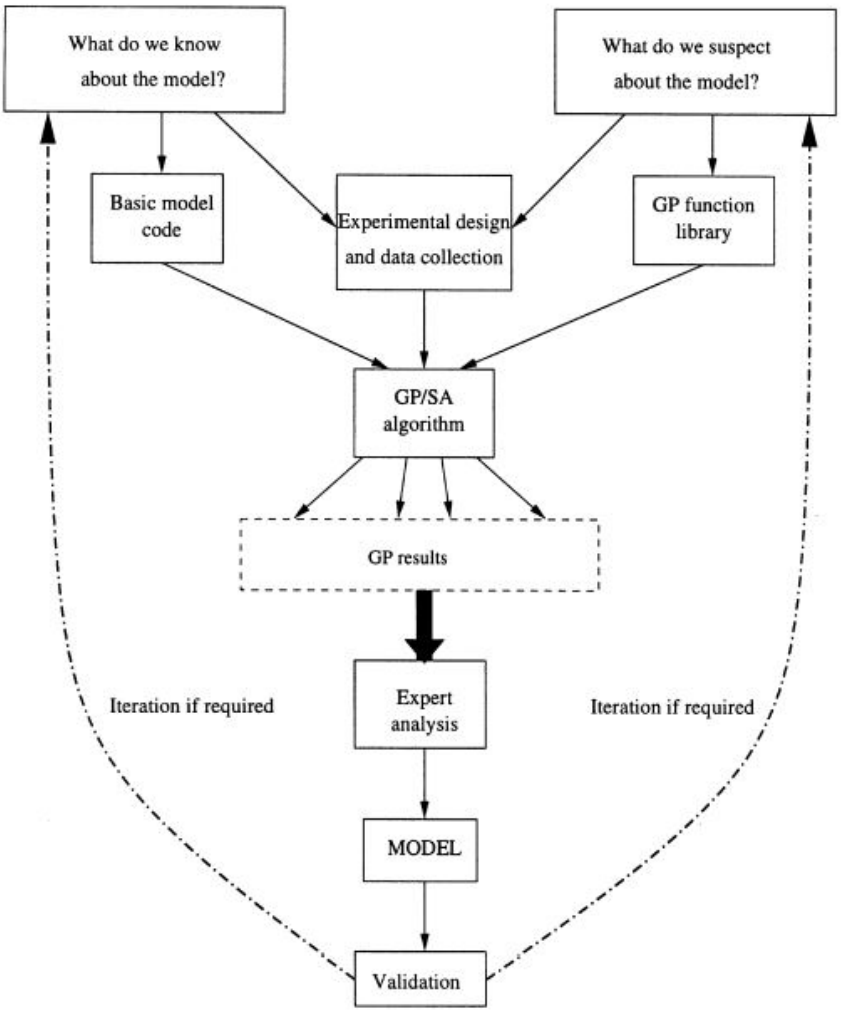

Figure 1.8. *Genetic Programming (Gray et al., 1998)*

The non-linear terminal nodes (in *Figure 1.9 u* and *v*) are functions from the library selected for the dynamic system. The library is a pool of potential and non-linear components of the model structure and using the GP it is possible to select the functions from this library and build the model structure that represents in a more appropriate way our data. For this reason, this function is very important and it should be flexible enough to be able to represent a wide range of functions.

Usually the GP procedures start with a population of a few hundred trees and evolves through three possible actions:

- Crossover: it is applied to 80% of each generation and involves branches from two "parent" trees (see *Figure 1.10*);
- Mutation: it has a low rate and it consists in the creation of a completely new branch

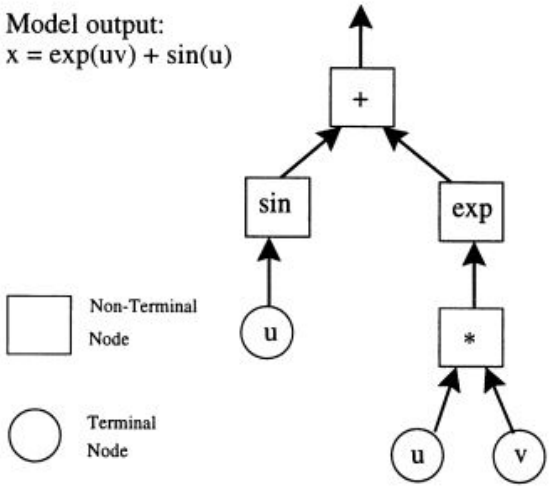

Figure 1.9. *Tree structure of GP (Gray et al., 1998)*

determined randomly;

- Selection: it is the process in which the fittest model is selected.

it is important to specify that only few trees from the initial population "survived" because only the most appropriate models are selected. Obviously the worst models are discarded. There

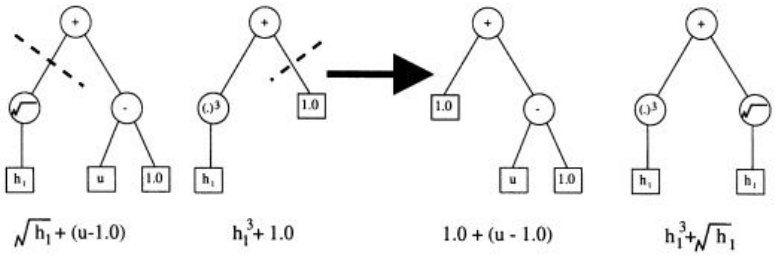

Figure 1.10. *Crossover (Gray et al., 1998)*

are several examples of the use of Genetic Programming to identify the models behind some data (Farag et al., 1998, Gray et al., 1998). The field in which it is more used is the economic field, for example Iba et al. (2000) applied a Genetic Programming methodology to identify the trend of financial data. More precisely they tried to identify the non-linear behaviour behind the financial trends to obtain a model capable to predict the trend of the data. Another example is the use of Genetic Programming for the automatic development of clinical prediction models (Bannister et al., 2018). In their work, Bannister et al. started from an initial population of 1000 mathematical models. The fit of all models (to the training data) in the initial candidate set is calculated. Then in an iterative process:

- A random sample is generated with the current set of models;

- An action is applied in the models to create a new set of models;
- The goodness of fit of the new set of models is evaluated
- The models in the current population are replaced by the newly created models that have better fitness

These steps are repeated for a fixed amount of time. At the end of the time the model that fits the data in the most appropriate way is proposed as solution. A graphic representation of this method is reported in the following figure (*Figure 1.11*).

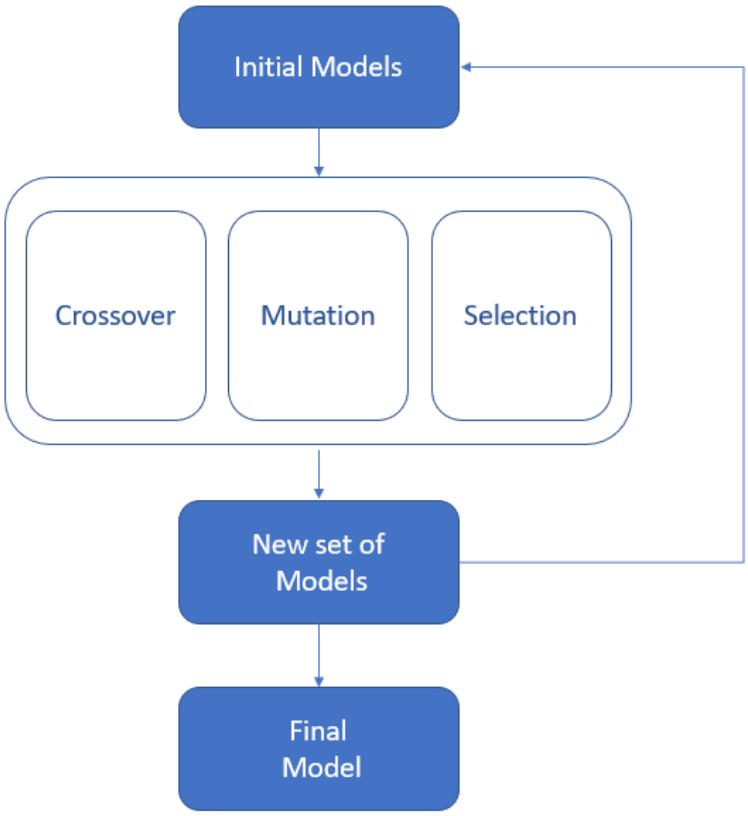

Figure 1.11. *Functioning of Genetic Programming*

1.3 Statistical techniques used for model improvement

1.3.1 Model Modification Indexes

In the previous sections, two methods to identify the structure of the model are explained. In this section instead, a diagnostic procedure used for improving the structure of an existing kinetic model proposed by Quaglio et al. (2020a) is reported. With this method, a setup is assumed to be available to perform some experiments and obtain experimental data. The experimenter

proposes a kinetic model that consists of a series of differential and algebraic equations. Differently from the previous method, this method requires one model that will be modified during the procedure. This method has different steps:

- Step 1: *Parameters Estimation*. In this step the parameters of the model are identified using the maximum likelihood approach proposed by Bard, 1974;
- Step 2: *Goodness-of-fit test*. The adequacy of the model to describe the data is evaluated.

The second step has three possible results:

- Passed: the model is not falsified and the complexity is adequate for representing the data (this mean that there is no over/under-fitting);
- Falsified for over-fitting: if this result is obtained it means that the model is too complex to describe the process. In this case, it is necessary to remove some parameters to make the model simpler;
- Falsified for under-fitting: in this case, the model is too simple to describe the data and it can not capture the mechanism behind the process. To solve this problem it is necessary to evaluate the Model Modification Indexes (MMI) to understand which parameters it is necessary to modify to make the model more complex.

The Model Modification Indexes are a statistic that evaluates how much is possible to improve the performance of the model by replacing the parameters with a state function. If the MMI is larger than 1 it indicates that it is possible to improve the performance of the model by replacing the parameter with a state function. Instead, if the MMI is smaller than 1 then there is no statistical evidence that justifies the replacement of the parameter with a more complex state-function. In the *Figure 1.12* is reported a schematic representation of how the Model Modification Indices work.

In this section, the methodologies for identifying the structure of the model have been reported. It is possible to notice that all these methods are well-proven but none of them identifies the model and simultaneously estimates the parameters. In this work the goal is to define a technique, starting with one of the above techniques, that managed to completely define the model and the parameters simultaneously with no user intervention. So now it is necessary to introduce the techniques that can be used to automate the whole process.

1.4 Reinforcement Learning

In this section there is an overview of the *Reinforcement Learning* (RL). Over the last decades, Machine Learning has started to deepen the artificial intelligence field. The branch of Machine

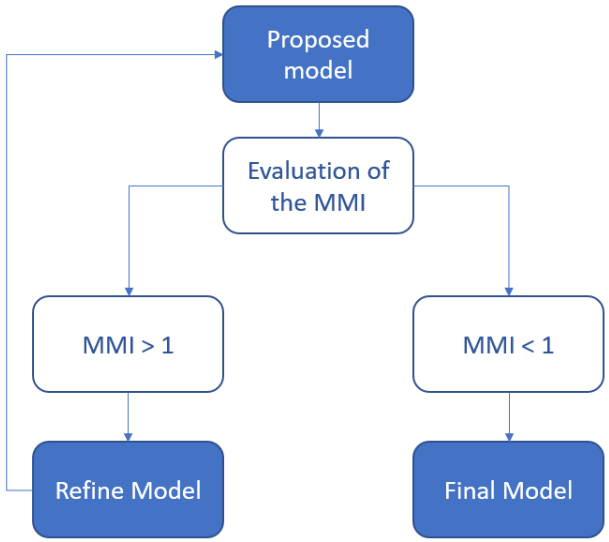

Figure 1.12. *Functioning of the Model Modification Indexes*

Learning that deals with this topic is called Reinforcement Learning (RL). The idea behind the RL methodologies is learning from interactions with the environment (Johnson et al., 2000). In these problems, the goal is to learn how to act to maximize a numerical reward. The main difference between an RL problem and an other Machine Learning problem is that in the RL there are no instructions to follow in order to maximize the reward. A schematic representation of the functioning of the reinforcement Learning algorithms is reported in *Figure 1.13*.

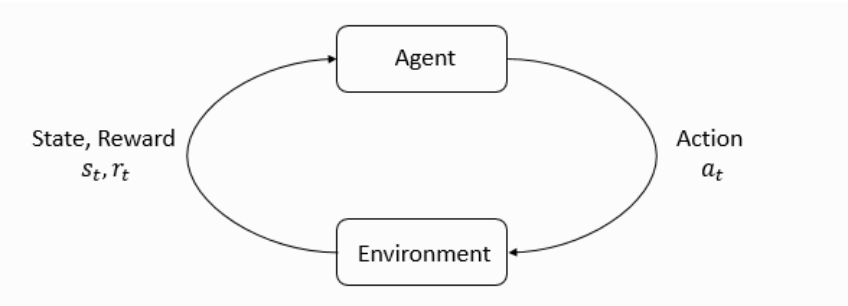

Figure 1.13. *Agent-environment interaction loop*

To fully characterize the RL problem is necessary to introduce some fundamental concepts.

The first concept is the concept of *States and Observations*. Since the RL methodologies are based on learning from interacting with the environment it is necessary to describe the world that surrounds the learner. A *State* (s) is a complete description of the state of the world. Since usually, it is not possible to obtain a complete description of the environment in RL methods

instead of *states* we use the *observations* (o) that are a partial description of the state. The difference between a *state* and an *information* is that the *information* might omit some details of the environment. In RL algorithm usually *states* and *observations* are a real-value vector or an higher-order tensor.

Another fundamental character in the reinforcement learning algorithms is the *agent*. The *agent* is essentially the learner that interacts with the environment learning from these interactions how to maximize the reward. Every action that the agent takes corresponds to a reward that returns to the agent. The goal, therefore, is to maximize the amount of reward.

The third concept that is necessary to define is the concept of *policy*. The *policy* is used by the agent to decide what actions to take. There are mainly two types of *policy*: a deterministic policy (denoted with the letter μ) and a stochastic policy (denoted with π). The *policy* is essentially the "brain" of the agent for this reason it is very common to substitute the word *agent* with the word *policy*.

Every state and every action taken by the agent and any reward are collected in a vector called *trajectory* (τ). In this vector there is a first state s_0 that is randomly selected. The state transition, i.e. what happens between the state at time t and the state at time t+1, is governed by natural laws of the environment that depends only on the most recent *action*. The fact the state depends only on the most recent action is typical of a particular type of problem called Markov Decision Process (MDP).

Another pivotal element on Reinforcement Learning is the *Reward Function*. It depends on the current state of the environment, on the action and on the next state. As mentioned above the objective of RL is to maximize the the cumulative reward over a trajectory ($R(\tau)$). The easiest type of *reward* is the finite-horizon undiscounted reward which is just the sum of the reward obtained in a fixed number of steps. Another type of reward is the infinite-horizon discounted return. In this case there is no limit to the number of time steps. In this case a discounted factor, i.e. a constant between 0 and 1, is added. this factor is necessary because without it the convergence is not assured.

The last important feature of Reinforcement Learning the *Value Function* (V^{π}). The *Value Function* gives the expected return if the initial starting state is s and if the agent act always accordingly to a policy π. An improvement of the *Value Function* is the *Optimal Action-Value Function* that gives the expected return if the agent start in a state s, take an arbitrarily action a and then act always accordingly to the policy.

RL techniques have proven to be a very powerful tool that can solve non-linear problems and systems of differential equations. For this reason RL has been used to use handle complex

optimal control problems. In addition RL methodologies have already been used also for the optimization of bio-processes (Pan et al., 2021a, Pan et al., 2021b). Since these techniques are already been applied for similar applications it will be interesting to see how they can handle the optimization of the process of model identification and parameter estimation and if they can.

1.5 Objective

The aim of this work is to develop a new methodology that is capable of identifying the structure of the model and to identify the parameters with good precision simultaneously. The important thing is that this new method has been developed coupling the standard methodologies of MBDoE with RL techniques to obtain an approach that manages to identify the most suitable model starting from the data autonomously. In fact in this work RL has been used to optimize some statistics obtained using the MBDoE techniques in order to find the optimal conditions to run the experiment.

Chapter 2

Superstructure and Reinforcement Learning-based approaches for model identification

In this chapter the developed method and the math behind are explained. In the first section of this chapter an overview of the method is reported where the steps of the method are explained in a very general way. In the following sections more detailed explanations are given of the methods used and the mathematics behind them.

2.1 Method introduction

In this first section the rationale behind the method developed is given. In *Figure 2.1* is reported a schematization of how this method work.

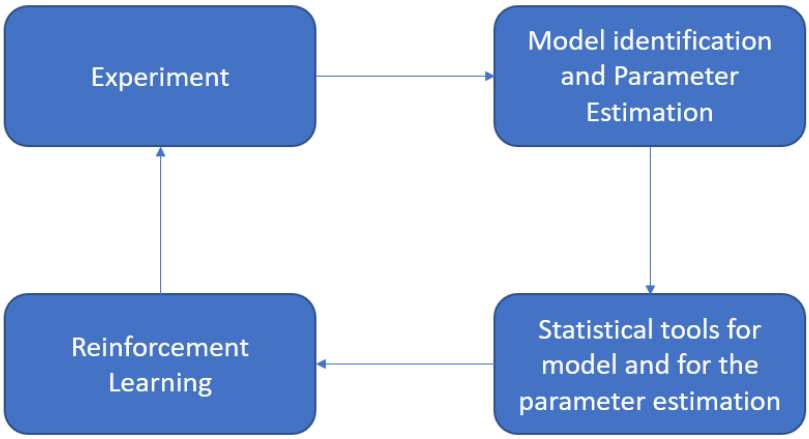

Figure 2.1. *Schematic representation of the proposed method*

It is possible to notice that it is organized into four different steps:

- Step 1: the first step is the experiment, from which the experimenter obtains the data that are necessary to build the model;

- Step 2: after building the dataset it is necessary to identify the structure of the model and to estimate the parameters (block 2);
- once identified the model and the parameters the statistics are evaluated, in particular, it is very important to evaluate the Model Modification Indexes of the model.
- Step 4: the indexes evaluated in Step 3 are then sent to the Reinforcement Learning technique that generates a new set of input for the considered system.

In the next sections all these steps are thoroughly explained.

2.2 Experiment

In the first step of this method it is necessary to build a dataset that contains the experimental measures of the process which will be compared with the measures obtained using the model. Since it is not possible to perform the actual experiment it is necessary to recreate an in-silico experiment in order to obtain the experimental data that are necessary to identify the structure of the model and to estimate the parameters.

2.3 Model identification and Parameter Estimation

2.3.1 Superstructure Method and MINLP for the model identification

This method has been used to identify the structure of the model using only the data. As explained in *Chapter 1* with this technique is possible to identify the structure of the mathematical model starting from an initial superstructure, i.e. a very complex mathematical expression that contains all the model alternatives. This technique was chosen because by using it, it is possible to determine both the structure and the value of the parameters simultaneously. This is a very complex task because to identify both the structure and the parameters it is necessary to work both with discrete and continuous variables. The discrete variables are used to identify the structure of the model instead the continuous variables are the parameters. To do this it is necessary to introduce a particular type of optimization called Mixed-Integer Non-Linear Problem (MINLP). This technique is a very general type of optimization that combines both Nonlinear programming (NLP) and Mixed-Integer Linear Programming (MILP). This optimization technique is widely used to solve this type of optimization in many fields, such as the engineering field (as in this case) but also in some financial applications. Usually, the MINLP is expressed as:

$$\begin{aligned} &min f(x,y) \\ &s.t.\ \ x \in X \subseteq \Re \\ &\quad y \in Y \subseteq Z \end{aligned} \tag{2.1}$$

where $f : (x,y) \mapsto \Re$. The *Equation 2.1* is very representative of engineering problems in particular of chemical applications where the non linear functions can be used to describe the behaviour of chemical reactions. It is possible to add to the *Equation 2.1* some equality or inequality constraints ($c(x,y)$), so the result will be:

$$\begin{aligned} & min f(x,y) \\ & s.t.\ \ c(x,y) \leq 0 \\ & \quad x \in X \subseteq \Re \\ & \quad y \in Y \subseteq Z \end{aligned} \tag{2.2}$$

Equation 2.2 is a very complex problem because there is an high number of potential solutions that have to be investigated.
From this consideration another problem arise, in fact being the optimization problem computationally very expensive it was necessary to change the integration method. In fact using the standard integration technique the method was too slow (it took about 6 hours only to obtain a solution of the MINLP). For this reason, it was necessary to change the standard integration method with an alternative one. The techniques that have been analyzed to solve this problem are:

- Runge Kutta;
- Implicit Euler;
- Orthogonal Collocation.

Among these methods the Implicit Euler was chosen, because the Runge Kutta was easy to implement but was not very accurate, the Orthogonal Collocation was very accurate but was very difficult to implement. The Implicit Euler was a good trade-off between accuracy and simplicity of implementation.

2.3.2 Implicit Euler

In this section the explanation the the Implicit Euler (also called Backward Euler method) is reported. Let's consider a differential equation (our model) in the standard form:

$$\begin{aligned} & \frac{dy}{dt} = f(y,t) \\ & y(t_0) = y_0 \end{aligned} \tag{2.3}$$

where f is the function, the initial state, $y(t_0)$ is known and y depends on the time (t). Then a sequence of state, y_0, y_1,y_2, ..., is produced until the state y_k that approximates $y(t_0 + kh)$ where h is the step size. Using the Implicit Euler it is possible to compute the approximations using the following expression:

$$y_{k+1} = y_k + hf(t_{k+1}, y_{k+1}) \tag{2.4}$$

From *Equation 2.4* it is possible to notice why the Backward Euler is also called implicit Euler. In fact in *Equation 2.4* the new approximation y_{k+1} appears both on the left-hand side and on the right-hand side of the equation. For this reason it is necessary to solve an algebraic equation to identify the unknown y_{k+1}, and this can be done using a fixed-point iteration:

$$y_{k+1}^{[0]} = y_k, \quad y_{k+1}^{[i+1]} = y_k + hf(t_{k+1}, y_{k+1}) \tag{2.5}$$

and if this equation converges, with a predefined tolerance then it is possible to take as new limit the new approximation y_{k+1}.

