POLITECNICO
MILANO 1863

School of Industrial and Information Engineering

Master of Science in Chemical Engineering

Department of Chemistry, Materials and Chemical Engineering “Giulio Natta”

Online Adaptive Model-Based Optimal Experimental Design: an Example of Application to a Continuous Chemical Process

Supervisor: Prof. Flavio MANENTI
Co-Supervisor: Prof. Dr.-Ing. habil. Jens-Uwe REPKE
Advisor: M.Sc. Volodymyr KOZACHYNSKYI

M.Sc. Thesis of:
Sergio Porcelli 893463

Academic Year 2018 – 2019

Preface

This thesis has been made possible through the collaboration between the Sustainable Process Engineering Research (SuPER) team of Politecnico di Milano, led by Prof. Flavio Manenti, and d|b|t|a Dynamik und Betrieb technischer Anlagen of Technische Universität Berlin, led by Prof. Dr.-Ing. habil. Jens-Uwe Repke, thanks to Erasmus+ sponsorship.

It is also part of a bigger project of the Collaborative Research Centre/Transregio 63 “Integrated Chemical Processes in Liquid Multiphase Systems” (InPROMPT), to investigate the model of a mini-plant for Hydroformylation of Alkenes.

Acknowledgements

I would like to give thanks to both Prof. Dr.-Ing. habil. Jens-Uwe Repke and Prof. Flavio Manenti, who put faith in me in undertaking this international project and took me under their wings.
I would also like to extend my gratitude to Dr.-Ing. Erik Esche, M.Sc. Markus Illner and M.Sc. Volodymyr Kozachynskyi, who followed my experience abroad and work almost daily, complete with ups and downs. Especially you Vova, you've been a mentor and a friend to me, and I appreciate you.

A special thanks goes to the Fachgebiet Dynamik und Betrieb technischer Anlagen group of the Technische Universität Berlin as a whole, which accepted me as part of the family during my stay. Of course, thanks to SuPER group from Politecnico di Milano as well, who have made the writing and discussion of this thesis possible. And finally, thanks to Erasmus+ program. They strongly believe in cultural and ideas exchange to promote peace and understanding across countries, and that can never go unnoticed.

Thank you to my dad Francesco, who is a well of knowledge not to be underestimated.
Thank you to my mom Alessandra, you're a panther who has raised a tiger.
Thank you to my sister Vittoria, I will always remember our first debates about chemistry.

They say friends are the family you choose. I can say I am very lucky to have so many friends around me who have been there since day one. This is also for you: Fausto, who has the power to calm me down by just being there; Gabriella, who taught to laugh more; Serena, who taught me there is only one thing without remedy; Gabriele, who taught me you can be relaxed and determined at the same time; Marco, who is the sharpest person I know; Stella, who is mastering her passion; Stefania, who is always raising my bar on what can be achieved; Gianmarco, with whom I have uncorked way more champagne bottles than I care to remember; Antonio, you are my example of resilience; Ludovica, I love how honest we can be with each other; finally Vittorio, who has been my rock for the whole of my experience in PoliMi.

Abstract

Optimal Experimental Design (OED) is a proven way to increase efficiency of experimental campaigns for models identification. An optimal experimental campaign is capable of maximizing the amount of information obtained per experiment, thus reducing the total amount of experiments performed, their cost, their time, the material, energetic flows and human resource needed. OED is here implemented in a System Identification framework, to a model of a Continuously Stirred Tank Reactor (CSTR), jacketed, with reactions. Three classical criteria (A-, D-, E-) have been investigated to determine which of them is the fastest, most robust and accurate.

Keywords: *Optimal Experimental Design; System Identification*

Estratto

La progettazione ottimale degli esperimenti (OED) è una tecnica provata per migliorare l'efficienza delle campagne sperimentali con l'obiettivo di identificazione di un modello. Una campagna sperimentale ottimale è in grado di massimizzare la quantità di informazioni ottenuta dal singolo esperimento, quindi riducendo il numero totale di esperimenti per campagna, il loro costo, la loro durata e i flussi materiali, di energia e di risorse umane necessari. OED è, in questo lavoro, implementata in un framework per identificare un modello di un reattore CSTR (Continuously Stirred Tank Reactor), incamiciato, con reazione, con l'obiettivo di valutare tre criteri classici (A-, D-, E-), per determinare quale sia il più veloce, robusto e accurato.

Parole chiave: *Optimal Experimental Design; Identificazione di Sistemi*

Contents

List of Figures

List of Tables

Nomenclature & Abbreviations

PE: Parameter Estimation
OED: Optimal Experimental Design
$\boldsymbol{y}$: State Variables Vector
$\boldsymbol{y}_{model}$: State Variables predicted by the model Vector
$\boldsymbol{y}_{meas}$: State Variables measured from the real plant Vector
$\boldsymbol{\vartheta}$: Parameters Vector
$\boldsymbol{\vartheta}^{*}$: True Parameters Vector
$\boldsymbol{\vartheta}^{best}$: Parameters Vector yielding the lowest LSQ value
$\boldsymbol{\vartheta}^{active}$: Active Parameters Vector
Φ: Cost function
$NLSQ$: Nonlinear Least Square Error Function
LSQ: Least Square Error Function
$\boldsymbol{C}_y$: Measurement Covariance Matrix
$\boldsymbol{\sigma}_y^2$: Measurement Device Error
$\boldsymbol{FIM}$: Fisher Information Matrix
PSD: Positive Semidefinite
$\mathbb{R}$: Real Numbers
$\boldsymbol{S}$: Parameters Sensitivity Matrix
$\boldsymbol{C}_\sigma$: Parameter Variance-Covariance Matrix
σ_{ij}^2: i^{th},j^{th} Parameters covariance
Ψ: OED criterion
$\boldsymbol{u}$: Controls Vector
$\boldsymbol{u}^{best}$: Controls Vector yielding the lowest Criterion value
$\boldsymbol{\lambda}$: Eigenvalue Vector
$\boldsymbol{\lambda}_{min}$: Minimum Eigenvalue
A-: Average Variance
D-: Determinant
E-: Eigenvalue
$\boldsymbol{\mu}$: Parameters Sensitivity Matrix $\boldsymbol{S}$ Eigenvalue Vector
ε: Epsi-threshold
k: Condition Number
γ: Colinearity Index
k_{max}: Biggest acceptable Condition Number
γ_{max}: Biggest acceptable Colinearity Index
SsS: Subset Selection
IG: Initial Guess
IC: Initial Condition
^{new}: for the following experiment
$\boldsymbol{t}$: Timespan Vector
$_0$: Values at beginning of current experiment/Initial Conditions
$fast$: Related to the fast run
$long$: Related to the long, more precise run
$\boldsymbol{E_{att}}$: Energy of Activation Vector
$\boldsymbol{k_0}$: Frequency Factor Vector
UA: Global Heat Exchange Coefficient, coupled to Reference Area
c_{pmix}: Specific Heat of the Fluid Mixture
F: Volumetric Flow
$\boldsymbol{C}$: Concentrations Vector
in: Related to the Inlet
T: Temperature
T_j: Jacket Temperature
R: Universal Gas Constant
$\boldsymbol{\Delta h_r}$: Heat of Reaction Vector

1 Introduction

1.1 Motivation

The prediction of Chemical Plants behavior can be improved with the use of first-principle, black-box, or mixed models. With their implementation, production capacity, efficiency, safety can be increased, and costs can be reduced, improving market competitiveness. In order to have accurate and robust models, recovering unknown parameters values (i.e.: identify the model) from experimental data is of paramount importance.