Integrating the differential equation expressed in the *Equation 2.3* form a the initial time (t_n) to the following time step (t_{n+1}) yields:

$$y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} f(t, y(t))dt \tag{2.6}$$

it is necessary now to approximate the integral that appears in the right hand side of the equation. For this purpose is used the *rectangle method*:

$$y(t_{n+1}) - y(t_n) \approx hf(t_{n+1}, y(t_{n+1})) \tag{2.7}$$

From this expression it is possible to evaluate y_n that represent $y(t_n)$.

2.3.3 Parameter estimation

After identifying the structure of the model it is necessary to evaluate the parameters (θ). The estimation of the parameters required the fitting of the experimental data. To do that, a dataset containing the experimental measurements is necessary. In this work the dataset will be identified with the following notation:

$$Y = (y_1, y_2, ..., y_N) \tag{2.8}$$

where N is the number of measurements. First of all it is necessary to set the conditions adopted to run the experiment used to collect the samples. Moreover, it is fundamental to specify that all the measurements are corrupted by a Gaussian Noise with a preset covariance.
To estimate the parameter a method proposed by Bard called Maximum Likelihood (ML) is used. The rationale behind this method is to identify the parameters, called maximum likelihood estimates, that minimize the differences between the experimental data. To identify the maximum likelihood estimates, following the approach introduced by Bard, it is necessary to

maximize the likelihood function. In *Equation 2.9* the logarithmic form of the likelihood function is reported:

$$\mathbb{L}(Y|\theta) = \frac{N}{2}[N_y ln(2\pi) + ln(det(\Sigma_y))] - \frac{1}{2}\sum_{i=1}^{N}[y_i - \hat{y}_i(\theta)]^T \Sigma_y^{-1}[y_i - \hat{y}_i(\theta)] \tag{2.9}$$

where $\hat{y}_i$ are the predictions obtained with the model and Σ_y is the covariance. The maximum likelihood estimates, denoted with $\hat{\theta}$, are computed by maximizing the log-likelihood function:

$$\begin{aligned} &\hat{\theta} = argmax_\theta \mathbb{L}(Y|\theta) \\ &\hat{\theta} = [\hat{\theta}_1, ...\hat{\theta}_p]^T \end{aligned} \tag{2.10}$$

2.4 Statistical tools for model and for the parameter estimation

After having defined the structure of the model and identified the parameters, it is necessary to evaluate the adequacy of the model using some statistical tests. These tests are intended to assess whether the parameter estimates are precise enough and to detect if there is *overfitting* or *under-fitting* of the data.

2.4.1 Goodness of Fit

This test is always performed when there is a parameter estimation and its aim is to evaluate if the model can approximate correctly the experimental data. To do this, it is necessary to evaluate the distribution of the model residuals compared to the distribution that the measurement errors may have. This comparison can be performed with a χ^2 test that quantifies the probability of observing a certain distribution of the residual under the hypothesis that the model is correctly specified. This hypothesis is also called *null hypothesis*. To perform the χ^2 test, it is necessary to start from the maximum likelihood, $\hat{\theta}$, and to pose that the model residuals follow the same distribution of the measurement noise, that is a Gaussian distribution with mean equal to zero and covariance equal to Σ_y. What has just been said can be written also in mathematical form as:

$$y_i - \hat{y}_i(\hat{\theta}) \sim \mathbb{N}(0, \Sigma_y) \quad \forall i = 1, ..., N \tag{2.11}$$

From this assumption it is possible to notice that that the squared residuals is a variable that follow a χ^2 distribution. In this statistic the degree of freedom is $N \cdot N_y - N_\theta$.

$$\chi^2_Y = \sum_{i=1}^{N}[y_i - \hat{y}_i(\theta)]^T \Sigma_y^{-1}[y_i - \hat{y}_i(\theta)] \sim \chi^2_{N \cdot N_y - N_\theta} \tag{2.12}$$

To determine the fitting of the model it is possible to perform a two-tailed goodness-of-fit test based on the χ^2. There are three possible outcomes from this test that can be summarized in:

- failed for over-fitting;
- passed;
- failed for under-fitting.

These results are obtained when the following conditions are met:

$$\begin{cases} \chi_y^2 < \chi^2\left(\frac{1-\alpha}{2}\right) \\ \chi^2\left(\frac{1-\alpha}{2}\right) < \chi_y^2 < \chi^2\left(\frac{1+\alpha}{2}\right) \\ \chi^2\left(\frac{1+\alpha}{2}\right) > \chi_y^2 \end{cases} \tag{2.13}$$

where $\chi^2\left(\frac{1-\alpha}{2}\right)$ is the percentile f χ^2 distribution with $N \cdot N_y - N_\theta$ degree of freedom. The argument inside the brackets is the level of significance. From *Equation 2.13* it is possible to notice that when the test failed for over-fitting there is a small probability to find residuals smaller or equal to χ_y^2, and if the model fails the test for over-fitting usually it means that the expression contains an excessive number of parameters. If there are too many parameters it means that there are consequently too many degrees of freedom to capture the trend of the noise. On the contrary, when the model fails the test for under-fitting it means that there is a low probability to find a residual larger than χ_y^2 and this means that the model is not adequate to describe the system. Instead, if the model passes the test it means that it can describe the system in an appropriate way.

2.4.2 Statistics on Parameters

The aim of this work is to identify the model that can fit the experimental data in the most appropriate way possible and to do this it is necessary to identify the most suitable set of parameters. For this reason, it is important to evaluate how much a variation on the parameters influences the output of the model. To do this the sensitivity matrix is very useful. The sensitivity matrix, $\mathbf{Q}$, is defined as a matrix of dimension N x p where N is the number of samples and p is the number of parameters. In the following equation the expression of the sensitivity matrix is reported.

$$\mathbf{Q} = \begin{bmatrix} \frac{\partial \hat{y}_1}{\partial \theta_1} & \cdots & \frac{\partial \hat{y}_1}{\partial \theta_p} \\ \vdots & \ddots & \vdots \\ \frac{\partial \hat{y}_N}{\partial \theta_1} & \cdots & \frac{\partial \hat{y}_N}{\partial \theta_p} \end{bmatrix} \tag{2.14}$$

The sensitivity matrix in *Equation 2.14* is obtained starting from the model equations and applying a partial differentiation:

$$\frac{d}{dt}\frac{\partial \hat{y}}{\partial \theta} = \frac{\partial f}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta} + \frac{\partial f}{\partial \theta} \tag{2.15}$$

where $\partial f/\partial y$ is the *Jacobian* of the system. It is important to underline the fact that in dynamic systems the sensitivity matrix is also dynamic so it is time varying and this means that it assumes a different value at every time step t. The dynamic sensitivity matrix, Q_r, has the same dimensions of the standard sensitivity matrix and is defined as:

$$\mathbf{Q}_r = \begin{bmatrix} \left.\frac{\partial \hat{y}_r}{\partial \theta_1}\right|_{t_1} & \cdots & \left.\frac{\partial \hat{y}_1}{\partial \theta_p}\right|_{t_1} \\ \vdots & \ddots & \vdots \\ \left.\frac{\partial \hat{y}_r}{\partial \theta_1}\right|_{t_N} & \cdots & \left.\frac{\partial \hat{y}_r}{\partial \theta_p}\right|_{t_N} \end{bmatrix} \tag{2.16}$$

Both the sensitivity matrix and the dynamic sensitivity matrix depend on the conditions under which they were evaluated and obviously also on the values of the parameters on the model.
Starting from the sensitivity matrix and from the dynamic sensitivity matrix, and following the approach proposed by Bard (1974), can be used to obtain the expression of the covariance (*Equation 2.17*).

$$V(\hat{\theta}, \varphi) = \left[\Sigma_\theta^{-1} + \sum_{i=1}^{N_y} \sum_{j=1}^{N_y} s_{ij} Q_i^T Q_r \right]^{-1} \tag{2.17}$$

where φ is a vector containing the design parameters of the experiment. From *Equation 2.17* it is possible to notice that the variance covariance matrix depends on several variables:

- dynamic sensitivity coefficient of the ith response
- the variance-covariance matrix of the experimental measurements, Σ_y, (s_{ij} is the (i,j) element of the inverse of Σ_y).
- initial approximation of the variance covariance matrix.

Starting from the variance-covariance matrix it is possible to define another fundamental statistic: the Fisher Information Matrix. Zullo (1991) derived the expression of the dynamic information matrix that is particularly useful for solving optimal design problems in dynamic systems. The expression of the Information Matrix proposed by Zullo is:

$$M(\hat{\theta}, \varphi) = \sum_{k=1}^{N} \sum_{i=1}^{N} \sum_{j=1}^{N} s_{ij|k} Q_{i|k}^T Q_{j|k} + M^0 \tag{2.18}$$

where $Q_{i|k}$ is the dynamic sensitivity matrix of the $i-$th measured response of the experiment k and M^0 is an optional element that represents the initial amount of information on the parameters. The initial amount of information can be derived using the following expression:

$$M^0 = [\Sigma_y]^{-1} \tag{2.19}$$

$$M^0 = \frac{1}{12}(u_b - l_b) \tag{2.20}$$

For this project *Equation 2.20* has been used, where u_b and l_b are the upper bound and the lower bound of the parameters.
From the information matrix, evaluated as in *Equation 2.18* it is straightforward to obtain the variance-covariance matrix of the parameters:

$$V(\hat{\theta}, \varphi) = [M(\hat{\theta}, \varphi)]^{-1} \tag{2.21}$$

From *Equation 2.18* and *Equation 2.21* it is possible to notice that both the information matrix and the variance-covariance matrix depend on the parameters θ that are unknown. Starting from the variance-covariance matrix it is possible to derive the confidence interval of each estimated parameter and also the correlation coefficients c_{ij} between two parameter $\hat{\theta}_i$ and $\hat{\theta}_j$. The result of the combination of all the correlation coefficients is the correlation matrix. Each correlation coefficient can be evaluate using the following equation:

$$c_{ij} = \frac{v_{ij}}{\sqrt{v_{ii}}\sqrt{v_{jj}}} \tag{2.22}$$

where the v_{ij} are the elements of the variance covariance matrix. In the correlation matrix if there are values outside the diagonal that assume values close to one it means that there is an high correlation between the model parameters.

It is possible also to evaluate the quality of the estimation of the parameters, to do that is used a test called *one-tailed t-test*. With this test the t-values of each parameter are evaluated and they are compared with a reference t-value.

$$\frac{\hat{\theta}_i}{t\left(\frac{1+\alpha}{2}\right)\sqrt{v_{ii}}} \geq t(\alpha) \quad \forall i = 1, ..., p \tag{2.23}$$

The $t(\cdot)$ at the denominator in *Equation 2.23* is obtained using a Student's distribution with a number of degrees of freedom equal to $N \cdot N_y - N_\theta$. In *Equation 2.23* α represents the significance of the test and $t(\alpha)$ is the reference t-value. If the condition expressed in *Equation 2.23* is satisfied this means that all the the parameters are evaluated with a good precision.

Moreover, the variance-covariance matrix can be used to evaluate also the confidence ellipsoid, which represents the confidence region of the parameter estimates. In the following equation (*Equation 2.24*) is reported an example of evaluation of the 95% of confidence ellipsoid.

$$[\theta - \hat{\theta}]^T V^{-1} [\theta - \hat{\theta}] \leq \chi^2(95\%) \tag{2.24}$$

To a better visualization of the confidence ellipsoid a graphical representation is reported in *Figure 2.2*.

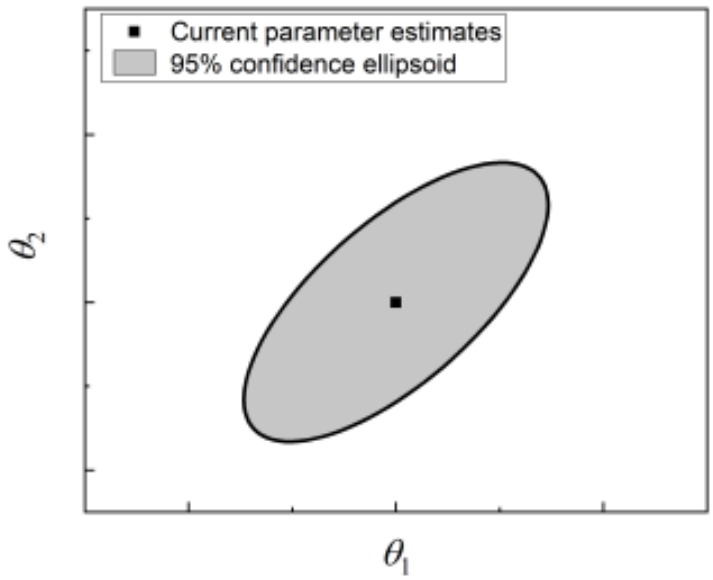

Figure 2.2. *Example of confidence ellipsoid for two parameters (θ_1 and θ_2) (Franceschini et al., 2008)*

2.4.3 Statistical techniques for model improvement

In this section, the math behind the method used to improve the structure of a mathematical model is explained. In the previous section, there is an overview of the different methods that can be used to detect the adequacy of the parameters. The problem now is to find a method to resolve the case in which the model fails the Goodness-of-Fit test. Using the Goodness-of-fit test it is possible to understand if the model over-fit or under-fit the data but if the model failed the test no information is obtained to improve the model.

Through the years different methodologies have been proposed, for example, the Wald test and the Lagrange multipliers test are applied very often in the field of structural modelling of equations. These tests aim to evaluate the null hypothesis that the parameters of the model satisfy some constraints. One may use these tests to prove the hypothesis that some parameters

are equal to zero and if there are insufficient elements to reject this hypothesis these parameters should be set equal to zero.
Whenever the model fails the Goodness-of-fit test for over-fitting, the Wald test can be used to inform the experimenter on which estimates can be set equal to zero and in this way removed from the model structure. The case in which the model fails the test for under-fitting is more complex, in fact in this situation the only way to solve the problem is to change the structure of the model. In this case, the Lagrange multipliers technique is applied to substitute the analyzed model with a more complex alternative, but there is an important limitation that is that the definition of the appropriate alternative relies only on the intuition of the experimenter. In *Figure 2.3* is reported the systematic approach that is usually used to solve the under-fitting or the over-fitting problem. In the following section is explained a new method proposed by

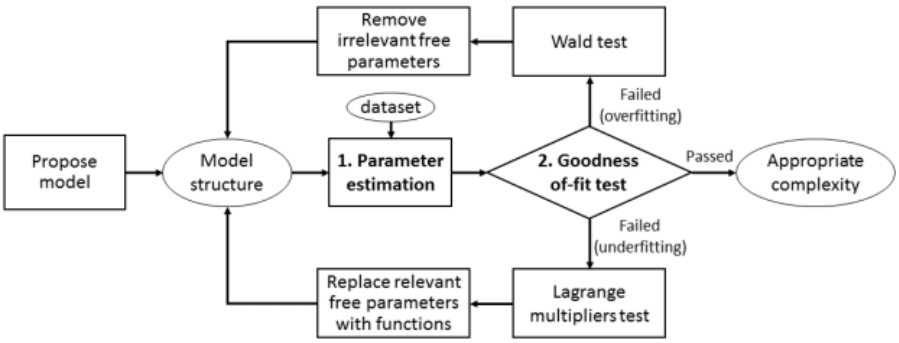

Figure 2.3. *Framework of the model improving procedure (Quaglio et al., 2020a)*

Quaglio et al., 2020a to improve in an iterative way the structure of the model until a good level of complexity is reached.

If the model fails the Goodness-of-fit test it means that there are several differences between the predicted value and the experimental measurements. As said a few lines above it is possible to reduce this discrepancy by replacing some model parameters with a function of the state variables. With the method proposed by Quaglio et al., a statistical test is applied to diagnose the model misspecification challenging the initial hypothesis that a parameter θ_i is a constant that does not depend on the states. So this method is used to evaluate if changing a specific component of the parameter with a free parameter it is possible to improve the fitting of the model.
First of all, it is necessary to specify the two initial hypotheses:

- **Null Hypothesis**(H_0): θ_i and $\theta_j \ \forall j \neq i$ are state-independent constants;
- **Alternative Hypothesis**(H_a): θ_i is a state-dependent function and $\theta_j \ \ \forall j \neq i$ are stet-independent constants.

For the initial parameter estimation problem it is assumed that θ_i is a function of the experimental conditions so $\theta_i = \theta_i(\varphi)$. For sake of simplicity and without loss of generality it is possible to assume that θ_i is θ_1. So θ_1 is a function of the initial conditions and all the other parameters are constants. In this case, the log-likelihood function is equal to:

$$\begin{aligned}\Phi_d({}_d|Y) =& \frac{N}{2}[N_y ln(2\pi) + ln(det(\Sigma_y))] \\ &- \frac{1}{2}\sum_{i=1}^{N}[y_i - \hat{y}_i(\hat{\theta}_{1,i}, \hat{\theta}_2, ...\hat{\theta}_N)]^T \Sigma_y^{-1}[y_i - \hat{y}_i(\hat{\theta}_{1,i}, \hat{\theta}_2, ...\hat{\theta}_N)]\end{aligned} \tag{2.25}$$

The elements in the sum are function only of the parameter θ_1 because all the other parameters are fixed and are set equal to their maximum likelihood value. The set of functions (s) is defined as:

$$\mathbf{s} = [\theta_{1,1} - \theta_{1,2}, ..., \theta_{1,N-1} - \theta_{1,N}]^T \tag{2.26}$$

Using the set of functions it is possible to express mathematically the two initial hypotheses:

$$\begin{aligned} H_0 &: s = 0 \\ H_a &: s \neq 0 \end{aligned} \tag{2.27}$$

If s is equal to 0 it means that the parameter θ_1 is constant and so it is independent of the experimental conditions. So under the hypothesis H_0, all the elements in the set of parameters are equal to their maximum likelihood estimate obtained with *Equation 2.25*. This constrained maximum satisfies also the set of constrained maximum likelihood equations:

$$\nabla\Phi_d(\hat{\theta}_d|Y) + \nabla s\hat{\alpha} = 0 \tag{2.28}$$

where α is the array of the Lagrange multipliers associated with the constraints. It is possible to evaluate the Lagrange multipliers using the following equation:

$$\xi_1 = \hat{\alpha}^T \nabla s^T M_d^{-1} \nabla s\hat{\alpha} \sim \chi^2_{N-1} \tag{2.29}$$

where M_d is the Fisher information matrix of the model under-diagnosis evaluated using the following equation:

$$M_d = \sum_{i=1}^{N} \nabla\hat{y}_i(\hat{\theta}_{1,i})\Sigma_y^{-1}\nabla\hat{y}_i(\hat{\theta}_{1,i}) \tag{2.30}$$

One thing it is necessary to pay attention to is that the solution of the *Equation 2.28* is not required to identify the Lagrange multipliers in fact they can be computed as a function of the log-likelihood:

$$\xi_1 = \nabla\Phi_d(\hat{\theta}_d|Y)M_d^{-1}\nabla\Phi_d(\hat{\theta}_d|Y) \sim \chi^2_{N-1} \tag{2.31}$$

In this way, it is not necessary to compute $\hat{\alpha}$. In the following examples, the Lagrange multipliers statistic is evaluated using the *Equation 2.31*. This procedure is repeated for every parameter of the model and then using these Lagrange multipliers it is possible to identify a measure of the misspecification of the model:

$$MMI_i = \frac{\xi_i}{\chi^2_{N-1}(95\%)} \quad \forall i = 1,...p \tag{2.32}$$

From *Equation 2.32* is evident that the Model Modification Index (MMI) is the ratio between the Lagrange multiplier and the χ^2 distribution at 95% of significance. An MMI larger than one indicates that the null hypothesis is falsified by a $\chi^2(95\%)$, and this means that it is possible to expect a significant improvement of the fitting capability of the model if the parameter θ_i were replaced by a state-dependent function. Instead, if all the MMIs are smaller than one there is no evidence that an improvement of the model is possible changing the parameter with a state-dependent function. Another important fact is that the MM indexes are proportional to the improvement that can be reached, in fact, if the MMI is larger than 1 for more parameters one shall expect a more significant improvement in the model fitting if the parameter with the MMI with the higher value were replaced with a function of the state.

2.5 Reinforcement Learning

In this section the algorithm of Reinforcement Learning is reported that has been used in the second part of this project (Chapter 4).

2.5.1 Hyperparameter Optimization and Bayesian Optimization

Bayesian Optimization has been widely used to tune the hyperparameters in Machine Learning applications. This method is conceptually not very complex, even though very complex equations with many terms are often involved. This method, together with three others, belongs to the class of the hyperparameters optimization methods:

- Random Search
- Manual Search
- Grid Search
- Bayesian Optimization

The Bayesian Optimization and the Manual Search differ from the other two methods because they exploit the iterations already carried out, instead, the Random Search and the Grid search are independent of past evaluation. The basic concept behind the Hyperparameter Optimization is reported in the *Figure 2.4*.

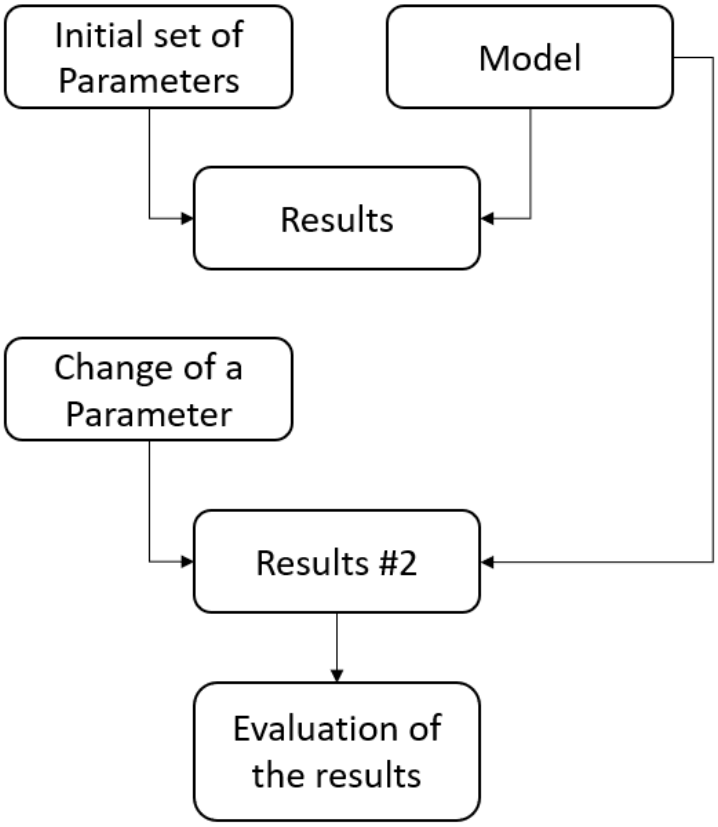

Figure 2.4. *Schematic representation of the proposed approach*

- step 1: a set of parameters is proposed;
- step 2: evaluation of the results obtained using the parameters of the Step 1;
- step 3: change of one parameters of the initial set;
- step 4: evaluation of the results obtained using the new set of parameters;
- step 5: comparison of results.

Since the Bayesian Optimization learns from the previous iterations is much more efficient than the other proposed methodologies.
Another important thing to define in the Bayesian Optimization is the objective function, so the function to be minimised. The main problem is that the shape of the objective function (or the probability distribution of the function) is unknown so the optimization must evaluate the model.
The main purpose of Bayesian Optimization is to build a probability model of the objective function. If the available resources were unlimited, it would be possible to identify every point of the objective function, but unfortunately, since the resources are always limited this is not

possible. For example, in the case presented in *Chapter 3* with an infinite amount of resources it is possible to use a very high number of measurements and thus obtain a model representation identical to reality. As it is possible to notice from *Chapter 3* this is not possible, in fact, limited measurements were carried out (indeed even the case with 6 measurements is quite unrealistic). So in the Bayesian Optimization using the limited measurements it is necessary to build a surrogate model used to approximate the real objective function. A surrogate model is then defined as the probability representation of the objective function. The problem now is how to select the samples, since they are limited. For this purpose it is necessary to build an *Acquisition Function* (that can be also called *Selection Function*). So the new sample is identified where the Acquisition Function reaches its maximum. In this way, it is possible to add a new sample to the measurements until the maximum number of iterations or the max time is reached. The steps to create a Bayesian Optimization Algorithm are the following:

- first of all it is necessary to build a domain of the hyperparameters that we want to explore;
- definition of the objective function. This function must obviously depend on hyperparameters;
- creation of a criterion of selection. Using this criterion the hyperparameters are chosen from the surrogate model;
- the previous iterations are used to update the surrogate model.

The logic behind the coupling between the method used to identify the model and the parameters with the RL technique is shown in the *Figure 2.5*.

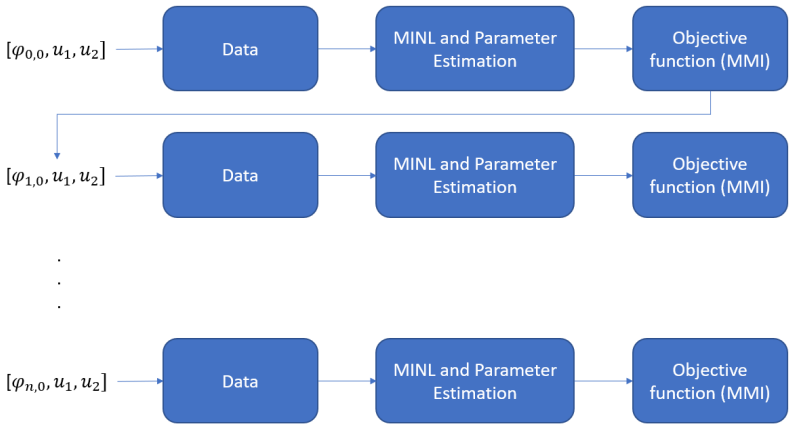

Figure 2.5. *Schematic representation of the proposed approach*

Where φ are the initial states at each experiment and u_1 and u_2 are the parameters chosen for each experiment. At the beginning of the process the initial guess on the conditions is given as input to the Bayesian Optimisation. From this guess the dataset is constructed using the

correct model (this is necessary as it is not possible to perform real experiments). Then, after the experimental measurements are obtained, the structure of the model and the parameters are identified and the statistics are calculated. At the end of all this, the objective function is calculated. Depending on the value of the objective function, the algorithm will return the new experimental conditions to be tested. This procedure is repeated a fixed number of times.

Chapter 3

Model identification using superstructure optimization methods

In this chapter, the first case study with the result obtained is presented. In this case study the first three steps of the procedure described in Section 2.1 (Figure 2.1) are applied. The first step is the experiment, through which the dataset is built. After obtaining the data from the experimental measurements it is necessary to proceed with the identification of the model and the estimation of the parameters (Step 2) and once obtained it is possible to calculate the statistics (Step 3).

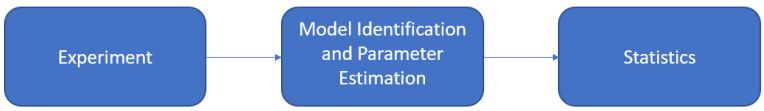

Figure 3.1. *Schematic representation of the proposed approach*

3.1 Experiment: Baker's yeast growth model

The system considered is a cultivation of yeast inside a fed-batch reactor. This system has been selected because the governing equations are already well known, and also the parameters inside the equations have already been determined with excellent precision. The system kinetic is described by the following set of differential and algebraic equations:

$$\begin{aligned} \frac{dx_1}{dt} &= \left(\frac{\theta_1 x_2}{\theta_2 x_1 + x_2} - u_1 - \theta_4 \right) x_1 \\ \frac{dx_2}{dt} &= -\frac{\theta_1 x_2 x_1}{(\theta_2 x_1 + x_2)\theta_3} + u_1(u_2 - x_2) \end{aligned} \tag{3.1}$$

The aim of the experimenter is to evaluate the dynamics of yeast concentration $x_1(t)$ $\left[gL^{-1}\right]$ and the substrate concentration x_2 $\left[gL^{-1}\right]$ as a function of two different inputs: the dilution factor, called u_1 $\left[h^{-1}\right]$ and the substrate concentration in the feed, u_2 $\left[gL^{-1}\right]$. In the following figure (*Figure 3.2*) the considered fed-batch system is represented.

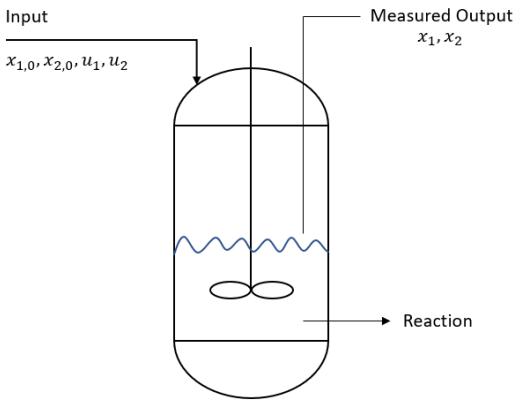

Figure 3.2. *Sketch of the fed-batch reactor*

From *Equation 3.1* it is possible to notice that there are four parameters in the model:

$$\theta = [\theta_1, \theta_2, \theta_3, \theta_4] \tag{3.2}$$

As the model is very well known the parameters have already been extensively studied and their values have been determined with very good precision. The values of the parameters are reported in Table 3.1.

Table 3.1. *Assumed "true" parameter values*

Parameter	**Value**
θ_1	0.310
θ_2	0.180
θ_3	0.550
θ_4	0.050

With the experiments, it is possible to sample a vector, $y = [x_1, x_2]^T$ of the system states. It is important to underline that the measurements are corrupted by Gaussian measurement noise. The experiments, in this first case study, are run with a fixed level of dilution factor and a fixed concentration in the feed (*Table 3.2*).

Table 3.2. *Inputs values*

Inputs	**Value**
u_1	0.05
u_2	5

As the dilution factor and the substrate concentration in the feed also the initial conditions are kept fixed in this first case study.
The model described above has been used to build the dataset. In fact, since in this case it is not possible to perform real experiments the in-silico experiments are performed using Equation 3.1 to describe the system.

Table 3.3. *Initial state values*

State$_0$	**Value** $\left[gL^{-1}\right]$
$x_1(0)$	1
$x_2(0)$	0.01

3.1.1 Superstructure definition

After defining the dataset using *Equation 3.1* it is necessary to define the superstructure. To make things more complex the superstructure used for this case study is composed of different models very similar to each other. The superstructure is composed of four different modes and is reported in *Equation 3.3*.

$$\begin{aligned}\frac{dx_1}{dt} &= \lambda_1\left(\frac{\theta_1 x_2}{\theta_2 x_1 + x_2} - u_1 - \theta_4\right)x_1 + \lambda_2\left(\frac{\theta_1 x_2}{\theta_2 + x_2} - u_1 - \theta_4\right)x_1 \\ \frac{dx_2}{dt} &= -\lambda_3\left(\frac{\theta_1 x_2 x_1}{(\theta_2 x_1 + x_2)\theta_3} + u_1(u_2 - x_2)\right) + \lambda_4\left(\frac{\theta_1 x_2 x_1}{(\theta_2 + x_2)\theta_3} + u_1(u_2 - x_2)\right)\end{aligned} \tag{3.3}$$

Modes one and two are very similar to each other and the same thing can be observed for modes three and four. These modes are taken from the paper of Asprey and Macchietto (Asprey et al., 2000). The difference between the modes is that modes one and three assume a Monod-type kinetic for the consumption of the substrate, instead, modes two and four assume Cantois-type kinetics. The modes are deliberately chosen similar to each other because if the algorithm manages to identify the appropriate model to describe the system this means that is very accurate. In *Equation 3.3* it is possible to notice that there are parameters indicated with the letter λ, those parameters are the integer values used in the Mixed Integer Non-Linear Problem to identify the real structure of the model starting from the superstructure. In this case, there are four different integers, one for each mode.
Another thing that is very important to underline is that in the MINLP there is only one constraint on the modes that is reported in *Equation 3.4*.

$$\sum_{i=1}^{N} \lambda_i \leq N \tag{3.4}$$

As it is possible to notice from *Equation 3.3* the sum of the integer values must be lower than N where N is the number of modes that constitute the model. So the total number of modes that are different from zero should be less or at last equal to the overall number of modes. Besides this, no constraints have been added to the number of modes that compose the model. In this way, the problem is more general because for example both modes one and two can be active at the same time, and the same applies to modes three and four.

3.2 Results

In the following section are reported the four different cases and the respective results:

- Scenario 1: low level of noise and high number of measurements;
- Scenario 2: low level of noise and low number of measurements;
- Scenario 3: high level of noise and high number of measurements;
- Scenario 4: high level of noise and low number of measurements.

3.2.1 Scenario 1: high number of measurements and low noise level

The first scenario, scenario 1, is the most ideal situation the experimenter can have. In fact, in this first case, the number of measurements is very high and the noise applied to the measurements is very low. The conditions of this first case are reported in *Table 3.4*. In the table above

Table 3.4. *Scenario 1: conditions*

Measurements	96
Noise	low

is reported the number of measurements, that is very high. The noise instead in this first case is very low and has a normal distribution with mean equal to zero and variance (σ) equal to the 10% of the higher value of the state (*Equation 3.5* and *Equation 3.6*)

$$\sigma_1 = 0.1 max(x_1) \tag{3.5}$$

$$\sigma_2 = 0.1 max(x_2) \tag{3.6}$$

where σ_1 is the variance of the error of the measurements of the state x_1 and σ_2 is instead the variance of the noise of the measurements of the state x_2.
In the following figures (*Figure 3.3* and *Figure 3.4*) the distribution of noise in the measurements of state one and state two is reported respectively.
These images confirm the Gaussian distribution of the noise, moreover for safety has been also calculated the mean of both distributions that are reported in *Table 3.5*, so even if it is not zero it is possible to affirm that is still a good approximation.

Table 3.5. *Scenario 1: noise mean*

μ	**Value**
μ_1	0.008
μ_2	0.004

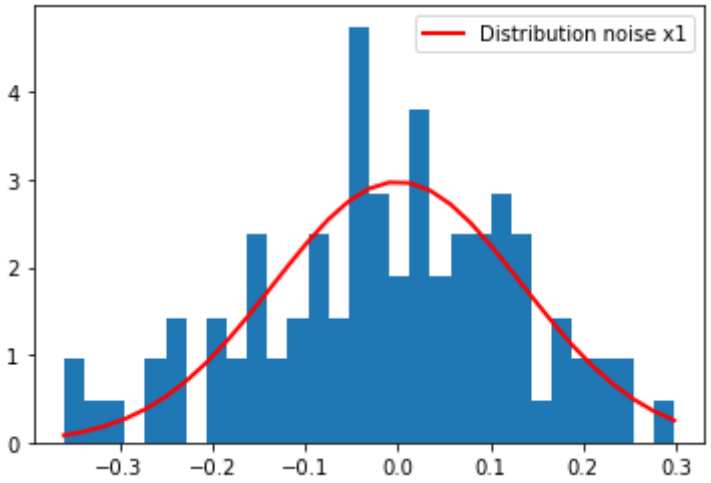

Figure 3.3. *Noise distribution for* x_1

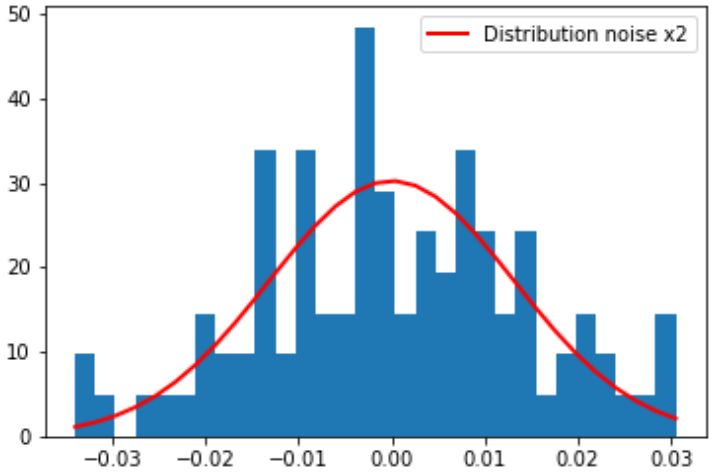

Figure 3.4. *Noise distribution for* x_2

In *Figure 3.5* and *Figure 3.6* are reported the predictions obtained using the superstructure compared to the measurements.

Figure 3.5 and *Figure 3.6* are composed by three different plot, the blue one represent the value of the experimental measurements in a noise-free ideal situation. The red dashed line is instead the real value of the measurement so with the noise. The last line, the green one, represents the trend of the states obtained using the algorithm. From these last two images it is possible to affirm that the predictions of the states using the model are very accurate, but to understand that it is necessary to analyze the prediction of the parameters and the statistics linked to the model.

First of all, it is necessary to understand if the algorithm managed to identify the right structure of the model, it is possible to see that looking at the integers evaluated by the algorithm. The values of the predicted integers are reported in *Table 3.6*. And using these values of the integers

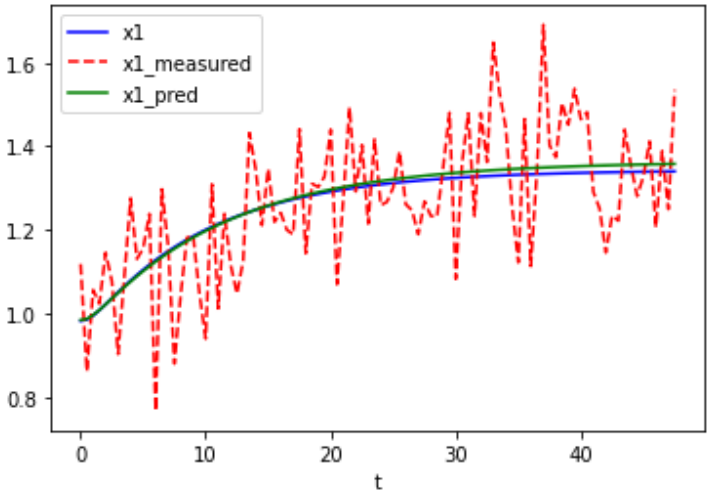

Figure 3.5. *Trend of the predicted x_1 and the measured x_1. The blue line represent x_1 without noise, the red dashed line represent the measured x_1 and the green line represent the predicted value of x_1*

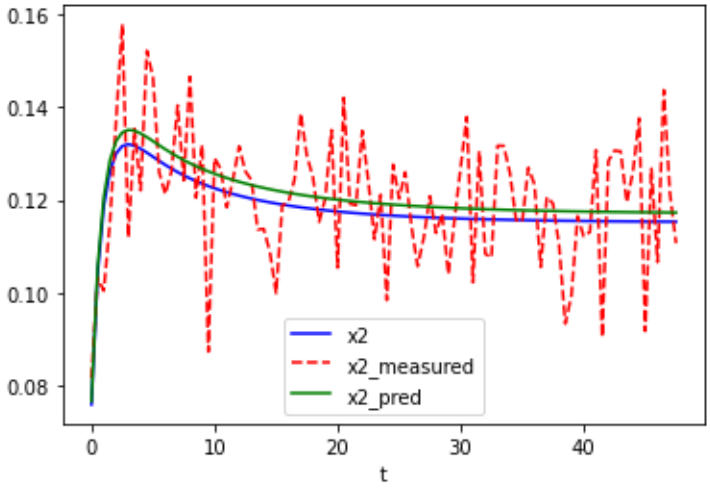

Figure 3.6. *Trend of the predicted x_2 and the measured x_2. The blue line represents x_2 without noise, the red dashed line represents the measured x_2 and the green line represent the predicted value of x_2*

it is possible to obtain the following equation for the model:

$$\begin{aligned} \frac{dx_1}{dt} &= \lambda_1 \left(\frac{\theta_1 x_2}{\theta_2 x_1 + x_2} - u_1 - \theta_4 \right) x_1 \\ \frac{dx_2}{dt} &= \frac{\theta_1 x_2 x_1}{(\theta_2 x_1 + x_2)\theta_3} + u_1(u_2 - x_2) \end{aligned} \tag{3.7}$$

that is the same model that has been used to generate the data. So it is possible to affirm that in scenario 1 the algorithm managed to identify the structure of the model. After these considerations, it is necessary to evaluate if the algorithm managed to identify also the values

Table 3.6. *Scenario 1: integers*

Integers	**Estimated Value**	**True Value**
λ_1	1	1
λ_2	0	0
λ_3	1	1
λ_4	0	0

of the parameters. The prediction obtained using the algorithm are reported in the following table. Comparing these values with the values reported in *Table 3.7* it is possible to notice that

Table 3.7. *Scenario 1: parameters prediction and variance*

Parameter	**Estimated Value**
$\hat{\theta}_1$	0.316 $\pm$0.05
$\hat{\theta}_2$	0.172 $\pm$0.03
$\hat{\theta}_3$	0.571 $\pm$0.025
$\hat{\theta}_4$	0.055 $\pm$0.006

the algorithm managed to evaluate the parameters with good precision but to be sure of this it is necessary to estimate the statistics of the parameters.
To estimate the statistics of the parameters it is necessary to define the initial guess for each parameter and also the upper and lower bounds.

Table 3.8. *Scenario 1: initial guesses and bounds*

Parameter	**Guess**	u_b	l_b
θ_1	0.3	0	0.5
θ_2	0.152	0	0.5
θ_3	0.6	0.5	1
θ_4	0.057	0	0.5

With the values in *Table 3.8* it is possible to build the initial Fisher Information matrix. Using *Equation 2.20* and *Equation 2.18* the Fisher Information matrix that is reported below is build.

$$\mathbf{M} = \begin{pmatrix} 2773.05 & 810.25 & 624.42 & 1140.37 \\ 810.25 & 4847.86 & 3696.57 & 3072.08 \\ 624.42 & 3696.57 & 4973.70 & 3048.45 \\ 1140.37 & 3072.08 & 3048.45 & 3937.77 \end{pmatrix} \tag{3.8}$$

As it is possible to notice from the *Equation 3.8* the matrix is symmetric, as it should be, and also the values inside it are very high so there is a high amount of information in the parameters.

The other test on the parameters is the t-test; in *Table 3.9* the results and the reference t-value are reported.

Table 3.9. *Scenario 1: t-test results*

Parameter ID	**Predicted value**	**95% t-value** t-ref = 1.69
θ_1	0.316	9.11
θ_2	0.172	4.29
θ_3	0.571	13.20
θ_4	0.055	2.17

From *Table 3.9* it is possible to notice that all the t-values of the parameters are greater than the reference value and then all the parameters pass the test.
After evaluating the parameter estimation it is necessary to evaluate how good is the fit of the model, so the Goodness of Fit test is performed. The results of the Goodness of Fit test are reported in *Table 3.10*. In this table is reported the value of the sum of squared residuals (SSR)

Table 3.10. *Scenario 1: goodness-of-fit test*

$\chi^2_{5\%}$	**SSR**	$\chi^2_{95\%}$
157.28	203.22	220.99

that is between the 5% and the 95% of the χ^2 value of reference and this means that the model fits the data in a good way, so there is not under or over-fitting.

Now it is necessary to evaluate the Model Modification Indexes to see if it is possible to improve the fitting of the model by changing its structure. The Model Modification Indexes of each parameter are reported in the following table (*Table 3.9*).

Table 3.11. *Scenario 1: Model Modification Indexes*

Parameter	θ_1	θ_2	θ_3	θ_4
MMI	0.38	0.24	0.05	0.85

So since all the Model Modification Indexes are lower than zero there is no statistical evidence that changing a parameter with a state function the fitting is improved.