Not all experiments are created equal, however, when comparing the amount of information one is capable of obtaining from each. Of course, when the experiments yield more information, the number of experiments needed to identify the model (i.e.: estimate its parameters) is smaller. This is also reflected in shorter experimental campaigns. This reduces their costs and improves accuracy of the estimated parameters, and in turn the predictive capacity of the model itself.

New experiments can thus be designed to maximize the amount of information they can produce, and this is the aim of the statistical discipline of Optimal Experimental Design. When applying this concept to chemical plants, this means that new operating conditions can be implemented, in order to investigate varying behaviors, and thus improve Parameter Estimation routines. In this sense, Optimal Experimental Design is preparatory for Parameters Estimation. Furthermore, the procedure can be performed iteratively during an experimental campaign, in order to optimally prepare new experiments. This is particularly interesting from a cost standpoint, because the plant can be kept running during the whole experimental campaign, reducing downtime and used resources, that otherwise would be needed during start-up and shut-down.

1.2 State of the Art

Through experiments, a hypothesis is either validated or disqualified. The traditional, archetypal workflow is as follows:

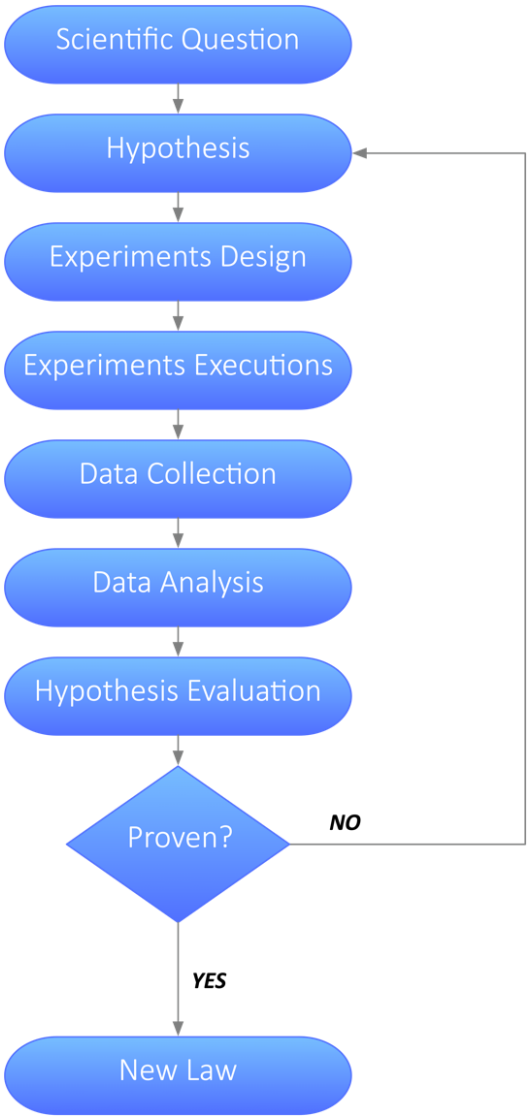

Figure 1: Archetypal experimental workflow

The first step is to ask a scientific question of interest regarding an observed phenomenon. Consequently, one must pose a hypothesis to explain the phenomenon, which could be disproven. Then, it is needed to design an experiment and execute it. The fifth step is to collect the data that the experiment yields. It is important that the experimental results can be reproduced by anyone, in order to provide proof of a specific behavior regarding the observed phenomenon. The collection of data is later followed by their analysis, in order to perform a judgement of value on the previously posed hypothesis. If it's proven by the experiment, and the experiment is reproducible indefinitely, then a new theory or law has been discovered. However, if the experiment disproves the hypothesis, a new one must be posed, so the researcher must go back to the second step.

In the particular case of Chemical Plants modelling, the hypothesis can be regarded as the model itself, assumed to properly describe the plant behavior. It could be comprised of empirical equations (black-box model), balance equations (mechanistic, first-principle model) or a combination of the two types (mixed model), and the parameters of the model itself.

Some considerations on the different types of model will now follow. The peculiarity of black-box models is that no knowledge of the physical phenomena is needed to build them: experimental conditions are considered as input variables in a system of equations, chosen a priori, with characteristics that give the black-box model its name (e.g.: FIR, ARX, ARMAX, etc.) and the responses of the plant are its outputs. This approach is simple to implement, and usually robust when linear systems, well-conditioned problems are involved (Ljung, 1998). However, when dealing with non-linear systems, such as the ones so common in Chemical Engineering, a set of initial guesses may result in more than one solution because of multiple local minima, reducing the robustness of the findings. More complex, nonlinear black-box models (e.g.: NFIR, NARX, NARMAX) can be used, but their prediction capabilities are usually worse than first-principle model (Ljung, 1998). If chemical processes phenomena are known, then first principle models can be used, and they would of course present the same degree of nonlinearity as the investigated plant. On the other hand, usually not all phenomena will be considered, since they would increase the computational complexity and time without an actual increase in accuracy, thus an optimal trade-off between complexity and accuracy in results must be found.

As mentioned beforehand, certain experiments have a better quality than others. That is, certain experiments are able to provide a larger amount of information than others at the same cost, or they are able to provide the same amount of information with less cost. Cost is here defined as a general term, as it's commonly done in engineering endeavors for the sake of simplicity, with a variety of meaning (e.g.: literal cost of the individual experiment, of the experimental campaign as a whole, but also time needed to obtain the desired results, amount of raw materials, disposal of waste, product quality, etc.) depending on the real life application. Once cost has been defined, its representative function (i.e.: the so-called cost function) can be implemented in an optimization algorithm (Bitterlich and Knabner, 2003; Huang, Hjalmarsson and Gerencser, 2012; Valenzuela, Rojas and Hjalmarsson, 2015; Barz *et al.*, 2016).

The aim of Optimal Experimental Design is precisely to define a cost function, the minimization of which in turn maximizes the amount of information provided by experiments. This field of statistics stems for the discovery by R. A. Fisher (Fisher, 1935; Fisher Box, 1980) that the potential amount for information depends on the design of the experimental campaign. Fisher later defined the Fisher Information Matrix $\boldsymbol{FIM}$, which represents the amount of information obtainable by an experiment. The problem of designing optimal experiments has then been shifted to maximizing the $\boldsymbol{FIM}$, new controls are found for the following experiment. Criteria, i.e.: cost functions representing the $\boldsymbol{FIM}$, for OED have been developed in literature (e.g.: A-, D- and E-) and their performance may be dependent on the case at hand (Pukelsheim, 2006). However, only decades later application of design optimality moved to include statistical experiments, especially with linear black-box models (Bardow, 2008; Haber, Horesh and Tenorio, 2008, 2010).

Model-Assisted OED is a hybrid of first principle mechanistic equations with black-box empirical functions. This is a more effective way of designing experiments than the black-box model counterpart, because it solves some of the problems involved with non-linearity (Chen and Wang, 2004). Obviously, some process knowledge is needed, otherwise it's inapplicable.

The last step in the evolution of OED is the so-called Model-Based OED: here full process dynamics knowledge is assumed and implemented in the Experimental Design routine. The current thesis work will investigate a nonlinear first principle model (Müller *et al.*, 2014), as a proxy for Chemical Engineering applications. However, simulations and the model identification depend greatly on model quality. Uncertainties on measured data are to be especially accounted for here, as they may cause significant error in parameters estimation and simulation results. Thus, it is a recommended way to tackle the problem only when the model is validated for a number of scenarios (Franceschini and Macchietto, 2008; Barz *et al.*, 2013, 2016; Annergren *et al.*, 2017; Abt *et al.*, 2018), as it is the case for the Arrhenius law for chemical reactions kinetics.