3.2.2 Scenario 2: low number of measurements and low noise level

In this second scenario differently from Scenario 1, the number of measurements is drastically reduced to simulate a more real situation. Performing 96 measurements would be extremely expensive and for this reason, it is necessary to reduce the number of measurements.
In this second case, the noise is the same as the previous case but the number of measurements is reduced to 20. In the following table are summarized the conditions. To double-check that

Table 3.12. *Scenario 2: conditions*

Measurements	20
Noise	low

the normal distribution has a mean equal to zero the mean of the noise has been evaluated and the results are reported in *Table 3.13*.

Table 3.13. *Scenario 2: noise mean*

μ	**Value**
μ_1	0.008
μ_2	0.004

In the following figures (*Figure 3.7* and *Figure 3.8*) are reported the distribution of the noises in x_1 and in x_2. From these figures, it is possible to notice that the distribution of the noise

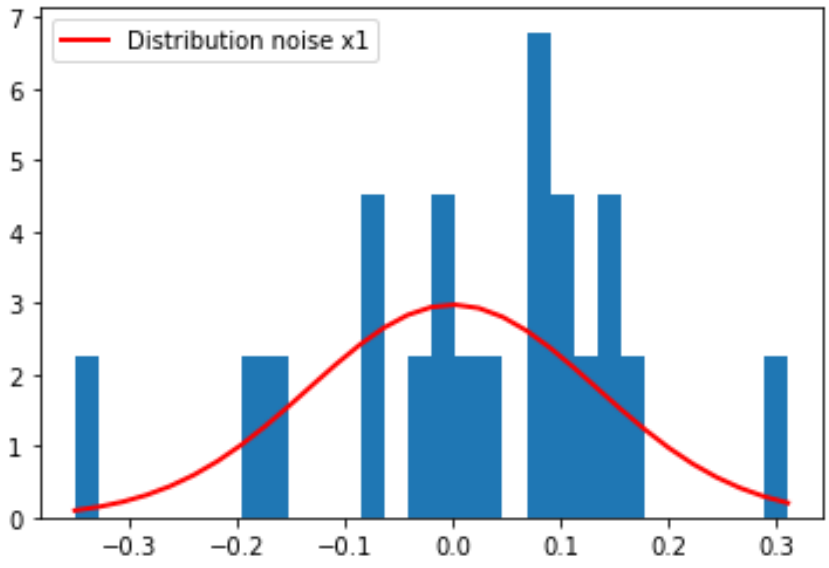

Figure 3.7. *Noise distribution for x_1*

is not a normal distribution, differently from scenario 1, but this is due to the low number of measurements.

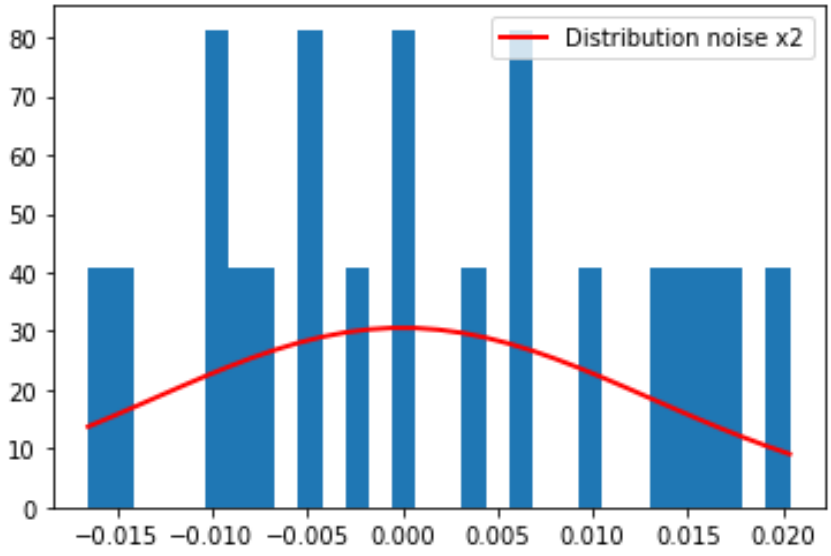

Figure 3.8. *Noise distribution for x_2*

In *Figure 3.9* and *Figure 3.10* are reported the trend of x_1 and x_2. As in the previous scenario, also in this one, the two plots are composed of different lines. It is possible to notice that in this

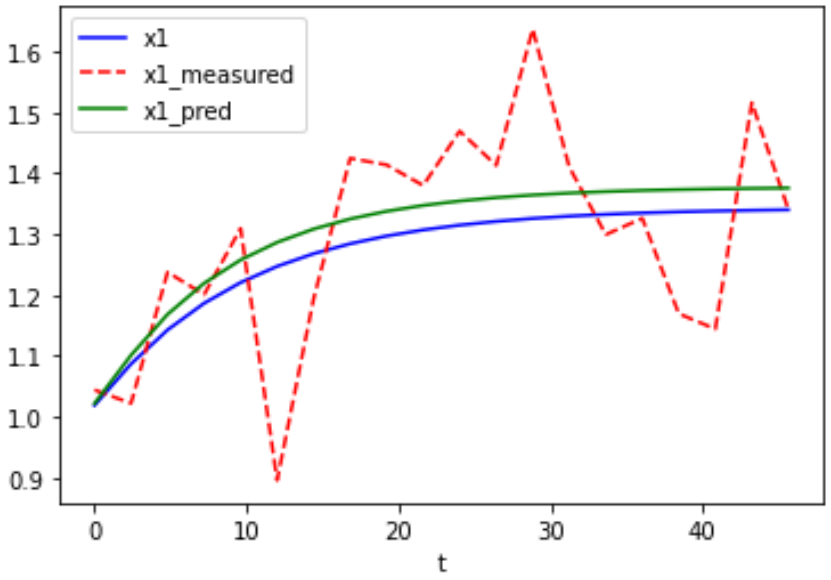

Figure 3.9. *Trend of the predicted x_1 and the measured x_1. The blue line represent x_1 without noise, the red dashed line represent the measured x_1 and the green line represent the predicted value of x_1*

case the predictions obtained using the model (green lines) do not overlap perfectly with the data obtained using *Equation 3.1*. This is due to the lower number of experiments. To evaluate if the fitting obtained using the algorithm is good enough it is necessary to evaluate the statistics of the model and the parameters. First of all, it is necessary to check if the algorithm manages to identify the right structure of the model and so if it manages to identify the right integers.

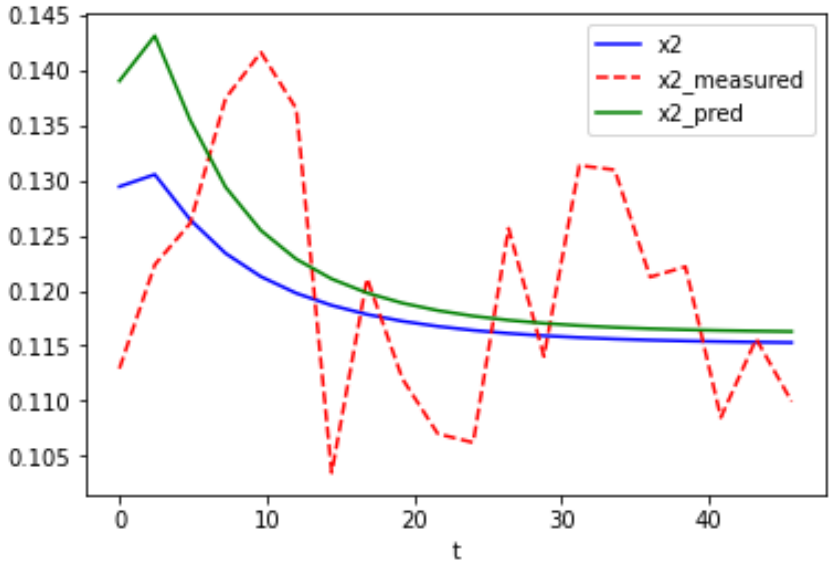

Figure 3.10. *Trend of the predicted x_2 and the measured x_2. The blue line represent x_2 without noise, the red dashed line represent the measured x_2 and the green line represent the predicted value of x_2*

In *Table 3.14* the predictions of the integers are reported and it is possible to notice that the algorithm manages to identify *Equation 3.1* in the superstructure.

Table 3.14. *Scenario 2: integers*

Integers	**Estimated Value**	**True Value**
λ_1	1	1
λ_2	0	0
λ_3	1	1
λ_4	0	0

Since the model is correctly identified now it is possible to evaluate the parameters with the corresponding variance. In *Table 3.15* the values of the parameters are reported.

Table 3.15. *Scenario 2: parameters prediction and variance*

Parameter	**Estimated Value**
$\hat{\theta}_1$	0.300 $\pm$0.03
$\hat{\theta}_2$	0.200 $\pm$0.02
$\hat{\theta}_3$	0.551 $\pm$0.05
$\hat{\theta}_4$	0.043 $\pm$0.01

Also in this case some constraints have been used on the initial guesses of the parameters. The constraints used are the same as scenario 1 and are reported in *Table 3.8*. Now it is possible to

evaluate the Fisher Information matrix:

$$\mathbf{M} = \begin{pmatrix} 4682.44 & -2922.80 & -1530.88 & 1187.09 \\ -2922.80 & 10848.72 & 4937.22 & 3361.29 \\ -1530.88 & 4937.22 & 3726.84 & 1689.99 \\ 1187.09 & 3361.29 & 1689.99 & 4243.59 \end{pmatrix} \tag{3.9}$$

Also in this case the information matrix is symmetric. Now it is necessary to evaluate if the predictions of the parameters are precise using the t-test. The results of the test are reported in *Table 3.16*.

Table 3.16. *Scenario 2: t-test results*

Parameter ID	**Predicted value**	**95% t-value** t-ref = 1.69
θ_1	0.300	8.27
θ_2	0.200	5.14
θ_3	0.551	9.68
θ_4	0.043	1.95

All the t-values of the parameters are higher than the reference t-value so it is possible to conclude that the parameters are evaluated with good precision.

Now is performed the Goodness of fit test on the model to evaluate the fitting of the data.

Table 3.17. *Scenario 2: goodness-of-fit test*

$\chi^2_{5\%}$	**SSR**	$\chi^2_{95\%}$
23.27	45.91	51.00

As it is possible to notice from *Table 3.17* the model passed the Goodness of fit test because the sum of square residual is between the χ^2 with 5% and 95% of confidence.

After the test on the parameters and the model, it is necessary to evaluate if it is possible to improve the fitting of the model by changing some parameters with a state function. The model modification indexes are:

Table 3.18. *Scenario 2: Model Modification Indexes*

Parameter	θ_1	θ_2	θ_3	θ_4
MMI	0.19	0.11	0.06	0.98

From *Table 3.18* it is possible to notice that the model can not be improved by changing the parameter with a state function.

3.2.3 Scenario 3: high number of measurements and high noise level

In the third case, the main difference with the previous scenarios is that in this case, the level of noise is much higher. In fact in this case the number of measurements is kept high but also the level of noise has been increased. In scenario 1 and scenario 2 the noise was very small (10%) of the higher value of the states). Instead in the third case, the noise is 50% of the higher measurement. The conditions are reported in the following table. Now predicting the real model

Table 3.19. *Scenario 3: Conditions*

Measurements	96
Noise	high

is more difficult because of the noise. The value of the noise has been set very high on purpose because if the algorithm manages to identify the structure of the model and the parameters in these conditions it will certainly succeed in normal conditions. Even if the noise level has been increased, it must still have a normal distribution with a mean equal, or very near, to zero (*Table 3.20*).

Table 3.20. *Scenario 3: Noise mean*

μ	**Value**
μ_1	0.07
μ_2	0.001

It is possible to notice from *Figure 3.11* and *Figure 3.12* that the noise has a normal distribution, both for x_1 and for x_2.

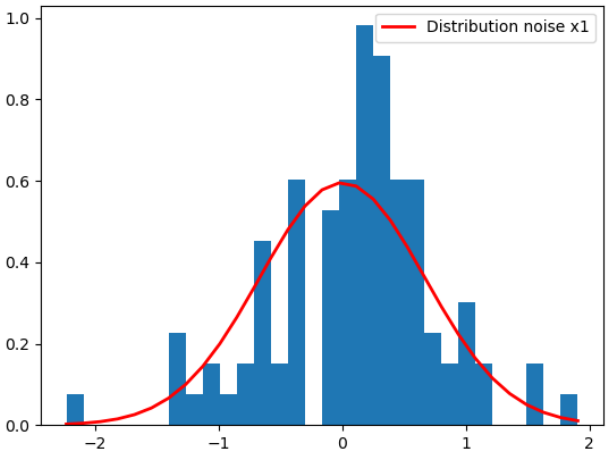

Figure 3.11. *Noise distribution for x_1*

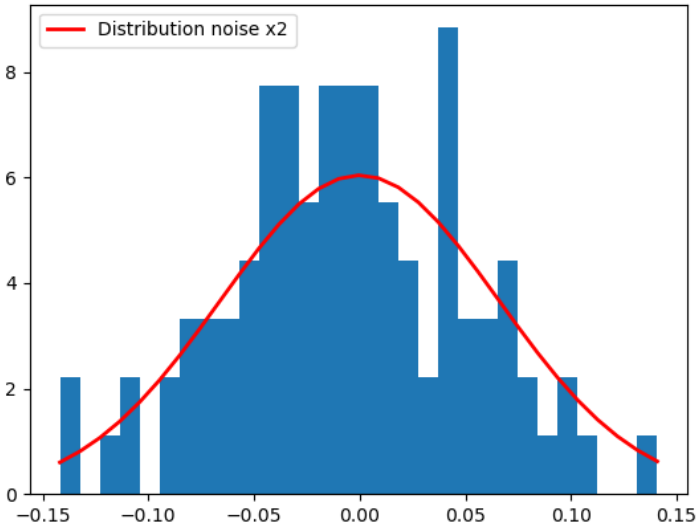

Figure 3.12. *Noise distribution for x_2*

Now it is possible to evaluate the trend of the predictions compared with the trend of the data obtained using the following figures. From *Figure 3.13* it is possible to notice that the noise is very high and the line representing the states predicted using the model is completely overlapped to the state obtained with the measurement (in this case using *Equation 3.1*). Also for the state x_2, it is possible to affirm that the model can still predict the states in a very good way but slightly worse respect x_1 as it is possible to notice that there is a little bias between the green line (measurement obtained using the model) and the blue line (experimental measurements). In *Table 3.21* are reported the integers values predicted by the algorithm. Using these integers

Table 3.21. *Scenario 3: integers*

Integers	**Estimated Value**	**True Value**
λ_1	1	1
λ_2	0	0
λ_3	1	1
λ_4	0	0

the following model is obtained, that is exactly the model reported in *Equation 3.1*.

$$\begin{aligned} \frac{dx_1}{dt} &= \lambda_1 \left(\frac{\theta_1 x_2}{\theta_2 x_1 + x_2} - u_1 - \theta_4 \right) x_1 \\ \frac{dx_2}{dt} &= \frac{\theta_1 x_2 x_1}{(\theta_2 x_1 + x_2)\theta_3} + u_1(u_2 - x_2) \end{aligned} \tag{3.10}$$

The parameters obtained using the MINLP are reported in *Table 3.22*.

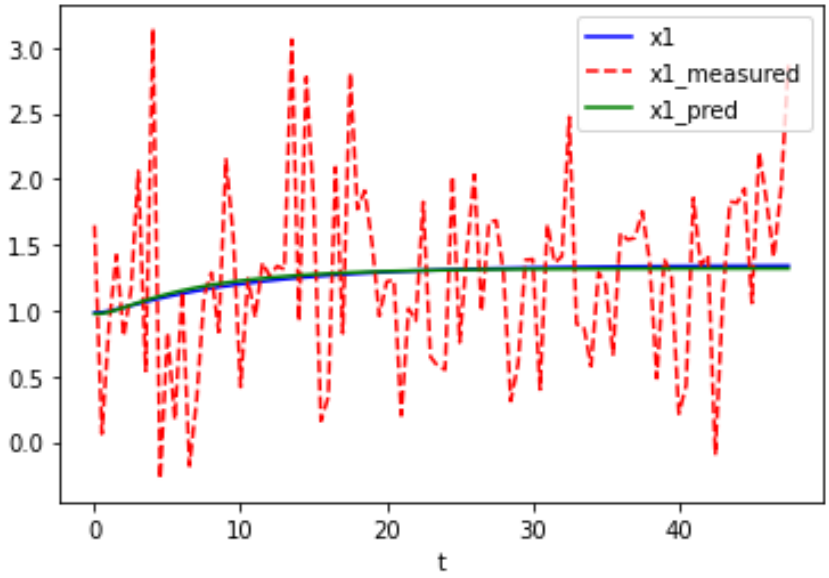

Figure 3.13. *Trend of the predicted x_1 and the measured x_1. The blue line represent x_1 without noise, the red dashed line represent the measured x_1 and the green line represent the predicted value of x_1*

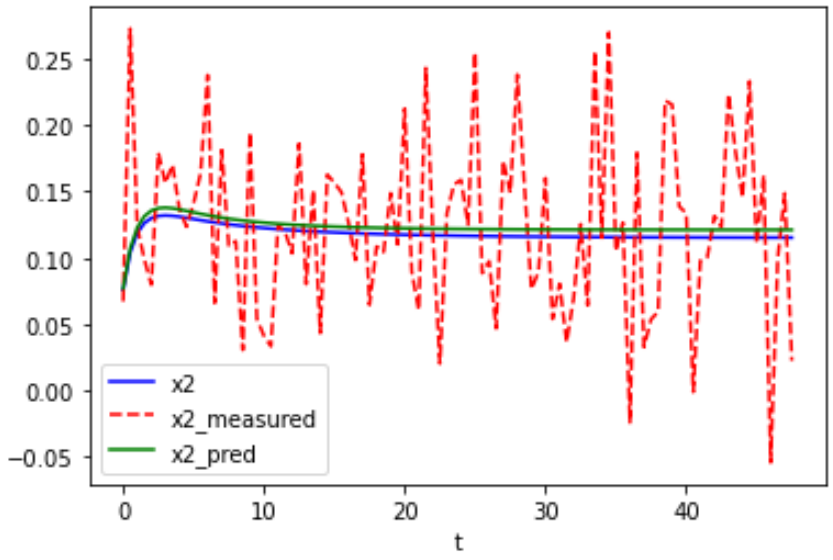

Figure 3.14. *Trend of the predicted x_2 and the measured x_2. The blue line represent x_2 without noise, the red dashed line represent the measured x_2 and the green line represent the predicted value of x_2*

These approximations are similar to the real values reported in *Table 3.1* but are worse respect the predictions obtained in scenario 1 and scenario 2. These results were predictable because of the noise.

Table 3.22. *Scenario 3: Parameters prediction*

Parameter	Estimated Value
$\hat{\theta}_1$	0.356 ±0.08
$\hat{\theta}_2$	0.200 ±0.07
$\hat{\theta}_3$	0.600 ±0.07
$\hat{\theta}_4$	0.051 ±0.008

The Fisher Information Matrix of this case is:

$$\mathbf{M} = \begin{pmatrix} 1287.04 & 54.30 & 45.84 & 70.17 \\ 54.30 & 1372.22 & 171.08 & 148.03 \\ 45.84 & 171.08 & 1370.88 & 144.32 \\ 70.17 & 148.03 & 144.32 & 1335.13 \end{pmatrix} \tag{3.11}$$

Also here it is possible to notice that the values inside the matrix are a lot smaller with respect to the values in the Information Matrices of scenario 1 and scenario 2 and also this is due to the noise. To check if the estimations of the parameters are still correct despite the noise it is necessary to perform the t-test on the parameters:

Table 3.23. *Scenario 3: t-test results*

Parameter ID	Predicted value	95% t-value t-ref = 1.69
θ_1	0.356	6.46
θ_2	0.200	3.70
θ_3	0.600	11.12
θ_4	0.051	1.77

Also, in this case, all the t-values of the parameters are greater than the reference t, but in this case, differently from the other the t-value of θ_4 is only slightly higher than the t-reference and this is due to the noise.

From the Goodness of fit test, it is possible to determine that the model obtained with the MINLP starting from the superstructure fits the data in a good way to avoid under-fitting and over-fitting. In fact, from *Table 3.24* it is possible to notice that also in this case the Sum of Square Residual is between the $\chi^2(5\%)$ and $\chi^2(95\%)$.
Also in this case as in the previous ones, it is necessary to evaluate if it is possible to improve the fitting of the model by changing the structure of the model. The evaluations of the Model Modification Indexes are reported in the following table.

Table 3.24. *Scenario 3: goodness-of-fit test*

$\chi^2_{5\%}$	**SSR**	$\chi^2_{95\%}$
157.28	202.92	220.99

Table 3.25. *Scenario 3: Model Modification Indexes*

Parameter	θ_1	θ_2	θ_3	θ_4
MMI	0.21	0.17	0.034	0.059

So also in this case, since the values of the MMIs are lower than 1, there is no statistical evidence that it is possible to improve the fitting of the model by changing the parameter of the model with a function of the states.

3.2.4 Scenario 4: low number of measurements and high noise level

In the fourth and last scenario of this first case study are used the worst conditions possible, so the number of measurements is reduced to 20 measurements and there is a high level of noise. Due to the conditions reported in *Table 3.26* this case is the most complicated because it is

Table 3.26. *Scenario 4: Conditions*

Measurements	20
Noise	high

not trivial at all to identify the structure of the model and the respecting parameters using only twenty measurements and corrupted by a very high level of noise. If the algorithm succeeds to identify the structure of the model and evaluating the values of the parameters with good precision in these conditions this means that this method is very solid and can work even in very complex conditions.
Also in this case it is necessary to check the distribution of the noise, that are reported in the following figures.

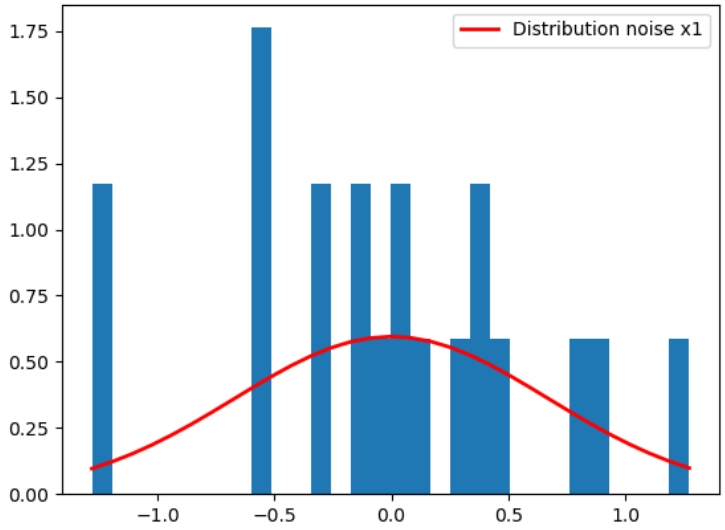

Figure 3.15. *Noise distribution for* x_1

From *Figure 3.15* and *Figure 3.16* it is possible to notice that as in Case 2, the noise does not follow a Gaussian distribution, but also in this case this is due to the low number of measurements.

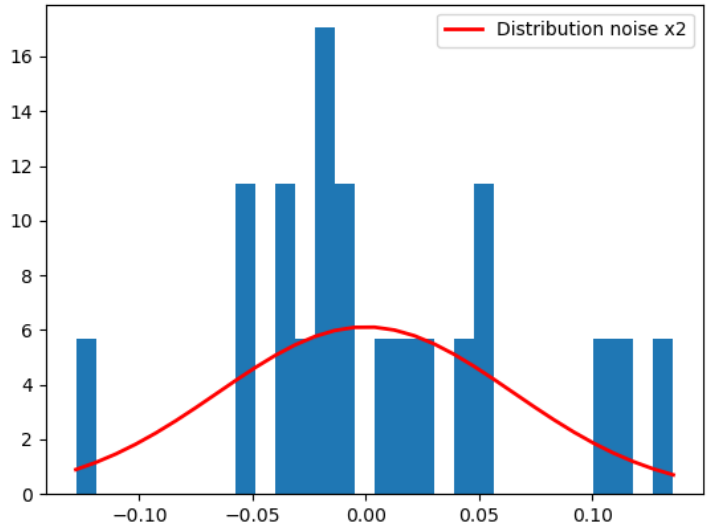

Figure 3.16. *Noise distribution for x_2*

The mean of the noise is reported in *Table 3.14*. So also in this case even if the mean isn't perfectly equal to zero can be considered a good approximation. In the following figures are

Table 3.27. *Scenario 4: noise mean*

μ	**Value**
μ_1	-0.016
μ_2	0.007

shown the trends of the state x_1 and of the state, x_2 compared with the experimental data. The graphical results of this simulation are worse compared to the results obtained in the other simulations because the level of noise is the same but the number of measurements is four-time smaller. In *Figure 3.17* the trend of the prediction of the state x_1 is reported (green line) and it is possible to affirm that it is not precise because the predictions are quite far from the experimental measurements (blue line).
In *Figure 3.18* x_2 is reported. Also in this case, as for the state x_1, the predictions are not accurate, but differently from the other state in this case at higher t the predicted states approach the data obtained from experimental measurements. The model extracted from the superstructure is the following one:

$$\begin{aligned} \frac{dx_1}{dt} &= \left(\frac{\theta_1 x_2}{\theta_2 + x_2} - u_1 - \theta_4\right) x_1 \\ \frac{dx_2}{dt} &= \frac{\theta_1 x_2 x_1}{(\theta_2 x_1 + x_2)\theta_3} + u_1(u_2 - x_2) \end{aligned} \tag{3.12}$$

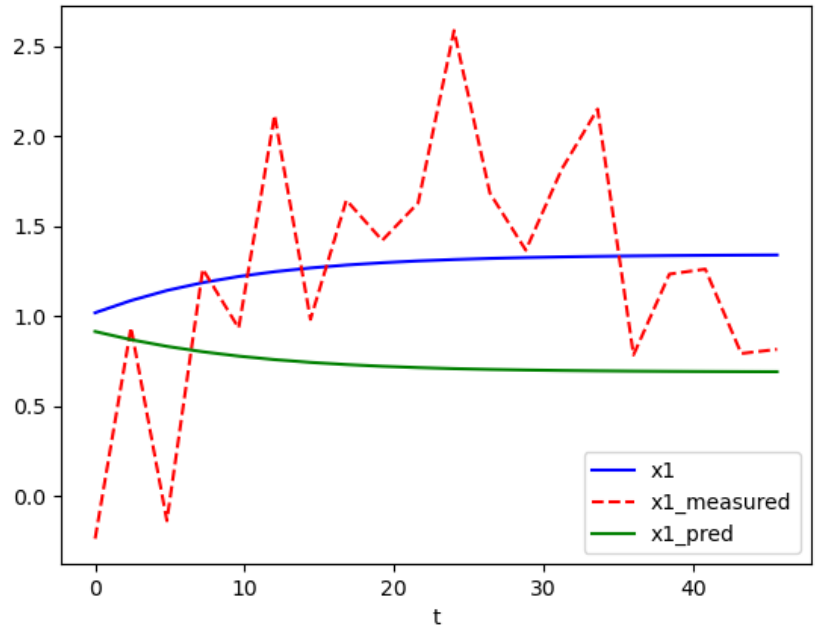

Figure 3.17. *Trend of the predicted x_1 and the measured x_1. The blue line represent x_1 without noise, the red dashed line represent the measured x_1 and the green line represent the predicted value of x_1*

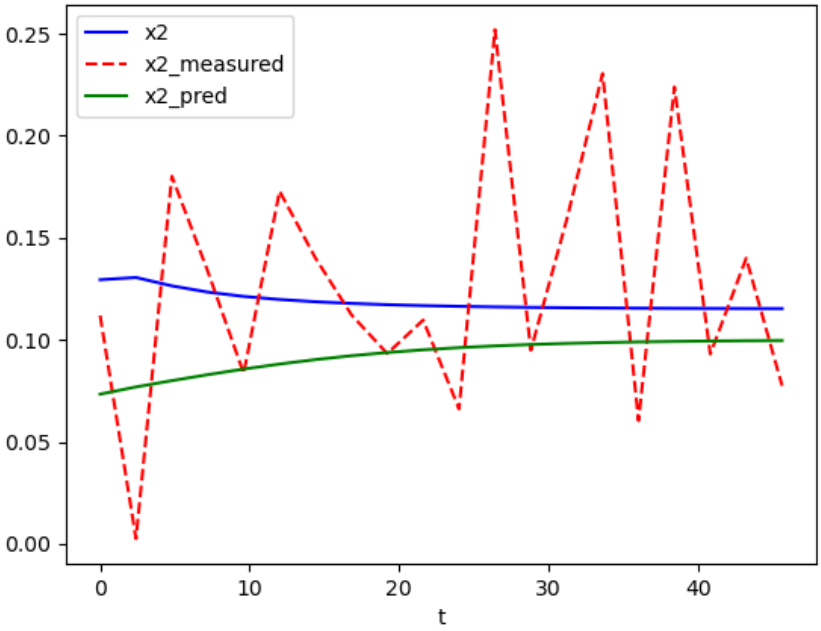

Figure 3.18. *Trend of the predicted x_2 and the measured x_2. The blue line represent x_2 without noise, the red dashed line represent the measured x_2 and the green line represent the predicted value of x_2*

so the integers values are:

Table 3.28. *Scenario 4: integers*

Integers	**Estimated Value**	**True Value**
λ_1	0	1
λ_2	1	0
λ_3	1	1
λ_4	0	0

The model reported in *Equation 3.12* is different from the model used to generate the data (*Equation 3.1*) and this can be noticed also looking at the identified integers. Looking at *Table 3.28* it is possible to notice that the active modes are the second and the fourth. The wrong identification of the model structure is the cause of the bad fitting of the data that can be seen in *Figure 3.16* and *Figure 3.17*.
The parameters identified using the Mixed Integer Non-Linear Problem, and their respective variance are the following:

Table 3.29. *Scenario 4: parameters prediction*

Parameter	Value
$\hat{\theta}_1$	0.300 $\pm$0.01
$\hat{\theta}_2$	0.100 $\pm$0.075
$\hat{\theta}_3$	0.500 $\pm$0.057
$\hat{\theta}_4$	0.100 $\pm$0.05

The values of the parameters, in this case, are slightly different from the values reported in *Table 3.1*, and this is because the algorithm identifies the wrong structure of the model. The variance of the parameters is still very low.

The Fisher Information Matrix is:

$$\mathbf{M} = \begin{pmatrix} 1404.69 & -465.19 & -281.27 & -111.17 \\ -465.19 & 2432.52 & 728.00 & 350.26 \\ -281.27 & 728.00 & 1631.46 & 202.19 \\ -111.17 & 350.26 & 202.19 & 1314.73 \end{pmatrix} \tag{3.13}$$

The values inside the FIM are larger than the values of the FIM of case 3 but are lower than the values in the FIM of the first two cases.

In the *Table 3.30* are reported the results of the t-test performed on the predictions of the parameters of the model.

Table 3.30. *Scenario 4: t-test results*

Parameter ID	**Predicted value**	**95% t-value** t-ref = 1.69
θ_1	0.300	5.533
θ_2	0.100	2.19
θ_3	0.500	9.20
θ_4	0.101	1.74

All the parameters passed the t-test. Now it is necessary to evaluate the performance of the model and check if the model fits the data in a good way. Since the identified model is different from the model used to generate the data it is already possible to hypothesize that the model fail the Goodness of fit test. The results of the test are reported in the following table.

Table 3.31. *Scenario 4: goodness-of-fit test*

$\chi^2_{5\%}$	**SSR**	$\chi^2_{95\%}$
23.27	56.88	51.00

From *Table 3.31* it is possible to confirm the hypothesis, in fact in this case the identified model failed the test because the sum of square residual is higher than the $\chi^2(95\%)$, so the model is falsified for under-fitting.
The fact that the model failed the test is positive because this means that the algorithm even though it failed to recognise the appropriate model, still gives us the indication that the identified model is not good.
The last things to check are the MMIs:

Table 3.32. *Scenario 4: Model Modification Indexes*

Parameter	θ_1	θ_2	θ_3	θ_4
MMI	0.19	0.11	0.068	0.12

All the MMI are lower than one so there is no statistical reason to change the parameters with a function of the states to improve the fitting.

From *Table 3.28* it is possible to notice that in this case, the algorithm identify the wrong model, this is possible because the number of measurement is low and the level of noise is very high. Results show that the model is not adequate to represent the process, as illustrated by the failed goodness-of-fit test.

3.2.5 Scenario 4.1: low number of measurements and high noise level with an additional constraint on the MINLP

In the previous section, it is possible to notice that the identified integers values are different from the real ones (*Table 3.28*) and so the identified model is not the model used to create the

dataset (*Equation 3.1*).
The only way to solve this problem is to act on the constraints of the Mixed Integer Non-Linear problem. As reported in *Equation 3.4*, the only constraint on the integers is that the sum of integers must always be less than the number of modes. But what happens if this constraint is changed in:

$$\sum_{i=1}^{N} \lambda_i \leq n_{eq} \tag{3.14}$$

where n_{eq} is the number of equations that make up the model. This constraint assures that only one mode for each equation is active. In the following pages are reported the results obtained using this more binding condition.

In *Figure 3.19* and *Figure 3.20* are reported the distribution of the noise in x_1 and x_2.

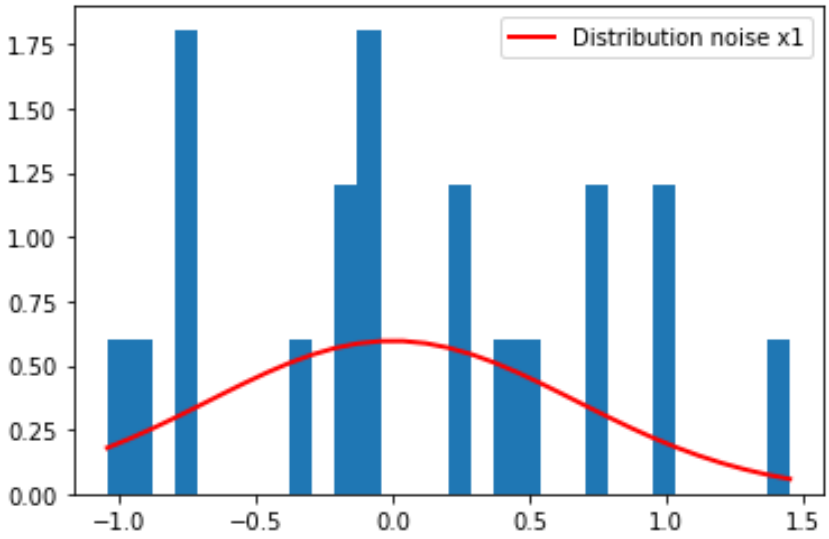

Figure 3.19. *Noise distribution for x_1.*

The distribution of the noise is very similar to the distribution of the noise in the second scenario. The average error is reported in *Table 3.33*

Table 3.33. *Scenario 4.1: noise mean*

μ	**Value**
μ_1	0.25
μ_2	0.01

The trend of the states are reported in *Figure 3.21* and *Figure 3.22* From these figures, it is possible to notice that with the more strict conditions the trend of the predicted states is very similar to the trend of the experimental measurement, so the fitting of the model is improved. From *Table 3.34* it is possible to notice that the algorithm succeeds to identify the right integers and so consequently identifying the right structure of the model.

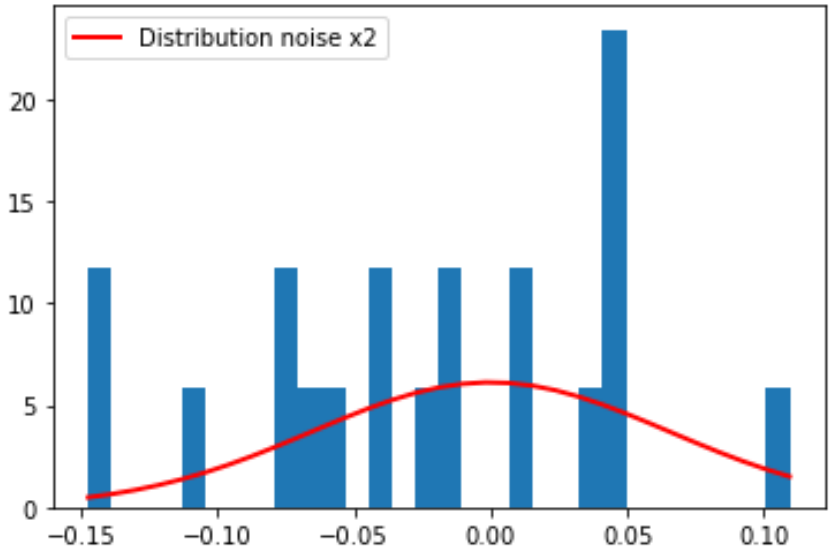

Figure 3.20. *Noise distribution for x_2.*

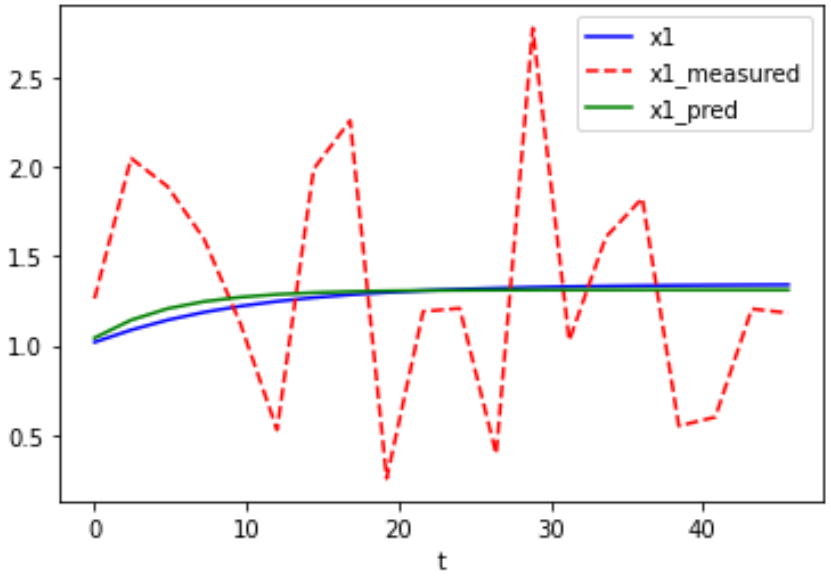

Figure 3.21. *Trend of the predicted x_1 and the measured x_1. The blue line represent x_1 without noise, the red dashed line represent the measured x_1 and the green line represent the predicted value of x_1*

The identified model is the following one:

$$\begin{aligned} \frac{dx_1}{dt} &= \left(\frac{\theta_1 x_2}{\theta_2 x_1 + x_2} - u_1 - \theta_4 \right) x_1 \\ \frac{dx_2}{dt} &= \frac{\theta_1 x_2 x_1}{(\theta_2 x_1 + x_2)\theta_3} + u_1(u_2 - x_2) \end{aligned} \tag{3.15}$$

The values of the parameters identified by the algorithm are reported in the following table. The

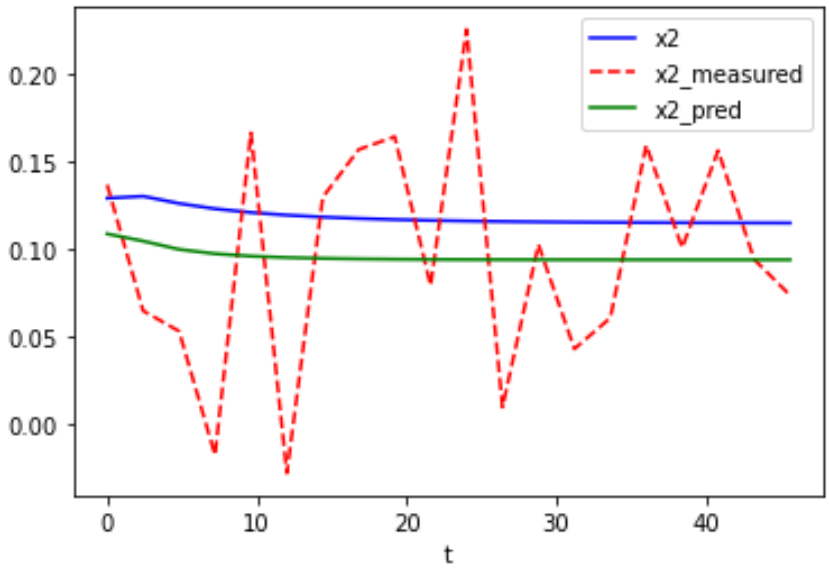

Figure 3.22. *Trend of the predicted x_2 and the measured x_2. The blue line represent x_2 without noise, the red dashed line represent the measured x_2 and the green line represent the predicted value of x_2*

Table 3.34. *Scenario 4.1: integers scenario*

Integers	**Estimated Value**	**True Value**
λ_1	1	1
λ_2	0	0
λ_3	1	1
λ_4	0	0

Table 3.35. *Scenario 4.1: parameters prediction and variance*

Parameter	**Estimated Value**
$\hat{\theta}_1$	0.323 $\pm$0.007
$\hat{\theta}_2$	0.200 $\pm$0.02
$\hat{\theta}_3$	0.600 $\pm$0.05
$\hat{\theta}_4$	0.040 $\pm$0.01

predictions of the values of the parameters, in this case, are more alike to the values reported in *Table 3.1*, and also the variance is of the same order of magnitude as the variances in the other cases. The Fisher Information Matrix is:

$$\mathbf{M} = \begin{pmatrix} 1496.16 & -85.62 & -28.02 & 166.73 \\ -85.62 & 1779.35 & 325.12 & 289.95 \\ -28.02 & 325.12 & 1383.86 & 177.52 \\ 166.73 & 289.95 & 177.52 & 1500.05 \end{pmatrix} \tag{3.16}$$

In *Table 3.* are reported the results of the t-test on the parameters and it is possible to notice that in this case the t-value of the parameter θ_4 is lower than the reference value but this can be due to the low number of measurement and to the high level of noise.

Table 3.36. *Scenario 4.1: t-test results*

Parameter ID	**Predicted value**	**95% t-value** t-ref = 1.69
θ_1	0.323	6.09
θ_2	0.200	4.01
θ_3	0.600	10.72
θ_4	0.040	1.74

Now it is necessary to verify if, differently from the case with the more general constraint, the model can fit the data in a good way. The results of the performed Goodness of fit test are reported in the following table.

Table 3.37. *Scenario 4.1: goodness-of-fit test*

$\chi^2_{5\%}$	**SSR**	$\chi^2_{95\%}$
23.27	37.10	51.00

In this case, the Sum of Square Residual is lower than the $\chi^2(95\%)$ and higher than the $\chi^2(5\%)$ so the model passed the test and fitted the data in a good way. Now the model Modification Indexes are evaluated to understand if it is possible to improve the fitting of the model

Table 3.38. *Scenario 4.1: Model Modification Indexes*

Parameter	θ_1	θ_2	θ_3	θ_4
MMI	0.085	0.058	0.037	0.062

From this table, it is possible to notice that all the MMI are very low so also in this case a change of a parameter with a state function is not justified by a statistical test.

3.3 Considerations

Chapter 3 reports four experiments performed at different noise level conditions and measurements. These four cases are summed in *Table 3.39*.

Table 3.39. *Cases*

	High N° of measurement	**Low N° of measurement**
Low level of noise	Case 1	Case 2
High level of noise	Case 3	Case 4

Below are some considerations for each case.

- Case 1: the first case is the most informative one because there is a high number of measurements and a low level of noise. In this case from *Table 3.6* and *Table 3.7*, it is possible to see that good results have been obtained, the values of the predicted parameters are very near to the value of the real parameters and the variance of the value is very small. Moreover, the model structure passed the Goodness-of-fit test. So in this case is very straightforward to find the appropriate model to describe the data.