In the classical sense, OED paired to PE is usually performed only for one experiment. Then the experimental campaign is reset, and OED can be performed again. However, this procedure presents the major drawback of considering each experiment as a standalone, and thus it is not able to adapt to the results in itinere, during

a cumulative experimental campaign. The solution lies in looping the PE with OED routine on the multitude of consequential experiments part of the whole experimental campaign, with an adaptive approach (Huang, Hjalmarsson and Gerencser, 2012; Fohring, 2016; Gu, 2016; Strömberg, 2016). Thus, after having performed each new experiment, the overall results set, previous experiments included, is globally considered. In this way, routine accuracy and robustness are increased, having an increasingly larger data set to draw results from, but computational times increase as well. This means that, with an additional experiment, the increase of accuracy and robustness may not be a good trade with respect to the time needed to run the additional loop. Consequently, for the loop adaptability to be effective, an optimum must be reached between the quality of the results and the time needed to obtain them. Defining the maximum number of previous experiments data set to be used in the Parameters Estimation may be beneficial to decrease the single loop computational time.

Expanding on the previously mentioned point, the time it takes to run a single experiment usually corresponds to the time needed to reach steady-state conditions, when considering continuous chemical plants. If the routine converges to accurate results, and computational time is short enough when compared to experimental time, online implementation can be considered (Barz *et al.*, 2013, 2016). The most important advantage of running the Adaptive PE & OED online is that, especially for continuous plants, shut down of the plant in between experiments becomes unnecessary. This drastically reduces the dead times required for shutdown and startup, and thus decreases the material and energy flows spent for an experimental campaign, that is its cost altogether.

Additionally, from a mathematical standpoint, the outputs of a PE routine (i.e.: estimated parameters, $\boldsymbol{FIM}$ of the model) are the inputs of the OED routine. The outputs of the latter (i.e.: new controls) are then inputs for the new experiment on which a new PE routine is run. This kind of iterative procedure, applied during an individual experimental campaign, is adaptive. Furthermore, if this routine is so fast that it can produce new controls for the following experiment before the current one has been completed, the routine could be run online on a real plant. This means that a Model Based, Online, Adaptive Parameter Estimation & Optimal Experimental Design routine can be built, and it has all the advantages previously mentioned. It warrants attention that CPU time increases with model complexity, mainly due to number of equations, and non-linearity. This imparts a direct bottleneck on implementation time for new experiments, thus limiting the capacity of running the PE & OED algorithm online.

Another important aspect that will be discussed lies in the difficulty of approximating non-linear problems as linear. For this reason, ill-posed problems, that is problems with ill-conditioned $\boldsymbol{FIM}$, may significantly reduce the robustness of the overall framework presented thus far. Since some of the parameters may be linearly dependent to varying degrees, their identifiability analysis needs to be performed, followed by the regularization of the $\boldsymbol{FIM}$ (López C. *et al.*, 2015; Carolina López Cárdenas *et al.*, 2016).

1.3 Thesis Objectives

The aim of this thesis is to implement a novel algorithm for Model-based Adaptive Online Optimal Experimental Design, inserted in a larger framework of model parameters estimation for a nonlinear chemical process model. A textbook example of a heated CSTR with reaction (Müller *et al.*, 2014), in order to investigate the performance of three nonlinear classical OED criteria (i.e.: Average Variance (A-), Determinant (D-), Eigenvalue (E-)).

The example model is rather simple and the "true" parameters values are already known for validation and assessment purposes. Of course, the framework may then be implemented to more complex applications, while retaining the continuous nature of the process.

The model is non-linear, as are most Chemical Engineering applications, and will be treated with proper numerical tools. In this regard, ill-posedness of the problem may arise and will be solved with a regularization routine, adapted from previous parameters estimation framework (Carolina López Cárdenas *et al.*, 2016).

The testing of the algorithm will revolve around the critique of the best choice of the optimality criterion for the OED problem among A-, D- and E. The analysis of the different approaches will mainly focus on robustness, computational time, and accuracy of the results.

The code has been adapted in MatLab® 2018b, for further implementation in Python®.

2 Theoretical Background

This chapter is intended as a review of the basic definition and theoretical knowledge needed to understand the mathematical formulation that will be presented in the thesis. In the first section, Model-Based Adaptive Parameter Estimation & Optimal Experimental Design will be explained considering the problem well-posed. This means that the parameters are assumed as identifiable and their value can be calculated effectively. In the second section, regularization techniques on how to manage the issue of ill-posed problems will be explained. In this way, complexity of the subject at hand can be developed sequentially, making it easier to understand.

2.1 Parameter Estimation

When pursuing the task of writing a model to predict physical phenomena, researchers operate under the assumption that the values of the model parameters affect the predicted state variables. The branch of statistics that handles that data (i.e.: empirical data with a random component) to estimate the parameters values, is called Estimation theory. It is aimed at using the finite set of sample data $\boldsymbol{y}_{meas}$ to calculate reasonable parameters values $\boldsymbol{\vartheta}$ close to their true values $\boldsymbol{\vartheta}^*$.

2.1.1 Mathematical formulation

The estimation of parameters is a minimization problem of an objective function Φ function of several parameters. These parameters are degrees of freedom of the problem and are defined in the vector $\boldsymbol{\vartheta}$. The best Parameters Estimation is given by the minimization procedure stated as follows:

$$\boldsymbol{\vartheta}^{best} = \min_{\boldsymbol{\vartheta}} \Phi(\boldsymbol{\vartheta})$$

where $\boldsymbol{\vartheta}^{best}$ is the parameters set that minimizes an error function Φ. An example of a widely used error function is the weighted nonlinear least-squares criterion Φ^{NLSQ}. It has been chosen in this thesis work, and is stated as follows:

$$\Phi^{NLSQ}(\boldsymbol{\vartheta}) = \frac{1}{2}(\boldsymbol{y}(\boldsymbol{\vartheta}) - \boldsymbol{y}_{meas})^T \boldsymbol{C}_y^{-1}(\boldsymbol{y}(\boldsymbol{\vartheta}) - \boldsymbol{y}_{meas})$$

where $\boldsymbol{C}_y^{-1} \epsilon \mathbb{R}^{N_y \times N_y}$ is the inverse of the variance matrix $\boldsymbol{C}_y$ of the measured data (i.e.: measuring device error $C_{y_{ii}} = \sigma_{y_i}^2$ for $i = 1, \dots, N_y$), that is the weight for the objective function. The latter will be referred to as LSQ from now on, for the sake of simplicity.

2.2 Optimal Experimental Design

The class of experiments that are considered optimal with respect to a statistical criterion are called optimal design. The creation of the branch of statistics called "Optimal Experimental Design" has been credited to Danish statistician Kirstine Smith (Smith, 1918) and further developed by R. A. Fisher. He proved the link between the design setup and the potential of information that can be extracted from an estimation (Fisher, 1935; Fisher Box, 1980).

This link has been explained via the definition of the Fisher Information Matrix $\boldsymbol{FIM}$, which is dependent on plant controls set $\boldsymbol{u}$ and model parameters $\boldsymbol{\vartheta}$, mathematically formulated as follows when $\boldsymbol{y}_{meas}$ distribution is assumed to be Gaussian:

$$\boldsymbol{FIM}(\boldsymbol{u}, \boldsymbol{\vartheta}) = \boldsymbol{S}^T \boldsymbol{C}_y^{-1} \boldsymbol{S} \qquad \text{where } \boldsymbol{S} = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{\vartheta}}$$

where the inverse of the $\boldsymbol{FIM} \in \mathbb{R}^{N_\vartheta \times N_\vartheta}$ is equal to the matrix product between the parameters Jacobian or sensitivity matrix $\boldsymbol{S} \in \mathbb{R}^{N_\mathcal{Y} \times N_\vartheta}$, i.e.: first derivative of the state variables with respect to the parameters. It is important to point out that, because of the way the $\boldsymbol{FIM}$ is defined, its computation becomes a rather simple problem of computing $\boldsymbol{S}$ eigenvalues.