- Case 2: in this case, there are the same conditions as Case 1 as regards noise but the number of measurements is reduced to 20 measures. Also in this case, the results obtained are very good because the algorithm manages to identify the right structure of the model and also a good approximation of the value of the parameters.

- Case 3: in this case, there are again 96 measurements but the level of noise is increased. The predicted parameters (*Table 3.22*) are very similar to the real parameters. The other statistics are similar to the test of Case 1 and also the model structure passed the Goodness-of-fit-test.

- Case 4: this was the most complex case. In fact, in the beginning, the algorithm did not succeed to identify the right structure of the model, and consequently, the model failed the Goodness of Fit test. This problem has been solved by applying a more binding constraint on the integer values. In this way, the algorithm managed to identify the right structure of the model and then to identify the values of the parameters. It is crucial to emphasise the fact that when the algorithm failed to identify the appropriate model it was possible to notice that the model was wrong looking at the Goodness of Fit test.

In this chapter, the goal was to show how this technique succeeds in identifying the model structure and its parameters under different noise conditions. In this case the experimental conditions (the values of u_1, u_2, $x_{1,0}$ and $x_{2,0}$) are kept constant and are not optimized.

Chapter 4

Reinforcement learning-based approach to model identification

4.1 Introduction

In this chapter, the coupling between the method proposed in the previous chapter and Reinforcement Learning is reported. In the first part of this chapter an example of exploration of the experimental space is reported, in which an algorithm of RL is not used. In this example, the experimental space is defined and then it is explored taking random experiments without exploiting the knowledge gained from each experiment.

4.2 Definition of design space

In this section, an example of exploration of the experimental space is disclosed. The algorithm used is the one proposed in the previous chapter. The parameters that will vary are the initial yeast concentration ($x_{1,0}$) and the dilution factor (u_1). Once the variables have been defined, it is necessary to define the bounds of the experimental space. The boundaries conditions shown in the *Table 4.1* are taken from the paper of Asprey et al. (2000).

Table 4.1. *Bounds of the experimental space*

	Upper Bound	Lower Bound
x_1	10	0
u_1	0.2	0.01

It is now necessary to define a number of experiments to explore the experimental space. To do this a Latin Hypercube for preliminary experiments is used. The necessary number of preliminary experiments was set equal to 20.
Differently from the method proposed in the previous chapter, in this case a function has been defined that is able to tell if the structure of the identified model is the correct one. This function is:

$$obj = \frac{\sum_{i=1}^{N_\theta} MMI_i}{MMI_{max}} \tag{4.1}$$

where N_θ is the number of parameters evaluated and MMI_{max} is the highest MMI.

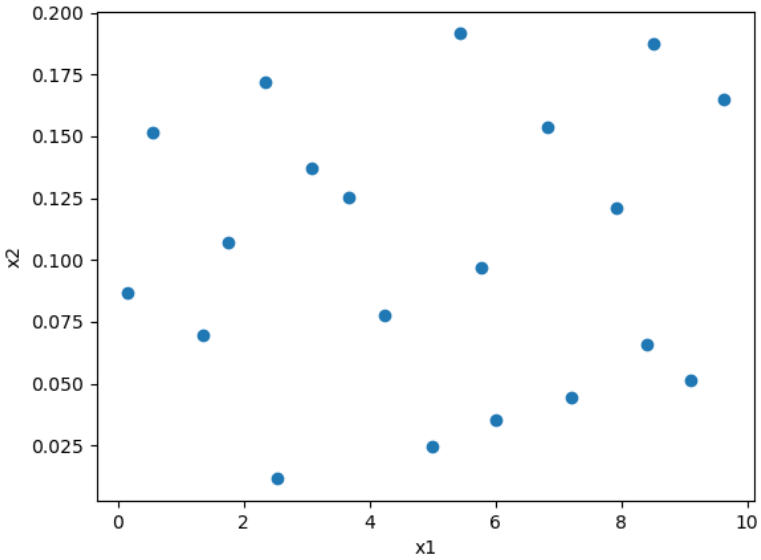

Figure 4.1. *Experiments to explore the experimental space.*

From *Figure 4.1*, where the selected initial experimental conditions are shown, it can be seen that the points are distributed in such a way as to cover the whole experimental space almost uniformly. This reduces the risk of leaving large portions of space "unexplored". Furthermore, using the 'maximin' method in the Latin Hypercube, the minimum distance between the points is maximized so that the points are not too close together (Le Guiban et al., 2018). Once the conditions have been defined, experiments can begin.

4.2.1 Explorative design of experiments

The procedure used to explore the experimental space is a random procedure. The conditions of the first experiment are chosen randomly among those proposed, the experiment is performed and the results are evaluated. Once the experiment is completed the conditions are removed from the possible ones (so that they cannot be reused) and the conditions for the next experiment are selected. Using this iterative procedure all the points shown in *Figure 4.1* were tested. Obviously, the experiments were carried out keeping the number of measurements in the dataset and the noise level constant. The standard deviation of the noise was set equal to 15% of the maximum value of the measured states and the number of measurements was set equal to 10. It can be seen that the number of measurements in the dataset is very small (half of the number in scenario 4 of Chapter 3) and the noise level is realistic. In the algorithm of exploration at each iteration the initial guess on the parameters is updated with the values identified in the previous guess and also the matrix of the initial information is equal to the matrix of Fisher evaluated in the previous iteration. In the next section, the obtained results and the considerations are reported.

4.2.2 Results of the Explorative DoE

In the following figures the graphs with the trends of the identified parameters, the integers and the MMIs are shown.

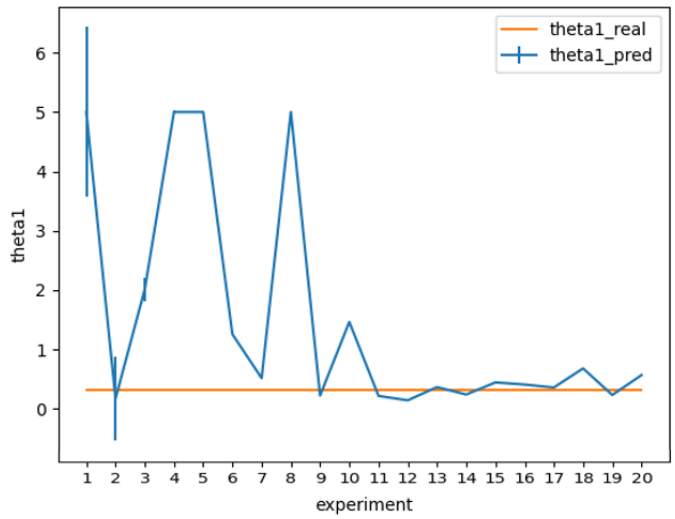

Figure 4.2. *Trend of the predicted parameter θ_1 and real value of θ_1. The blue line represents the predicted values, instead the orange line is the real value*

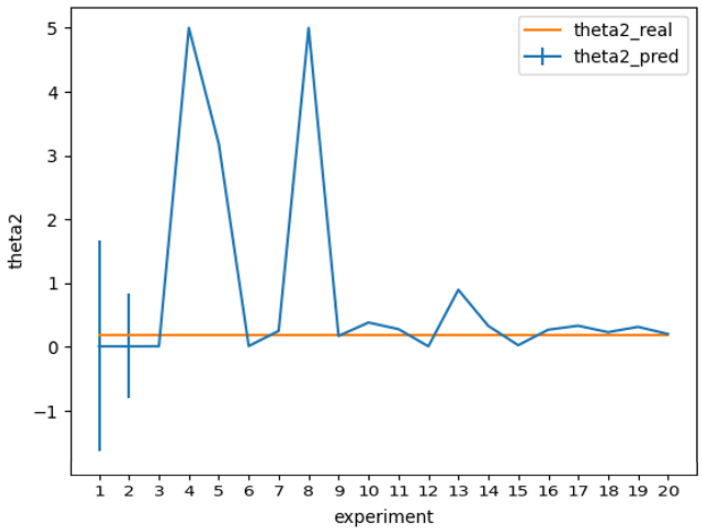

Figure 4.3. *Trend of the predicted parameter θ_2 and real value of θ_2. The blue line represents the predicted values, instead the orange line is the real value*

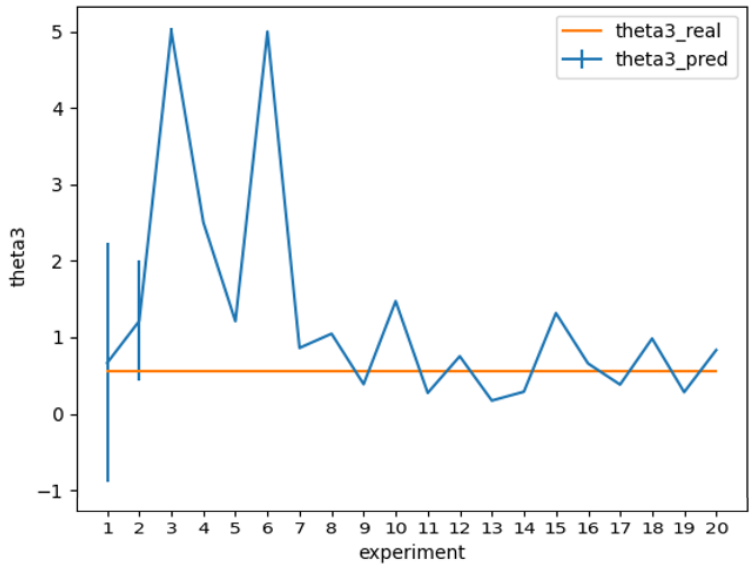

Figure 4.4. *Trend of the predicted parameter θ_3 and real value of θ_3. The blue line represents the predicted values, instead the orange line is the real value*

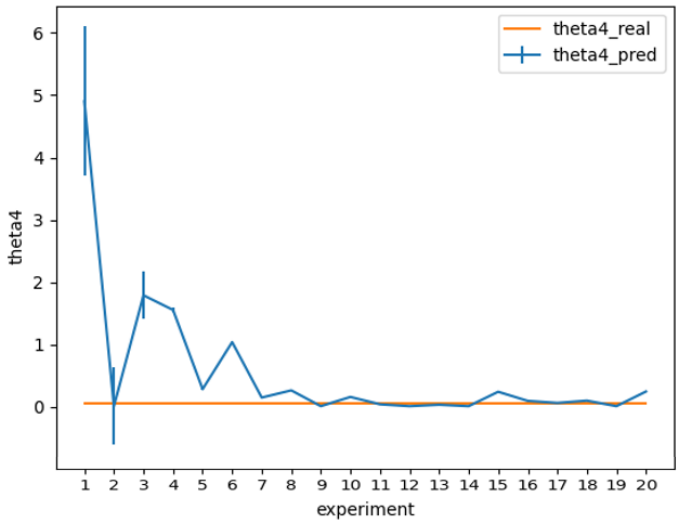

Figure 4.5. *Trend of the predicted parameter θ_4 and real value of θ_4. The blue line represents the predicted values, instead the orange line is the real value*

From these figures, it is possible to notice that in the first iterations, the predictions of the parameters are quite far from the value of the real parameters. The algorithm needs a certain number of experiments (about 8/9) to arrive at an acceptable approximation of the parameters. It can also be seen that as the experiments pass, the value of the identified parameters converges to the real value.

The next figures show the values of the integers. In *Figure 4.6* the first integer (λ_1) is reported. The value of λ_1 is always equal to 1 and therefore the algorithm is able to identify the exact mode of the model (this is why in the graph the blue and orange lines are perfectly superimposed). In the graph of the second integer (λ_2), the value of the integer is always equal to 0 (because when λ_1 is equal to 1 λ_2 is equal to zero and vice versa). The orange line (real value) and the blue line (predicted value) are perfectly overlapping.

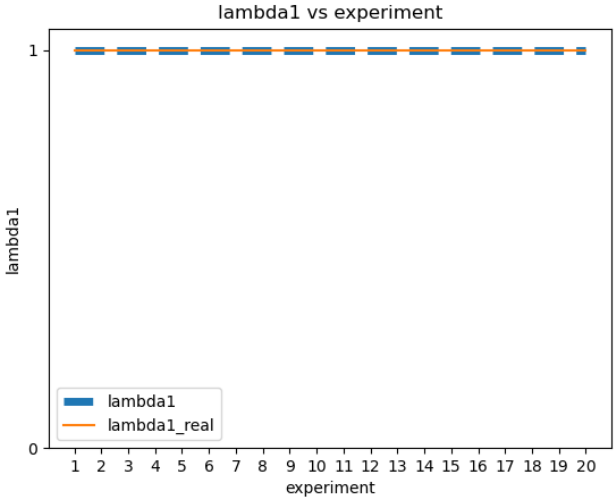

Figure 4.6. *Trend of the predicted integer λ_1 and real value of λ_1. The blue line represents the predicted values, instead the orange line is the real value*

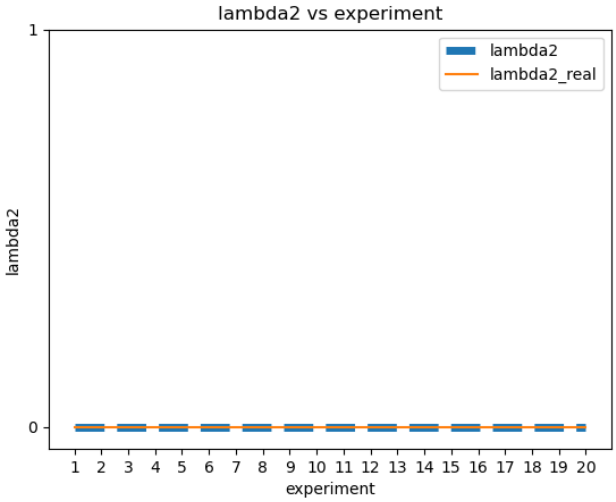

Figure 4.7. *Trend of the predicted integer λ_2 and real value of λ_2. The blue line represents the predicted values, instead the orange line is the real value*

In *Figures 4.7-8* λ_3 and λ_4 are constant (λ_3=1 and λ_4=0) and this means that the algorithm always manages to identify correctly the second equation of the system (*Equation 3.1*). So

also in this case the orange line (the real value of the integer) and the blue line are perfectly overlapping.

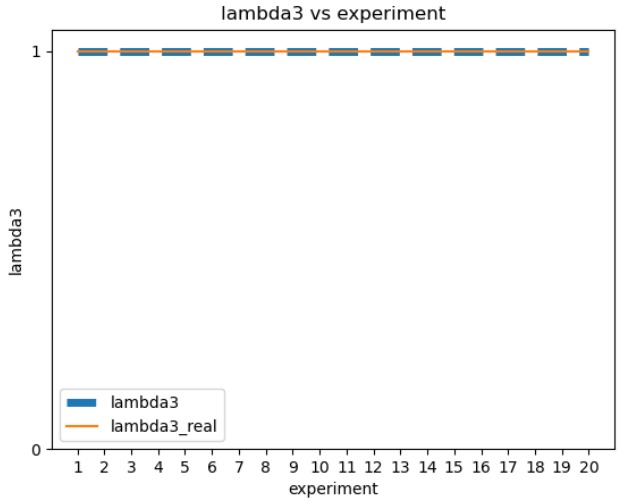

Figure 4.8. *Trend of the predicted integer λ_3 and real value of λ_3. The blue line represents the predicted values, instead the orange line is the real value.*

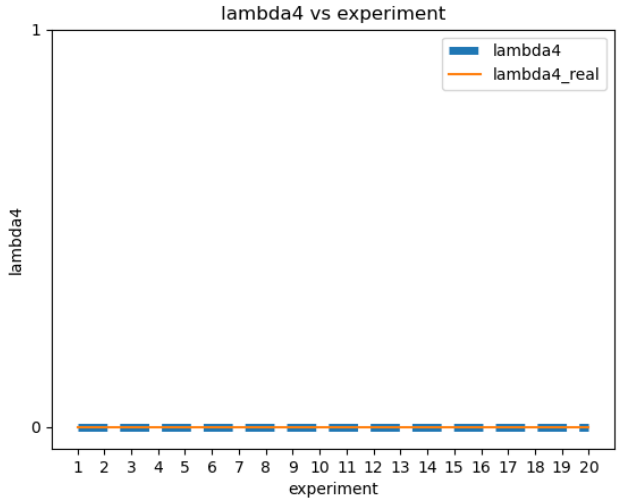

Figure 4.9. *Trend of the predicted integer λ_4 and real value of λ_4. The blue line represents the predicted values, instead the orange line is the real value.*

From *Figures 4.10-13*, it is possible to see that in many cases the MMIs are greater than 1 (orange line in the graphs). As it appears from results, in many experiments the algorithm was able to identify the correct model (as can be seen in *Figures 4.6-9*) and therefore theoretically these indices should be less than 1. A possible explanation for this fact could be that the MMIs need a higher number of experimental measurements to give reliable statistical information and that the 10 used are too few.

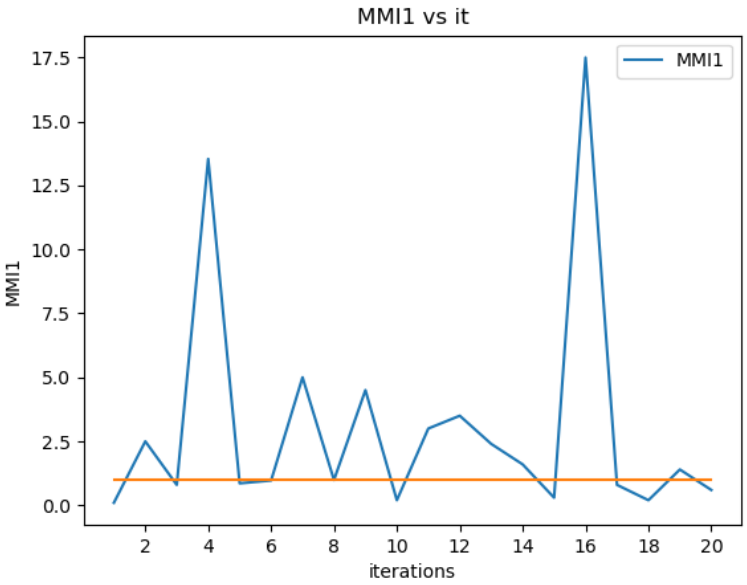

Figure 4.10. *Trend of MMI1. Orange line has constant ordinate equal to 1.*

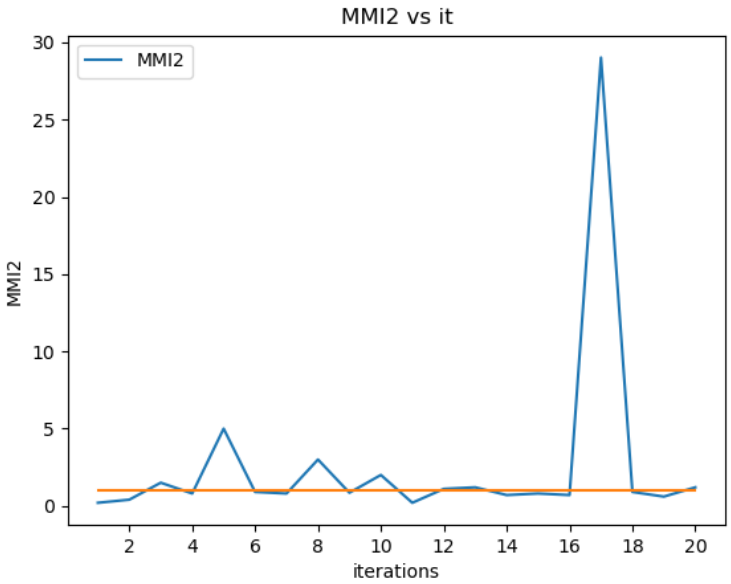

Figure 4.11. *Trend of MMI2. Orange line has constant ordinate equal to 1.*

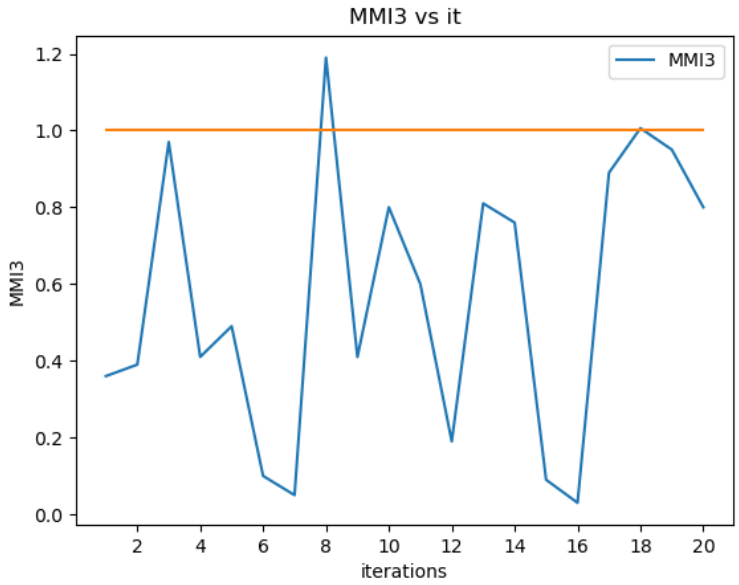

Figure 4.12. *Trend of MMI3. Orange line has constant ordinate equal to 1.*

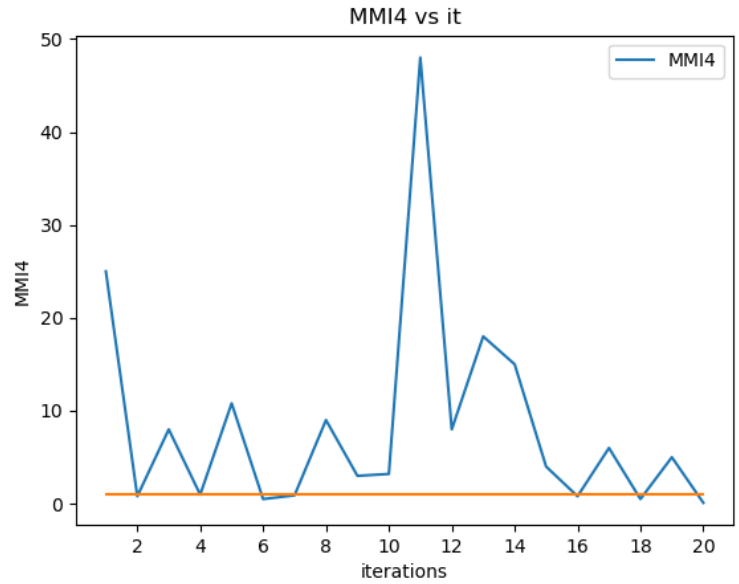

Figure 4.13. *Trend of MMI4. Orange line has constant ordinate equal to 1.*

To describe the performance of the identified models, (*Equation 4.1*) was used. The graph with the trend of the objective function is shown in the *Figure 4.14*.