A couple of years later, Cramer and Rao (David and Cramer, 1947; Rao, 1992) independently derived that the inverse of the $\boldsymbol{FIM}$ acts a lower bound on the covariance of deterministic parameters for unbiased estimators, stated as follows:

$$\boldsymbol{FIM}(\boldsymbol{u}, \boldsymbol{\vartheta})^{-1} \leq \boldsymbol{C}_\sigma(\boldsymbol{\vartheta})$$

where $\boldsymbol{C}_\sigma \in \mathbb{R}^{N_\vartheta \times N_\vartheta}$ is the parameters variance-covariance matrix, i.e.: $\boldsymbol{C}_{\sigma_{ij}} = \sigma_{ij}^2$ for $i, j = 1, \dots, N_\vartheta$. This rather simple formulation explains how the uncertainty of the parameters, i.e.: their variance, can be minimized, with a properly designed experiment that maximizes the $\boldsymbol{FIM}$ with respect to the parameters. From another point of view, it can also mean that, maximizing the $\boldsymbol{FIM}$, one can maximize the confidence with which the estimated parameters value is regarded as plausible, or minimize the parameters confidence region.
This theorem became the foundation for the optimization of experimental setup, with the aim of obtaining reliable estimates of parameters, that is high accuracy, that is small confidence regions.

2.2.1 Mathematical formulation

The optimization of experimental design is a minimization problem of a cost function Ψ dependent on the $\boldsymbol{FIM}$, with changing controls $\boldsymbol{u}$, stated as follows:

$$\boldsymbol{u}^{best} = \min_{\boldsymbol{u}} \Psi(\boldsymbol{FIM})$$

where $\boldsymbol{u}_{best}$ is the controls set yielding the lowest Ψ. This is true for linear models, however it is only an approximation for non-linear ones, because of the existence of multiple local minima. However $\Psi : PSD(N_\vartheta) \longrightarrow \mathbb{R}$, that is it extrapolates a number from a matrix, thus some amount of information is lost. This implies that a suitable metric of the $\boldsymbol{FIM}$ must be chosen to define the cost function. From a geometrical point of view, this means that only one representative geometrical feature of the confidence region can be used at a time.

2.2.2 OED criteria

As stated before, the aim of any cost function Ψ is to represent a feature of the $\boldsymbol{FIM}$, or the largeness of the confidence area represented by $\boldsymbol{C}$. The cost function Ψ has been classically defined in 3 main ways (Pukelsheim, 2006), for nonlinear problems such as the one at hand, identified by the letters A-, D-, E-, presented in the current section, with their mathematical formulation first, and their geometrical interpretation later. In order to be able to apply the Information functions theory and to compare non-linear problems, thus with dimensions greater than 1, each classical formulation has been scaled with respect to the number of the problem's dimension, i.e.: N_ϑ in this case.

The Average Variance (A-) criterion Ψ^A represents the trace of the $\boldsymbol{FIM}$, and is stated as follows:

$$\Psi^A(\boldsymbol{FIM}) = \frac{1}{N_\vartheta}[tr(\boldsymbol{FIM}^{-1})] = \frac{1}{N_\vartheta}\left[\sum_{i=1}^{N_\vartheta} \frac{1}{\lambda_i(\boldsymbol{FIM})}\right]$$

This criterion minimizes the trace of the inverse $\boldsymbol{FIM}$, that is it maximizes the smallest $\lambda_i(\boldsymbol{FIM})$. It presents an additive nature. Its most notable quality is its simple evaluation, since it consists of computing only the diagonal entries of the $\boldsymbol{FIM}$.

The Determinant (D-) criterion Ψ^D represents the determinant of the $\boldsymbol{FIM}$, and is stated as follows:

$$\Psi^D(\boldsymbol{FIM}) = [det(\boldsymbol{FIM}^{-1})]^{\frac{1}{N_\vartheta}} = \left[\prod_{i=1}^{N_\vartheta} \frac{1}{\lambda_i(\boldsymbol{FIM})}\right]^{\frac{1}{N_\vartheta}}$$

This criterion maximizes the $\boldsymbol{FIM}$ determinant, that is it maximizes the largest $\boldsymbol{FIM}$ eigenvalue $\lambda_i(\boldsymbol{FIM})$. On the other hand, it will minimize the smallest $\lambda_i(\boldsymbol{FIM})$ if its value is less than 1, thus making the problem worse-conditioned. It presents a multiplicative nature. Its most notable quality is its invariance property under re-parametrization.

The Eigenvalue (E-) criterion Ψ^E represents the smallest eigenvalue of the $\boldsymbol{FIM}$, and is stated as follows:

$$\Psi^E(\boldsymbol{FIM}) = \frac{1}{\lambda_{min}(\boldsymbol{FIM})}$$

This criterion maximizes the smallest $\lambda_i(\boldsymbol{FIM})$. It may yield similar results to A-criterion, especially when strong parameters correlation is present. Its most notable quality is its ability to measure problems ill-conditioning, thanks to the direct observation that if $\lambda_{min}(\boldsymbol{FIM}) \approx 0$, $\boldsymbol{FIM}$ is singular.

2.2.3 OED criteria geometrical interpretation

In the current section, the geometrical interpretation of the 3 mentioned criteria will be shown, referring to the following graph:

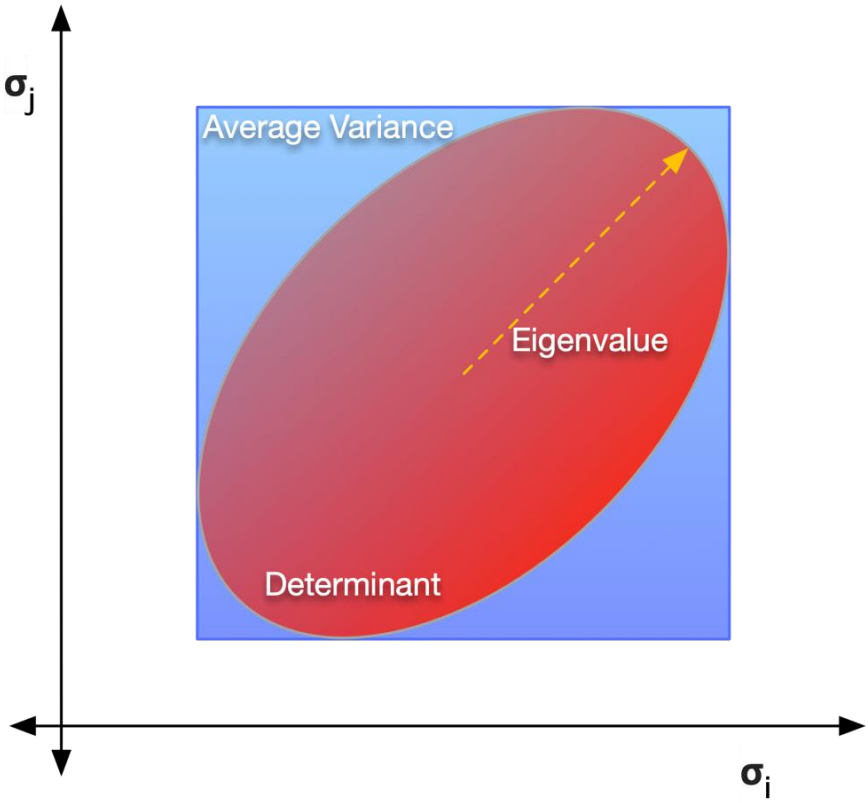

Figure 2: Confidence ellipsoid in 2D space

The shown ellipse is drawn in the (σ_i, σ_j) space, for $i, j = 1, \dots, N_\vartheta$, meaning that it represents the parameters variance if $i = j$, or covariance if $i \neq j$. This 2D representation has been chosen for the sake of simplicity in visualization, but of course it could be built for higher dimension spaces. The shown ellipse represents the area in which the researcher has a certain confidence, usually 90% or more, that the estimated parameters are

accurate. It is thus called confidence region ellipsoid. The smaller it is, the more is the confidence with which the parameters can be assumed as accurately estimated. Each criterion will minimize a certain geometrical feature of the ellipsoid, in the same way that it maximizes a mathematical feature of the $\boldsymbol{FIM}$.

A-criterion minimizes the area of the square inscribing the ellipsoid. D-criterion minimizes the area of the ellipsoid itself. This could be accomplished by decreasing the ellipsoid major axis, corresponding to maximizing the smallest $\lambda_i(\boldsymbol{FIM})$, or minimizing the minor axis to 0, thus collapsing the ellipsoid in an infinitely long confidence region, increasing the parameters correlation. E-criterion minimizes the ellipsoid major axis.

2.3 Problem ill-posedness and Regularization techniques

In the current chapter, notions on problem formulation, well-posedness and regularization techniques will be presented.

2.3.1 Need for regularization

By definition, a model is a simplified representation of real physical and chemical phenomena taking place in a reactor. It is of paramount importance to identify a model that is accurate and robust and in the meanwhile easy to solve. The consequence of this simplification is twofold. On the one hand, it makes the calculations easier, if not even possible. On the other hand, a certain amount of predictive information is lost. Some phenomenon may be erroneously not considered. Furthermore, some of the parameters describing the considered phenomena may be correlated, i.e.: linearly dependent to each other. This implies that an infinite number of values for the correlated parameters may give the same predictive power, i.e.: infinite solutions to the minimization problem, also known as multiple local minima, thus making the individual parameter unidentifiable. This occurrence causes the sensitivity matrix $\boldsymbol{S}$ to become singular, because it is rank deficient or ill-conditioned, making the problem ill-posed and calculations impossible, especially when applying the alphabetic criteria for OED. For example, if even one of the sensitivity matrix eigenvalues is equal to 0, it would be impossible to minimize the D-criterion further. Geometrically, the confidence ellipsoid area stretches to infinite.

It becomes necessary to numerically aid the sensitivity matrix singularity, through one of the many regularization techniques. The choice fell on Subset Selection (SsS), because of its easy implementation. It consists of selecting the subset of the most linearly independent parameters, and thus solving the OED problem with respect to that subset only. Obviously, this means that not all the parameters will be identified, but the identifiable ones will be more accurate.

2.3.2 Regularization steps

The first step is to apply the Singular Value Decomposition to the sensitivity matrix $\boldsymbol{S}$ (Stewart, 1993), that is to compute its parameters eigenvalues $\boldsymbol{\mu}$ and order them in descending order, such that:

$$\mu_1 \geq \mu_2 \geq \cdots \geq \mu_{N_\vartheta} \geq 0$$

The second step is to define the ε-threshold for the sensitivity matrix $\boldsymbol{S}$, as follows:

$$\varepsilon = \max\left\{\varepsilon_k = \frac{\mu_1}{k_{max}}, \varepsilon_\gamma = \frac{1}{\gamma_{max}}\right\}$$

where k_{max} is the biggest acceptable condition number and γ_{max} is the biggest acceptable collinearity index. They have been posed $k_{max} = 1000$ (Grah, 2004) and $\gamma_{max} = 10$ (Brun *et al.*, 2002). The third step is to obtain the rank of $\boldsymbol{S}$ by comparing its eigenvalues $\boldsymbol{\mu}$ and the ε-threshold in the following way:

$$\mu_1 \geq \mu_2 \geq \cdots \geq \mu_{rank} \geq \varepsilon \geq \mu_{rank+1} \geq \mu_{rank+2} \geq \cdots \geq \mu_{N_\vartheta} \geq 0$$

that is the rank of the sensitivity matrix $\boldsymbol{S}$, and thus of the Fisher Information Matrix $\boldsymbol{FIM}$, is equal to the number of the $\boldsymbol{S}$ eigenvalues $\boldsymbol{\mu}$ greater than the ε-threshold. The parameters corresponding to the eigenvalues greater than ε-threshold are regarded as identifiable, and usually they can be estimated accurately enough.

2.3.3 Subset Selection (SsS)

Once the rank has been estimated, the technique of Subset Selection (SsS) is performed by selecting the identifiable parameters subset $\boldsymbol{\vartheta}^{active}$. They are the parameters having an eigenvalue μ_i, for $i = 1, \dots, N_\vartheta$, greater than the ε-threshold. Thus, a new problem is obtained, which is regularized yet reduced. In this way, calculations become possible, however the parameter space is changed, since, by definition, the dimensions of $\boldsymbol{\vartheta}^{active}$ is less than or equal to the dimensions of $\boldsymbol{\vartheta}$. This means that several parameters are excluded from the OED problem solution, thus improved operating conditions will not be searched for them, and if their IG values are far from the true ones, the estimation bias will remain large.

3 Methodology

3.1 Framework

This section describes the computational framework that has been used. It is composed of 3 main parts:

1. Chemical plant data retrieval;
2. Parameter Estimation loop;
3. Optimal Experimental Design loop.

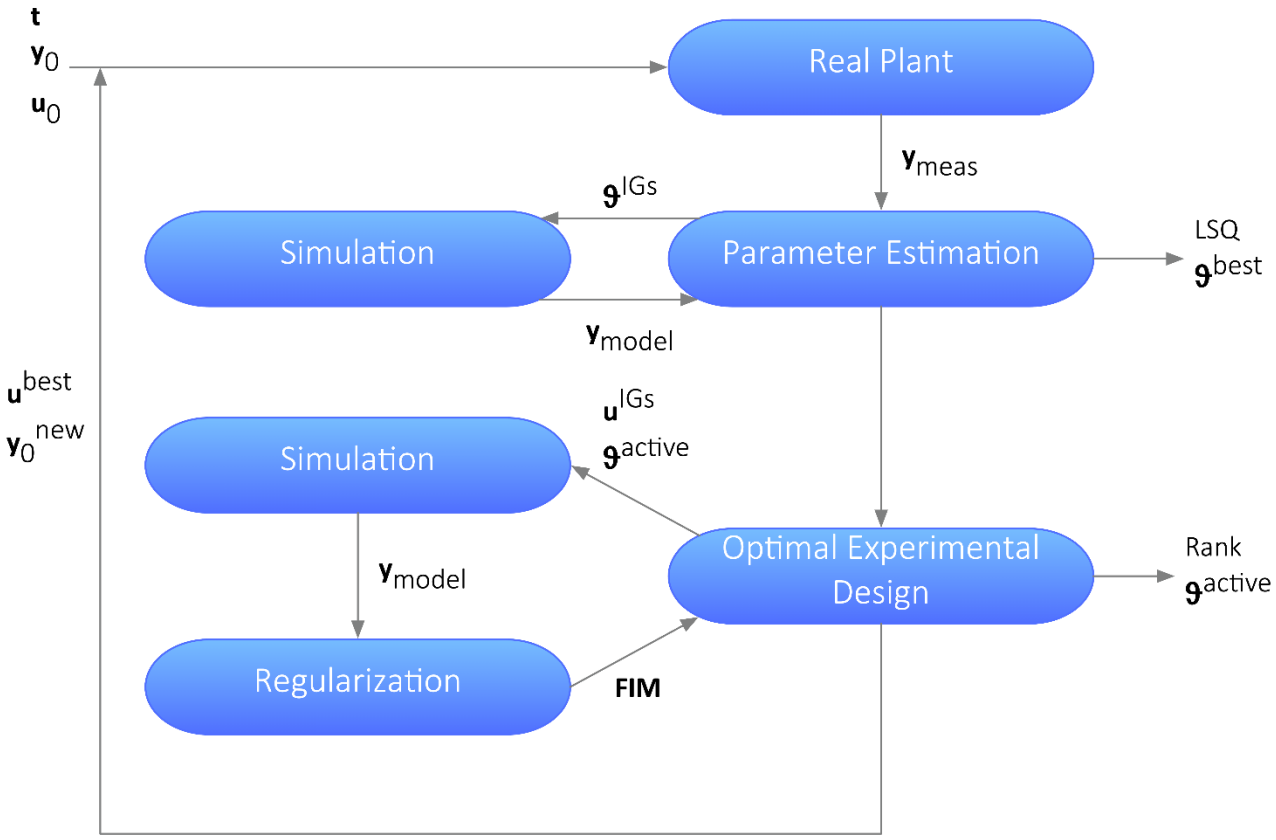

Figure 3: PE & OED workflow

3.1.1 Chemical plant data retrieval

The data used for calculations can be measured from real plant, if available, or adding measurement noise to model predictions. Inputs are experiment timespan $\boldsymbol{t}$, Initial Conditions (ICs) $\boldsymbol{y}(t = t_0) = \boldsymbol{y}_0$, control set $\boldsymbol{u}$. Proper to the model are the true parameters set $\boldsymbol{\vartheta}^*$ and measurement variance $\boldsymbol{\sigma}_y^2$, which are proxies for real plant measurements retrieval. If a real plant is available, these last two inputs are not necessary. Output is the measured state variables $\boldsymbol{y}_{meas}$, and their last value is used in the following experiment to simulate a plant running continuously while the controls change during the experimental campaign.

3.1.2 Parameter Estimation (PE)

It is divided in 2 runs. The first, faster run is used to pick the best set of estimated parameters IG, on which to later perform a more thorough, computationally longer PE which produces more accurate results. Inputs are $\boldsymbol{y}_{meas}$ and PE Initial Guess (IG) sets $\boldsymbol{\vartheta}_{fast}^{IG}$. The outputs are the best estimated parameters $\boldsymbol{\vartheta}_{fast}^{best}$, yielding the lowest LSQ function LSQ_{fast}^{best} from the best parameters initial guess set $\boldsymbol{\vartheta}_{fast}^{bestIG}$. The second, more precise run refines the best parameter set obtained so far to produce the final PE results for the current experiment.

Inputs are the best estimated parameter set $\boldsymbol{\vartheta}_{fast}^{best}$, output of the fast run. Outputs are the more accurate best estimated parameter set $\boldsymbol{\vartheta}^{best}$.

3.1.3 Optimal Experimental Design (OED)

The OED loop is divided in 2 runs, similarly to the PE loop. Before any OED is run, regularization on the estimated parameter set $\boldsymbol{\vartheta}^{best}$ is performed via Subset Selection (SsS). Its outputs are the $\boldsymbol{FIM}$ and the set of active Parameters $\boldsymbol{\vartheta}^{active}$. The first, faster run is used to find the best set of estimated controls, on which to later perform a more thorough, computationally longer OED which produces more accurate controls to actually use in the following experiment. Inputs are the set of active Parameters $\boldsymbol{\vartheta}^{active}$ and OED IG sets $\boldsymbol{u}_{fast}^{IG}$. The outputs are the best controls $\boldsymbol{u}_{fast}^{best}$ yielding the lowest criterion Ψ from the best controls set IG $\boldsymbol{u}_{fast}^{bestIG}$. The second, more precise run refines the best control set for the following experiment. Inputs are $\boldsymbol{u}_{fast}^{best}$, and outputs are the best estimated controls $\boldsymbol{u}^{best}$.

3.2 Real plant vs Model

As previously mentioned, to produce $\boldsymbol{y}_{meas}$, a real plant must be available, or one can simulate real plant measurements by adding noise to model predictions. For practicality purposes in testing the criteria for this thesis work, this second path has been chosen. The model used in the OED program will be described in the current section.

3.2.1 Test case

In order to keep the simulation as simple, yet as representative of a common, real plant, a jacketed Continuously Stirred Tank Reactor (CSTR) with reaction has been chosen (Müller *et al.*, 2014). four generic components (A, B, C, D) undergo three reactions according to the following kinetic scheme:

$$R1)\ A \rightarrow C$$

$$R2)\ B \rightarrow A$$

$$R3)\ A \rightarrow D$$

The desired product is assumed to be C, given from A. B is regarded as a buffer/precursor for A, while D is considered an undesired by-product. In this way, a reaction scheme with reactions in series $B \rightarrow A \rightarrow C$ and a competitive reaction with respect to the desired one $A \rightarrow D$ has been implemented. All reactions have been assumed elementary, to completion, with the simplest Arrhenius Power Law. The corresponding rate of reactions were written in the following way:

$$r_1 = k_1 C_A \text{ where } k_1 = k_{0_1} \exp\left(-\frac{E_{att_1}}{RT}\right)$$

$$r_2 = k_2 C_B \text{ where } k_2 = k_{0_2} \exp\left(-\frac{E_{att_2}}{RT}\right)$$

$$r_3 = k_3 C_A \text{ where } k_3 = k_{0_3} \exp\left(-\frac{E_{att_3}}{RT}\right)$$

A cooling, evaporating (thus at constant temperature T_j) liquid has been assumed to flow in the jacket. The following is a graphical representation of the model:

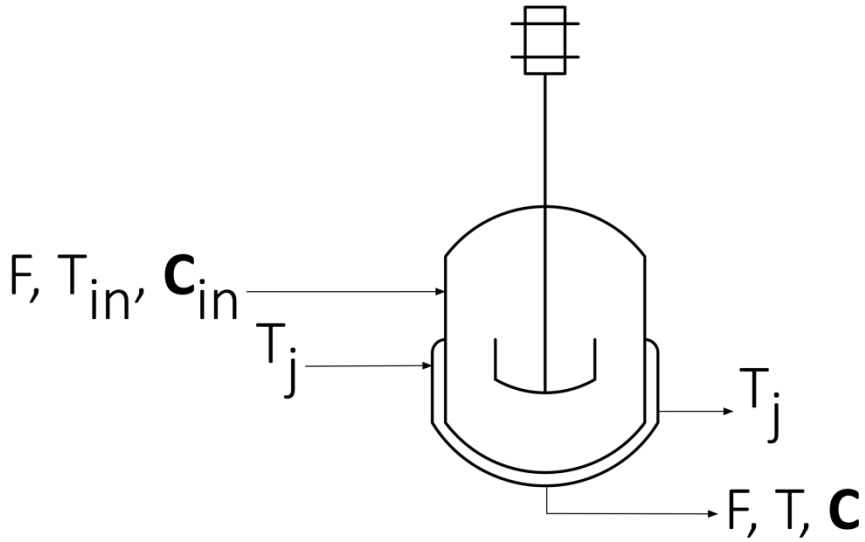

Figure 4: CSTR, jacketed with reactions

With these assumptions, a system of 5 Ordinary Differential Equations (ODEs) of 4 Mass Balances, 1 for each component, and 1 global Energy Balance for the temperature has been generated. It has been assumed that the concentration and temperature at the outlet are the same as the ones inside the reactor, and the flow is constant. Perfect mixing assumption ensures that the reaction is instantaneous in the moment the inlet reactants reach the reaction chamber. Furthermore, density, specific heat of the liquid mixture and heat of the three reactions are assumed constant with time and temperature. Following is the mathematical formulation of the model with the stated assumptions:

$$\frac{dC_A}{dt} = \frac{F}{V}\left(C_{in_A} - C_A\right) - k_{0_1}\exp\left(-\frac{E_{att_1}}{R\,T}\right)C_A + k_{0_2}\exp\left(-\frac{E_{att_2}}{R\,T}\right)C_B - k_{0_3}\exp\left(-\frac{E_{att_3}}{R\,T}\right)C_A$$

$$\frac{dC_B}{dt} = \frac{F}{V}\left(C_{in_B} - C_B\right) - k_{0_2}\exp\left(-\frac{E_{att_2}}{R\,T}\right)C_B$$

$$\frac{dC_C}{dt} = \frac{F}{V}\left(C_{in_C} - C_C\right) + k_{0_1}\exp\left(-\frac{E_{att_1}}{R\,T}\right)C_A$$

$$\frac{dC_D}{dt} = \frac{F}{V}\left(C_{in_D} - C_D\right) + k_{0_3}\exp\left(-\frac{E_{att_3}}{R\,T}\right)C_A$$

$$\frac{dT}{dt} = \frac{F}{V}(T_{in} - T) + \frac{UA}{\rho\,c_{pmix}\,V}\left(T_j - T\right) + \frac{-\Delta h_{r_1}}{\rho\,c_{pmix}}k_{0_1}\exp\left(-\frac{E_{att_1}}{R\,T}\right)C_A + \frac{-\Delta h_{r_2}}{\rho\,c_{pmix}}k_{0_2}\exp\left(-\frac{E_{att_2}}{R\,T}\right)C_B + \frac{-\Delta h_{r_3}}{\rho\,c_{pmix}}k_{0_3}\exp\left(-\frac{E_{att_3}}{R\,T}\right)C_A$$

where $t = 2000s$ is the individual experiment time duration, $\boldsymbol{C} = [C_A; C_B; C_C; C_D] = \left[\frac{mol}{m^3}\right]$ and $\boldsymbol{C}_{in} = \left[C_{in_A}; C_{in_B}; C_{in_C}; C_{in_D}\right] = \left[\frac{mol}{m^3}\right]$ are the outlet and inlet molar concentrations respectively, $[F] = \left[\frac{m^3}{s}\right]$ is the volumetric flow running through the reactor, $V = 1m^3$ is the reactor volume, $\boldsymbol{k_0} = \left[k_{0_1}; k_{0_2}; k_{0_3}\right] = \left[\frac{1}{s}\right]$ are the Arrhenius frequency factor, $\boldsymbol{E_{att}} = \left[E_{att_1}; E_{att_2}; E_{att_3}\right] = \left[\frac{J}{mol}\right]$ are the reactions' Activation Energy, $R = 8.314\frac{J}{mol\,K}$ is the universal gas constant, $[T] = [K]$ is the outlet temperature, $[UA] = \left[\frac{W}{K}\right]$ is the pair of global heat exchange coefficient with its associated area, $\rho = 800\frac{kg}{m^3}$ is the liquid mix density, $\left[c_{pmix}\right] = \left[\frac{J}{mol\,K}\right]$ is the liquid mix specific heat, $\boldsymbol{\Delta h_r} = \left[\Delta h_{r_1}; \Delta h_{r_2}; \Delta h_{r_3}\right] = [0.0045; -0.0055; 0.0045]\frac{J}{mol}$ are the heat of the three reactions.

All calculations have been carried out in MatLab® 2018b. To solve the ODEs system and obtain the sensitivity matrix to compute $\boldsymbol{FIM}$, solver *ode15s* has been selected with *odeset('RelTol',1e-8,'AbsTol',1e-11)*. The simulated real plant data set has been generated by adding noise with the *randn* function applied to measurement error variances $\boldsymbol{\sigma}_y^2 = [4.0E-8; 1E-4; 1E-4; 2.5E-5; 1E-2]$.

The parameters to be identified in this model are: $\boldsymbol{E_{att}}, \boldsymbol{k_0}, UA, c_{pmix}$ for a total of $N_\vartheta = 8$. The true parameters used for the real plant simulation are

$$\boldsymbol{\vartheta}^* = \left[96.0E3\frac{J}{mol}; 72.0E3\frac{J}{mol}; 69.0E3\frac{J}{mol}; 5.0E6\frac{1}{s}; 1.0E7\frac{1}{s}; 5.0E5\frac{1}{s}; 1.4\frac{W}{K}; 3.5\frac{J}{kg\ K}\right]$$

They have been searched for in the following bounds:

	Lower bound	Upper bound
E_{att_1}	1.0E3	2.0E7
E_{att_2}	1.0E3	2.0E7
E_{att_3}	1.0E3	2.0E7
k_{0_1}	5.0E4	5.0E9
k_{0_2}	5.0E4	5.0E9
k_{0_3}	5.0E4	1.0E9
UA	0.1	10
c_{pmix}	1	20

Table 1: Parameters research bounds

These values have been chosen because they include the true parameter values $\boldsymbol{\vartheta}^*$, and to give a high condition number k to the system, to test the script robustness. Additionally, to increase the calculations robustness, logarithmic scaling has been applied to the research bounds. The function used for PE is *lsqnonlin* with *optimoptions('FunctionTolerance',1.0e-12,'OptimalityTolerance',1.0e-12,'StepTolerance',1.0e-12)*.

The controls that have been changed to increase accuracy are: $F, \boldsymbol{C}_{in}, T_{in}, T_j$ for a total of $N_u = 7$, and they have been searched for in the following bounded region:

	Lower bound	Upper bound
F	1.0E-5	1.0E-3
C_{in_A}	1.0E-1	20
C_{in_B}	1	20
C_{in_C}	1E-9	1
C_{in_D}	1E-9	1
T_{in}	350	550
T_j	300	450

Table 2: Controls research bounds

The sensitivity matrix $\boldsymbol{S}$ has been scaled with the parameters values to increase robustness. The function used for OED is fmincon with optimoptions('OptimalityTolerance',1.0e-12,'MaxFunctionEvaluations',420)

The IG sets for both PE and OED have been generated by Hammersley set point method (Wong, Luk and Heng, 1997) in the respective research bounds. From the global pool of 1000 generated Hammersley points, 50 have been picked for each loop with the *randi* function. Thus, for each fast run of the PE and OED loops, 51 IG sets (50 new Hammersley with the previous experiments' best) have been compared to produce the lowest objective function.

4 Results and discussion

In the current section, graphs will be presented, describing the evolution of used controls and estimated parameters with respect to the number of experiments performed during the experimental campaign. On the x-axis, the number of experiments is present. It starts at 1, representing the initial experiment, and increases to 7, representing the last experiment for this analysis. To start up the loop the following ICs have been used:

$$\boldsymbol{y}_0 = \left[3.0\frac{mol}{m^3}; 10.0\frac{mol}{m^3}; 1.0E-6\frac{mol}{m^3}; 1.0E-6\frac{mol}{m^3}; 293.0K\right]$$

$$\boldsymbol{\vartheta}^{IG} = \left[80.0E3\frac{J}{mol}; 100.0E3\frac{J}{mol}; 90.0E3\frac{J}{mol}; 1.5E6\frac{1}{s}; 8.0E6\frac{1}{s}; 8.0E5\frac{1}{s}; 2\frac{W}{K}; 2.5\frac{J}{kg\,K}\right]$$

$$\boldsymbol{u}_0 = \left[6.5E-4\frac{m^3}{s}; 3.0\frac{mol}{m^3}; 10.0\frac{mol}{m^3}; 1.0E-8\frac{mol}{m^3}; 400K; 350K\right]$$

4.1 Model training outputs

The following graphs have been produced with the output data of the PE & OED loop used to estimate the parameters. There will be 5 lines: 1 for each classical criterion (A-, D-, E-,), and 2 more lines, representing 2 additional runs in which the OED loop has been substituted by randomly choosing the controls for the following experiment, in order to verify if the use of the criteria actually provides added value.