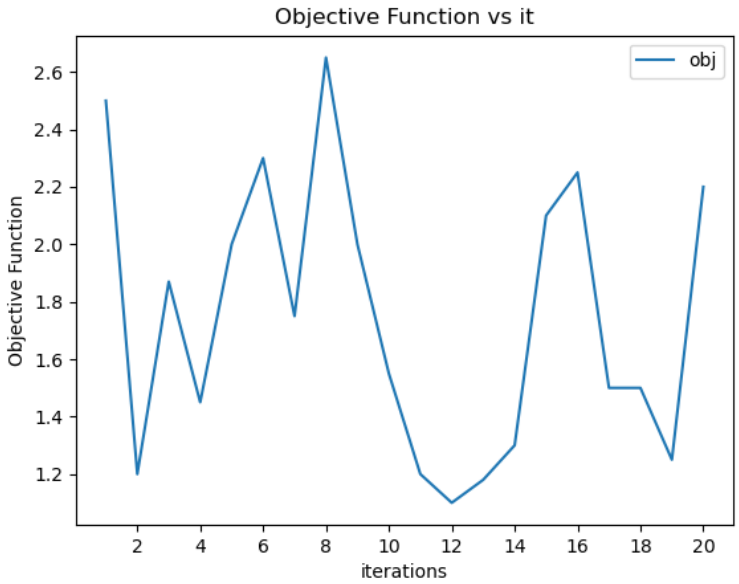

Figure 4.14. *Trend of the function that evaluates the prediction of the model structure (Equation 4.1)*

These results are also shown in correlation to the values of x_1 and u_1 in *Figure 4.15*.

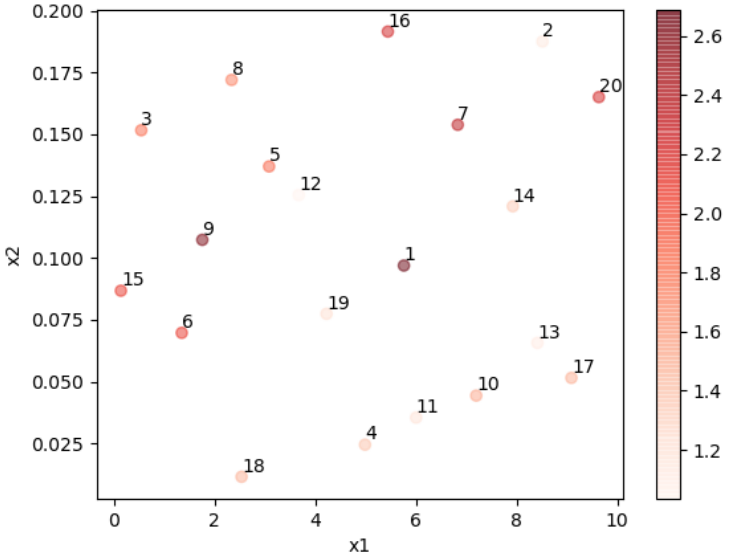

Figure 4.15. *Conditions of the experiments carried out. The colouring expresses the magnitude of the objective function (the darker the colour, the larger the objective function).*

From *Figure 4.14* it can be seen that the objective function does not have an increasing trend, but rather a random trend. This means that the performance of the experiment does not improve

with the number of experiments performed. This is because with the exploration *Equation 4.1* is not optimized and for this reason, the experimental conditions tested are not always the optimal ones to identify the structure of the model.

4.3 RL-based design of experiments

In the previous section, it was seen that with a purely exploratory approach, about 9 measurements are necessary to obtain a good approximation of the parameters. In this section, an exploiting part is added to the exploratory part by applying an RL algorithm. The algorithm used for this case study is a Bayesian optimisation (Chapter 2). The objective of this section is to compare the results obtained in the previous section with those obtained in this one. To do this, the experimental conditions are kept the same as those used in section 4.2 (*Table 4.2*). The bounds of the experimental space are also kept the same as in *Table 4.1*.
In contrast to the previous case, only the conditions (x_1 and u_1) of the first experiment are selected and the algorithm will optimize *Equation 4.1* that from now on will be called objective function. The RL algorithm will select the points and conditions of the following experiments according to the objective function. *Figure 4.16* shows all the conditions used for the experiments.

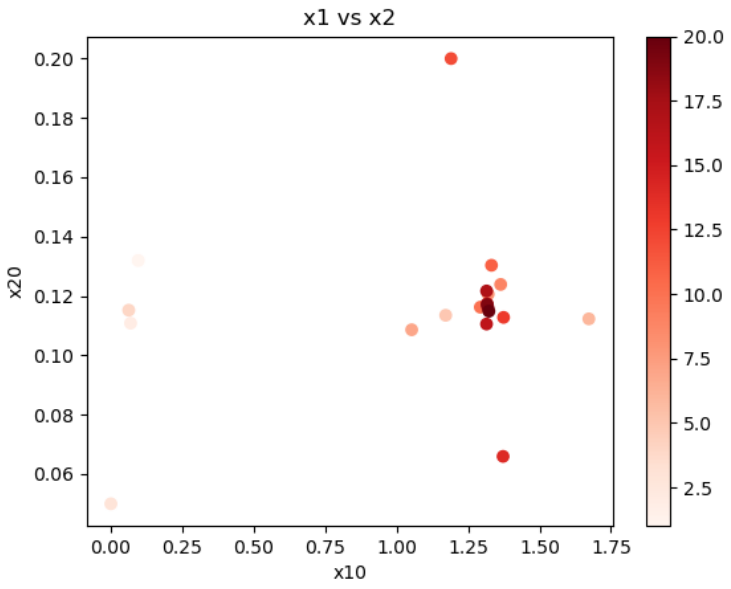

Figure 4.16. *Initial conditions of the experiments performed*

In *Figure 4.16* the experimental points are coloured according to the experiment number to which they correspond, the darker the shade of red the more recent the experiment. The interesting thing is the choice of initial experimental conditions. In fact, in this case, as can be seen from the figure, the points are not homogeneously distributed in the experimental space as in

the previous case. This is due to the exploitation part of the RL algorithm. If an experiment is carried out which returns a very high objective function, the algorithm will be inclined to select the experimental conditions around that point. However, if the optimal points are always exploited, the risk is to get stuck on a local and not global optimum point, failing to obtain the best possible conditions. The task to overcome is up to the exploratory part of the algorithm. As can be seen from *Figure 4.16*, some experiments are carried out at points far from the optimal points to explore the space to see if there are other optimal conditions for the experiment. The problem between exploration and exploitation has always been related to RL algorithms and it is not always possible to find a compromise between these two options. The following figures show the graphs comparing the trends of the identified parameters, integer, MMI and objective function in the case of the RL algorithm and the exploratory algorithm.

4.3.1 Parameter Estimation

From the *Figures 4.17-20* it can be seen that the value of the parameters identified by the RL algorithm (green line) converges to the real value (orange lines) much faster than the value found by the exploratory algorithm (blue line).
In addition, looking at these images it can be seen that in the parameters identified by the RL algorithm the peaks are absent, which are instead present in the parameters identified by the exploration algorithm (as in *Figure 4.17* the parameter θ_1 identified in experiment 10).

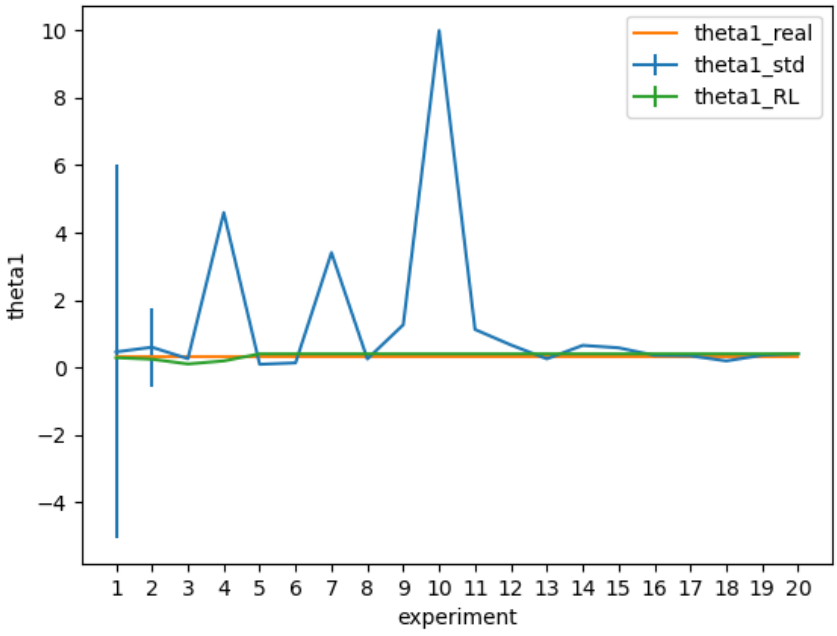

Figure 4.17. *Trend of the predicted θ_1. The green line represents the values identified by the RL algorithm, the blue line represents the values found by the exploratory algorithm and the orange line is the true value of the parameter.*

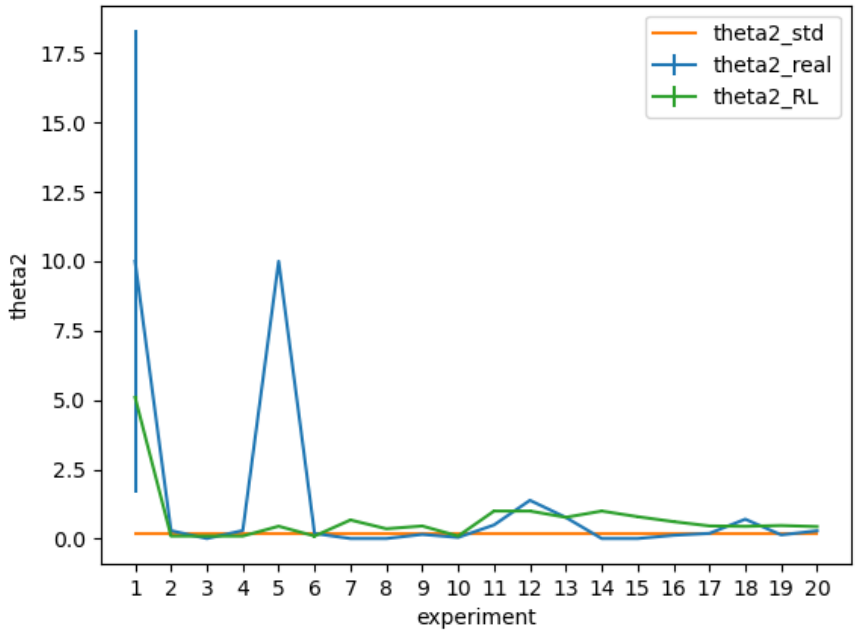

Figure 4.18. *Trend of the predicted θ_2. The green line represents the values identified by the RL algorithm, the blue line represents the values found by the exploratory algorithm and the orange line is the true value of the parameter.*

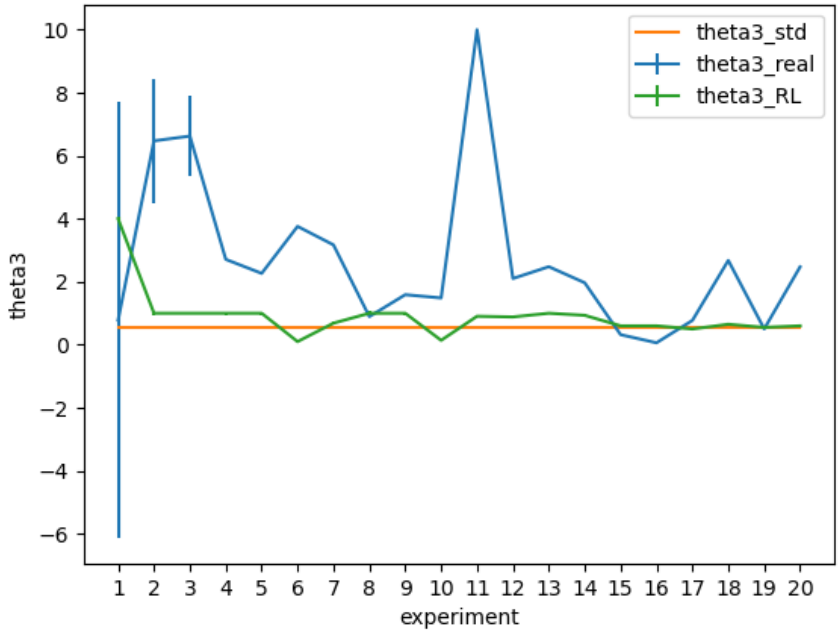

Figure 4.19. *Trend of the predicted θ_3. The green line represents the values identified by the RL algorithm, the blue line represents the values found by the exploratory algorithm and the orange line is the true value of the parameter.*

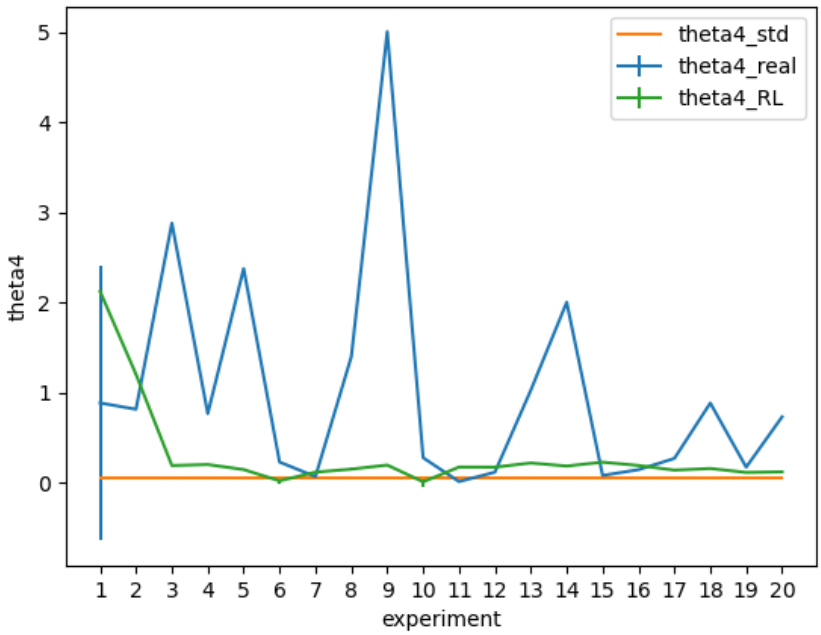

Figure 4.20. *Trend of the predicted θ_4. The green line represents the values identified by the RL algorithm, the blue line represents the values found by the exploratory algorithm and the orange line is the true value of the parameter.*

4.3.2 Model Structure Identification

As far as the identification of the structure of the model is concerned, it can be seen from *Figures 4.21-24* that even in this case there are no difficulties in identifying the real model. Only in the second experiment the algorithm identifies the wrong model and then immediately identified the right value of the integer in the following experiment.

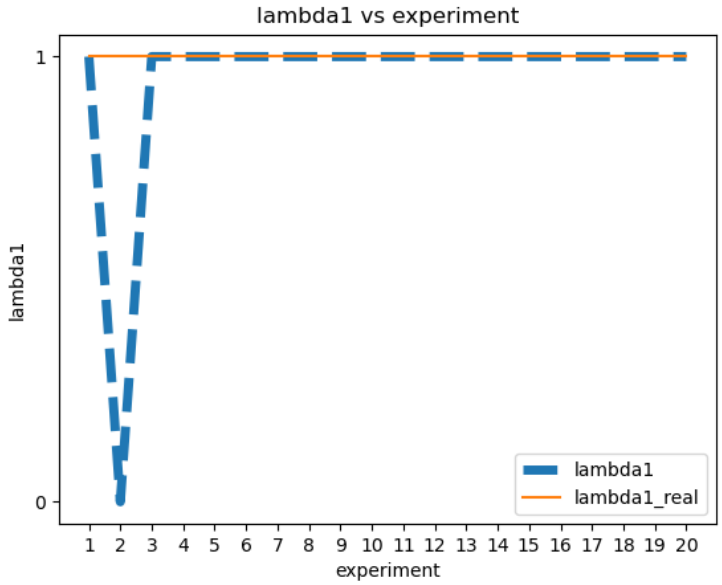

Figure 4.21. *Trend of the predicted* λ_1.

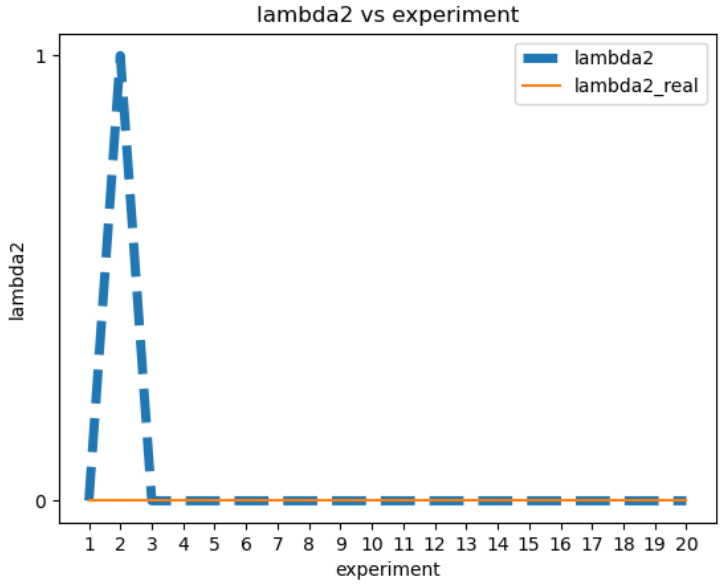

Figure 4.22. *Trend of the predicted* λ_2.

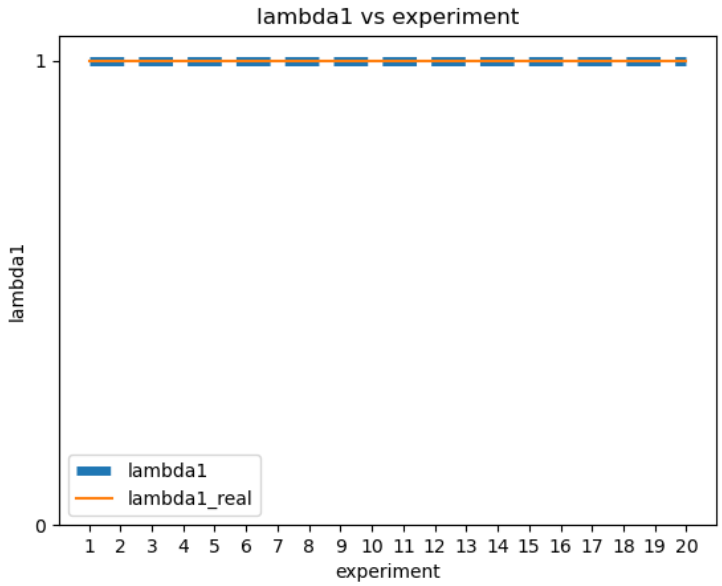

Figure 4.23. *Trend of the predicted* λ_3.

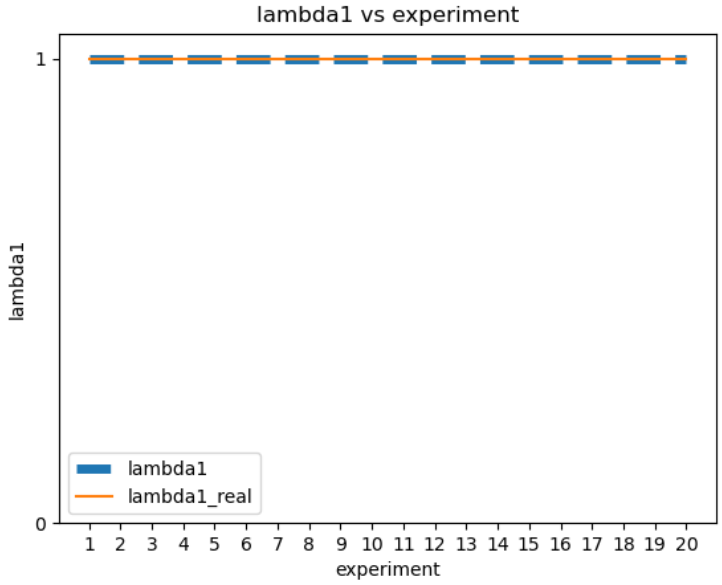

Figure 4.24. *Trend of the predicted* λ_4.

4.3.3 Model Modification Indices

After identifying the structure of the model and the values of the parameters, the RL algorithm calculates the Model Modification Indices. The values of the identified MMIs are reported in the following figures. It can be seen that the values of the MMIs are much better than the values obtained with the exploration algorithm. In fact, only the values found in the first two experiments are greater than one. The reason for this difference lies in the fact that RL

exploits the points that have a high value of the objective function (*Equation 4.1*) to select the experimental conditions of the next experiment. The points exploited are therefore those in which it is succeeded to obtain one better approximation of the model and the parameters. For this reason, the MMIs identified with this methodology are better than those identified with the exploratory algorithm.

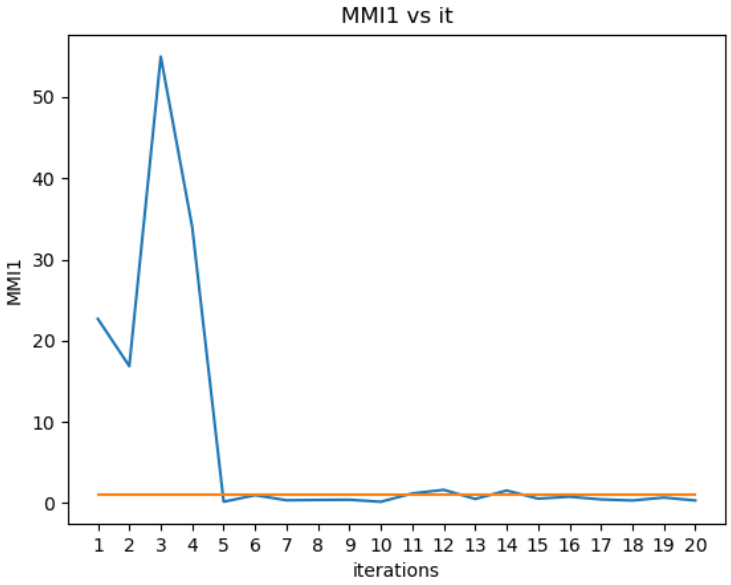

Figure 4.25. *Trend of the predicted MMI_1.*

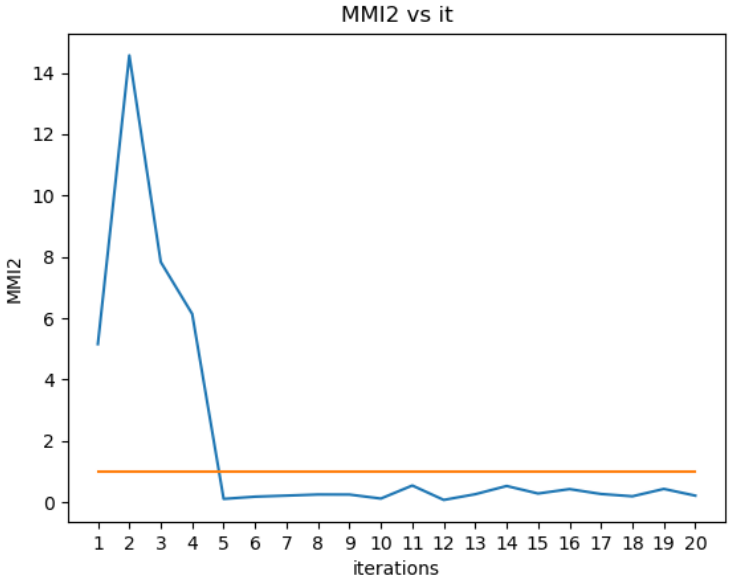

Figure 4.26. *Trend of the predicted MMI_2.*

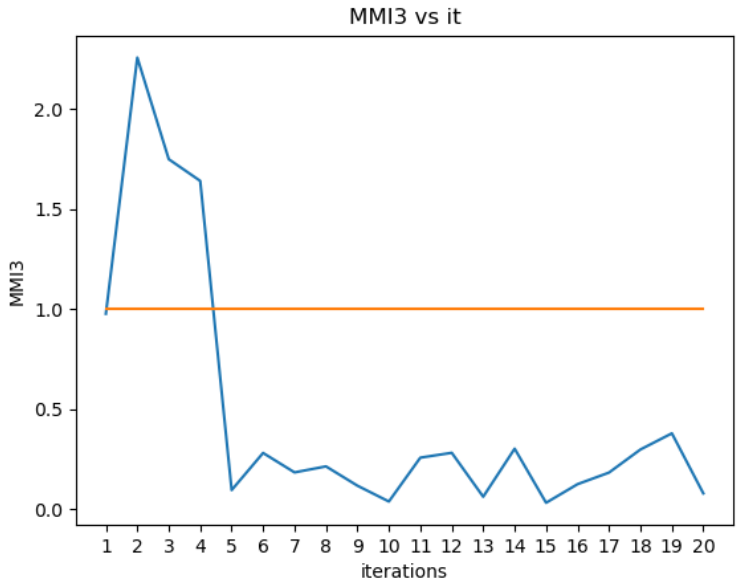

Figure 4.27. *Trend of the predicted MMI_3.*

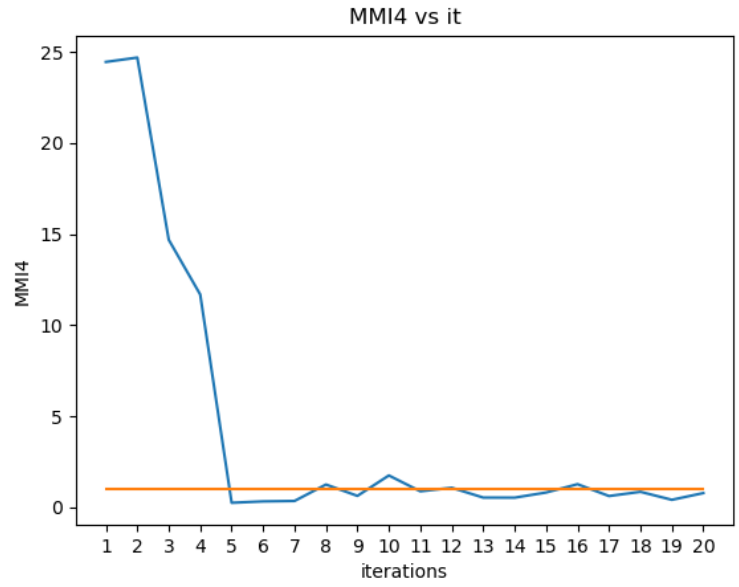

Figure 4.28. *Trend of the predicted MMI_4.*

Another interesting thing is shown in the *Figure 4.29*. From this picture, we can see the trend of the yeast concentration (x_1) in the various experiments. It can be seen that the trend of x_1 in the first experiments varies a lot while from the eighth experiment onwards it stabilises on a value. This is because in the first experiments the algorithm explored the space and then exploited the optimal points found.

In *Figure 4.30*, where the value of the dilution factor (u_1) is shown in the various experiments, a different trend can be seen. In fact, the value of u_1 for most of the experiments oscillates around

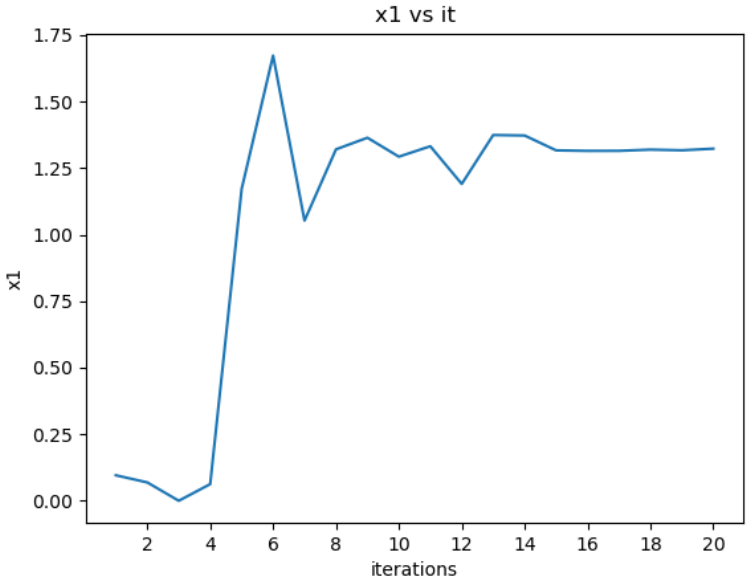

Figure 4.29. *Trend of the selected* x_1.

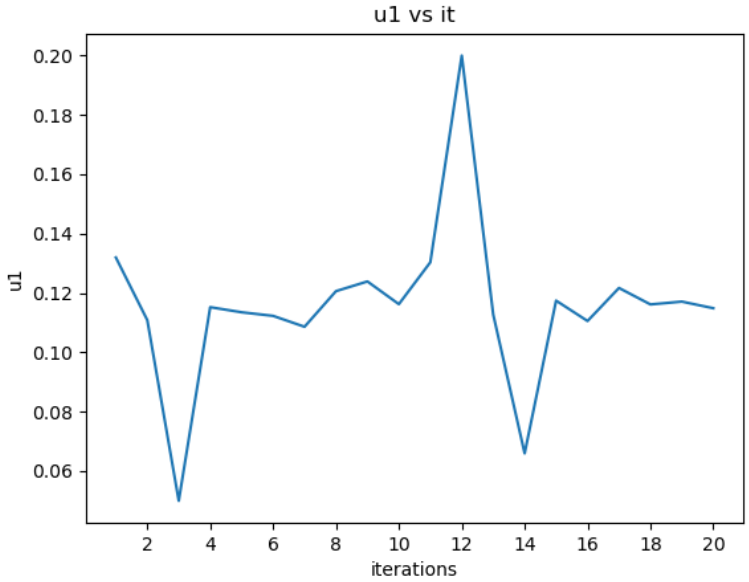

Figure 4.30. *Trend of the selected* u_1.

one value (0.12) and has exploratory peaks in experiments 3-12-14.

Another thing that can be noticed by comparing the *Figure 4.15* with the *Figure 16* is that the optimal point found with the exploratory method is practically at the same optimal point obtained with the Bayesian optimisation. The difference is that after finding the optimal point the RL algorithm concentrated the experiments in that point while the exploratory algorithm continued to choose non-optimal conditions for the experiment.

4.4 Future work

In this chapter, it has been demonstrated how the coupling between the method of individuation of the structure of the model and the identification of the parameters with an algorithm of RL is very effective. It also creates numerous ideas for future applications. Future work will test this method on more complex systems characterised by more design variables, possible structures and parameters.

In this case study, the objective was to find the experimental conditions that would allow the identification of the correct model, and therefore the objective function was used as an indicator to assess the correctness of the model. The use of RL also gives different options, in fact, in addition to the Bayesian Optimisation there are many other algorithms that can be used, some more exploratory in which the degree of learning is low, while others tend to exploit more optimal conditions, exploring less the experimental space

Conclusions

This project presents a new method for identifying the mathematical model to describe a physical system. The goal was to be able to simultaneously identify the model structure, the parameters and optimize the conditions under which to run the experiment. The first part of the study involved the development of the model identification algorithm using a superstructure method and an approach for optimization. The second part focuses on design of the experiments by exploiting the coupling between the RL and the model identification algorithm. The results obtained with this method led to the following considerations:

- the method was able to correctly identify the kinetic model even at high noise levels and low numbers of experimental measurements;
- the algorithm is able to choose the correct kinetic model even if very similar models appear within the superstructure;
- there is a much faster convergence of parameters to their real value;
- it takes only a few experiments to obtain an accurate estimate of the parameters;
- in addition to identifying the model it is also possible to identify the optimal experimental conditions.

The fundamental benefit of the proposed method is that it ensures a faster convergence of the identified parameters to the real parameters. In fact, using the RL-Design of Experiments the parameters converge very quickly to their real value. With this method, it takes about 10 experiments to obtain good estimates. Using a pure exploratory method, the convergence is much slower and often 20 experiments are not enough to obtain a good estimate of the parameters. This means that by using this method, fewer experiments can be performed to identify the kinetic model and this means saving time, money and resources.
These results are very promising and pave the way for new methodologies that combine traditional mathematical pattern recognition techniques with RL methodologies.
Future work will investigate the performance of this model with more complex systems, characterised by more design variables, possible structures and parameters. Furthermore, the RL technique used in this project is just one of the possible RL techniques that can be used to optimise experimental conditions. Therefore, in future work, it will be necessary to investigate which of the different RL techniques is able to identify the optimal conditions most quickly and accurately.

Nomenclature

x_1 = yeast concentration (gL^{-1})
x_2 = substrate concentration (gL^{-1})
u_1 = dilution factor (h^{-1})
u_2 = substrate concentration in the feed (gL^{-1})
y = measurement
$\hat{y}$ = prediction of the measurement
$\mathbf{V}(\hat{\theta}, \phi)$ = covariance matrix
v_{ij} = element ij of the covariance matrix
$\mathbf{M}^0$ = initial information matrix
φ = design vector
$\mathbf{M}(\hat{\theta}, \phi)$ = information matrix
$\mathbf{Q}$ = sensitivity matrix
w_i = weight of the neurons in the neural network
b = vector of biases
s = state
o = observation
$R(\tau)$ = reward
V^{π} = value function
$\mathbb{L}(Y|\theta)$ = logarithmic for of the likelihood function
u_b = upper bound
l_b = lower bound
N_θ = number of parameters

Greek letters

θ_i = generic parameter
$\hat{\theta}_i$ = prediction of the parameter i
θ_1 = first parameter of the case study
θ_2 = second parameter of the case study
θ_3 = third parameter of the case study
θ_4 = fourth parameter of the case study
Θ = library in the SINDy
Ξ = sparse vector
μ = deterministic policy
π = stochastic policy

τ = trajectory
λ_i = integer i of the superstructure

Acronyms

MBDoE = Model Based Design of Experiments
DoE = Design of Experiments
RL = Reinforcement Learning
MMIs = Model Modification Indices
DAEs = Differential Algebraic Equations
MINLP = Mixed Integer Non-Linear Problem
ANNs = Artificial Neural Networks
PINNs = Physic Informed Neural Networks
GP = Genetic Programming
SINDy = Sparse Identification of Non-Linear Dynamics
SSR = sum of squared residual

References

Andreini, Paolo, Thomas Hasenzagl, Lucrezia Reichlin, Charlotte Senftleben-König, and Till Strohsal (2021). Nowcasting German GDP: Foreign factors, financial markets, and model averaging. *International Journal of Forecasting*. DOI: https://doi.org/10.1016/j.ijforecast.2021.11.009. URL: https://www.sciencedirect.com/science/article/pii/S0169207021001837.

Asprey, S.P. and S. Macchietto (2000). Statistical tools for optimal dynamic model building. *Computers Chemical Engineering* **24**, 1261–1267. DOI: https://doi.org/10.1016/S0098-1354(00)00328-8. URL: https://www.sciencedirect.com/science/article/pii/S0098135400003288.

Baldi, Pierre and Kurt Hornik (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks* **2**.1, 53–58. DOI: https://doi.org/10.1016/0893-6080(89)90014-2. URL: https://www.sciencedirect.com/science/article/pii/0893608089900142.

Bannister, Christian, Julian Halcox, Craig Currie, Alun Preece, and Irena Spasic (2018). A genetic programming approach to development of clinical prediction models: A case study in symptomatic cardiovascular disease. *PLOS ONE* **13**, e0202685. DOI: 10.1371/journal.pone.0202685.

Bard, Yonathan (1974). *Nonlinear parameter estimation*. New York: Academic Press.

Brunton, Steven L., Joshua L. Proctor, and J. Nathan Kutz (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences* **113**, 3932–3937. DOI: 10.1073/pnas.1517384113. eprint: https://www.pnas.org/content/113/15/3932.full.pdf. URL: https://www.pnas.org/content/113/15/3932.

Farag, W.A., V.H. Quintana, and G. Lambert-Torres (1998). A genetic-based neuro-fuzzy approach for modeling and control of dynamical systems. *IEEE Transactions on Neural Networks* **9**, 756–767. DOI: 10.1109/72.712150.

Franceschini, Gaia and Sandro Macchietto (2008). Model-based design of experiments for parameter precision: State of the art. *Chemical Engineering Science* **63**, 4846–4872. DOI: https://doi.org/10.1016/j.ces.2007.11.034. URL: https://www.sciencedirect.com/science/article/pii/S0009250907008871.

Gray, Gary J., David J. Murray-Smith, Yun Li, Ken C. Sharman, and Thomas Weinbrenner (1998). Nonlinear model structure identification using genetic programming. *Control Engineering Practice* **6**, 1341–1352. DOI: https://doi.org/10.1016/S0967-0661(98)00087-2. URL: https://www.sciencedirect.com/science/article/pii/S0967066198000872.

Heo, Seongmin and Jay H. Lee (2018). Fault detection and classification using artificial neural networks. *IFAC-PapersOnLine* **51**, 470–475. DOI: https://doi.org/10.1016/j.ifacol.2018.09.380. URL: https://www.sciencedirect.com/science/article/pii/S2405896318320664.

Hoffmann, Moritz, Christoph Fröhner, and Frank Noé (2018). Reactive SINDy: Discovering governing reactions from concentration data. *bioRxiv*.

Johnson, Jeffrey, Jinghong Li, and Zengshi Chen (2000). Reinforcement Learning: An Introduction: R.S. Sutton, A.G. Barto, MIT Press, Cambridge, MA 1998, 322 pp. ISBN 0-262-19398-1. *Neurocomputing* **35**, 205–206.

Le Guiban, Kaourintin, Arpad Rimmel, Marc-Antoine Weisser, and Joanna Tomasik (2018). Completion of partial Latin Hypercube Designs: NP-completeness and inapproximability. *Theoretical Computer Science* **715**, 1–20. DOI: https://doi.org/10.1016/j.tcs.2018.01.014. URL: https://www.sciencedirect.com/science/article/pii/S0304397518300409.

Michel, Philippe (2021). Model of neo-Malthusian population anticipating future changes in resources. *Theoretical Population Biology* **140**, 16–31. DOI: https://doi.org/10.1016/j.tpb.2021.03.002. URL: https://www.sciencedirect.com/science/article/pii/S0040580921000198.

Pan, Elton, Panagiotis Petsagkourakis, Max Mowbray, Dongda Zhang, and Antonio del Rio-Chanona (2021a). Constrained Q-Learning for Batch Process Optimization. *IFAC-PapersOnLine* **54**, 492–497. DOI: https://doi.org/10.1016/j.ifacol.2021.08.290. URL: https://www.sciencedirect.com/science/article/pii/S2405896321010636.

Pan, Elton, Panagiotis Petsagkourakis, Max Mowbray, Dongda Zhang, and Ehecatl Antonio del Rio-Chanona (2021b). Constrained model-free reinforcement learning for process optimization. *Computers Chemical Engineering* **154**, 107462. DOI: https://doi.org/10.1016/j.compchemeng.2021.107462. URL: https://www.sciencedirect.com/science/article/pii/S0098135421002404.

Pearson, Karl, R. A. Fisher, and Henry F. Inman (1994). Karl Pearson and R. A. Fisher on Statistical Tests: A 1935 Exchange from Nature. *The American Statistician* **48**, 2–11. URL: http://www.jstor.org/stable/2685077.

Petsagkourakis, P., I. Orson Sandoval, E. Bradford, D. Zhang, and E.A. del Rio-Chanona (2019). Reinforcement Learning for Batch-to-Batch Bioprocess Optimisation. In: *29th European Symposium on Computer Aided Process Engineering*. Ed. by Anton A. Kiss, Edwin Zondervan, Richard Lakerveld, and Leyla Özkan. Vol. 46. Computer Aided Chemical Engineering. Elsevier, pp. 919–924. DOI: https://doi.org/10.1016/B978-0-12-818634-3.50154-5. URL: https://www.sciencedirect.com/science/article/pii/B9780128186343501545.

Quaglio, Marco, Eric S. Fraga, and Federico Galvanin (2020a). A diagnostic procedure for improving the structure of approximated kinetic models. *Computers Chemical Engineering* **133**, 106659. DOI: https://doi.org/10.1016/j.compchemeng.2019.106659. URL: https://www.sciencedirect.com/science/article/pii/S0098135419304818.

Quaglio, Marco, Louise Roberts, Mohd Safarizal Bin Jaapar, Eric S. Fraga, Vivek Dua, and Federico Galvanin (2020b). An artificial neural network approach to recognise kinetic models from experimental data. *Computers Chemical Engineering* **135**, 106759. DOI: https://doi.org/10.1016/j.compchemeng.2020.106759. URL: https://www.sciencedirect.com/science/article/pii/S0098135419302297.

Raissi, M., P. Perdikaris, and G.E. Karniadakis (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics* **378**, 686–707. DOI: https://doi.org/10.1016/j.jcp.2018.10.045. URL: https://www.sciencedirect.com/science/article/pii/S0021999118307125.

Raissi, Maziar, Paris Perdikaris, and George Em Karniadakis (2018). *Multistep Neural Networks for Data-driven Discovery of Nonlinear Dynamical Systems*. arXiv: 1801.01236 [math.DS].

Rosenberg, Noah A. (2021). Population models, mathematical epidemiology, and the COVID-19 pandemic. *Theoretical Population Biology* **137**, 1. DOI: https://doi.org/10.1016/j.tpb.2021.01.001. URL: https://www.sciencedirect.com/science/article/pii/S0040580921000022.

Taylor, Connor J. et al. (2021). Rapid, automated determination of reaction models and kinetic parameters. *Chemical Engineering Journal* **413**, 127017. DOI: https://doi.org/10.1016/j.cej.2020.127017. URL: https://www.sciencedirect.com/science/article/pii/S1385894720331454.

Van Der Malsburg, C. (1986). Frank Rosenblatt: Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. In: *Brain Theory*. Ed. by Günther Palm and Ad Aertsen. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 245–248.

Zhao, Jia and Jarrod Mau (2020). *Discovery of Governing Equations with Recursive Deep Neural Networks*. arXiv: 2009.11500 [math.NA].

Acknowledgments

First of all, I want to express gratitude to my supervisor Professor Fabrizio Bezzo for giving me the opportunity to attend UCL and for supervising me during the writing of my thesis. I am extremely grateful to my co-supervisor Professor Federico Galvanin who has advised, supported and guided me throughout my time at UCL. I would also like to thank Dr. Panagiotis Petsagkourakis for his availability and for helping me to carry out this thesis project.

I must also thank my parents Sara and Luciano and my sisters Emanuela and Silvia who, despite the distance, have always supported me emotionally on this journey.

I must also thank my lifelong friends for making this journey unforgettable.

And finally, a huge thank you to Giulia for always being by my side and because without you, all this would not have been possible.