4.1.1 Calculated Controls

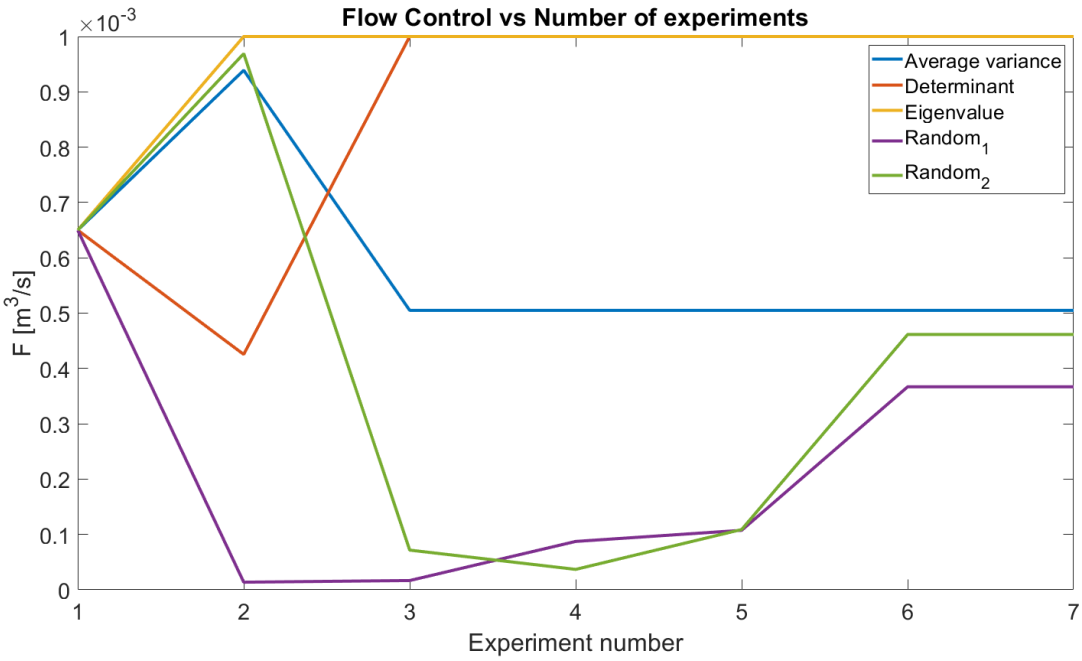

Figure 5: Control on volumetric Flow

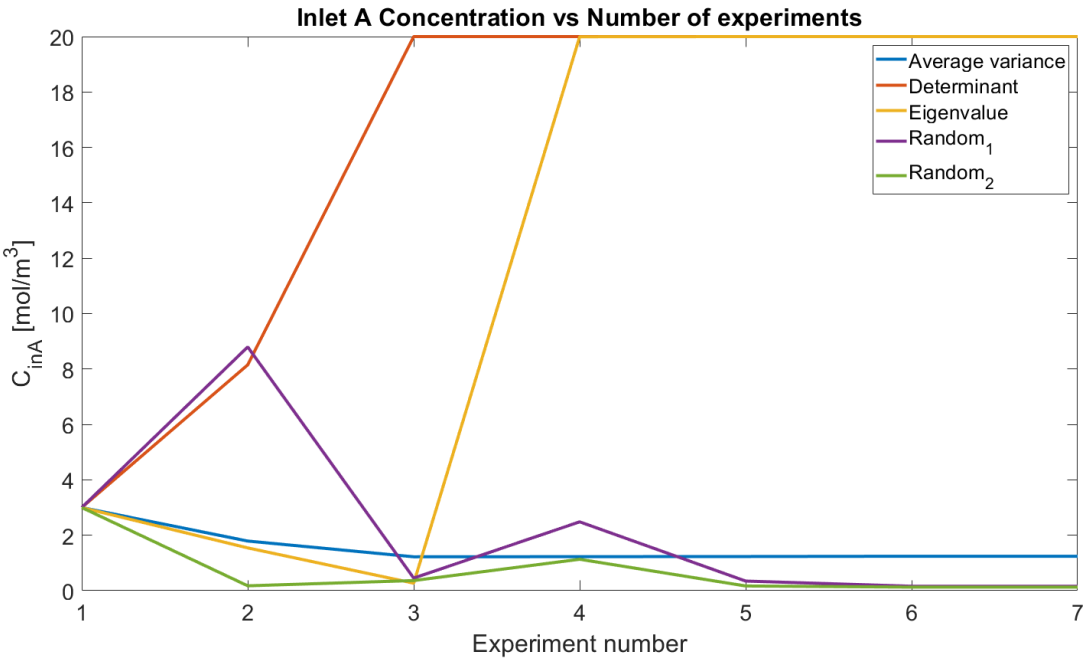

Figure 6: Control on Inlet Concentration of A

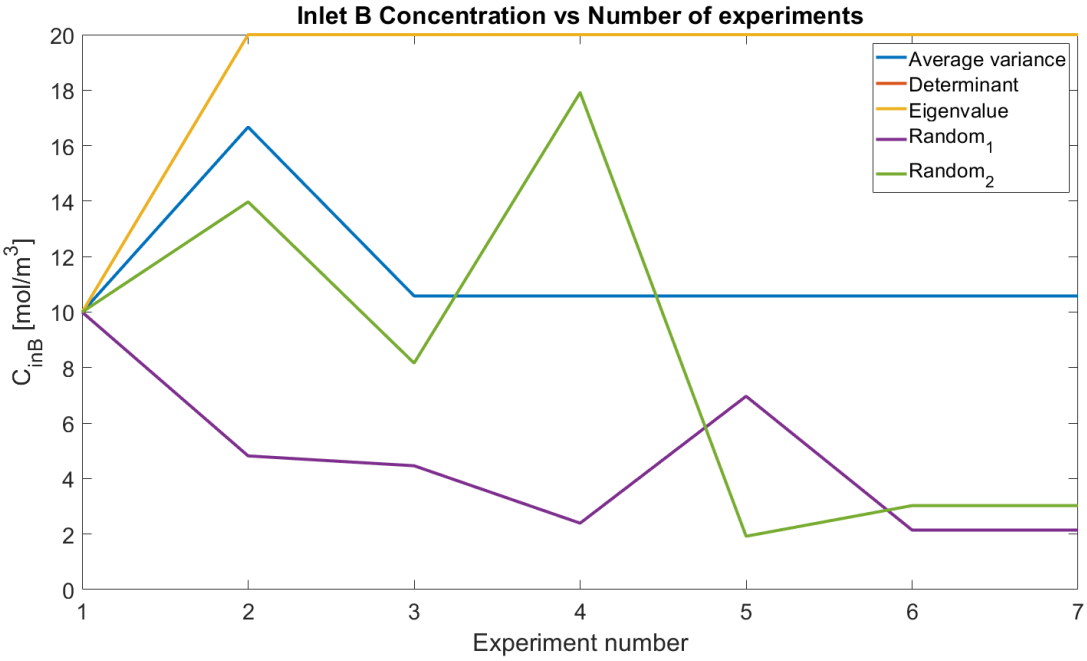

Figure 7: Control on Inlet Concentration of B

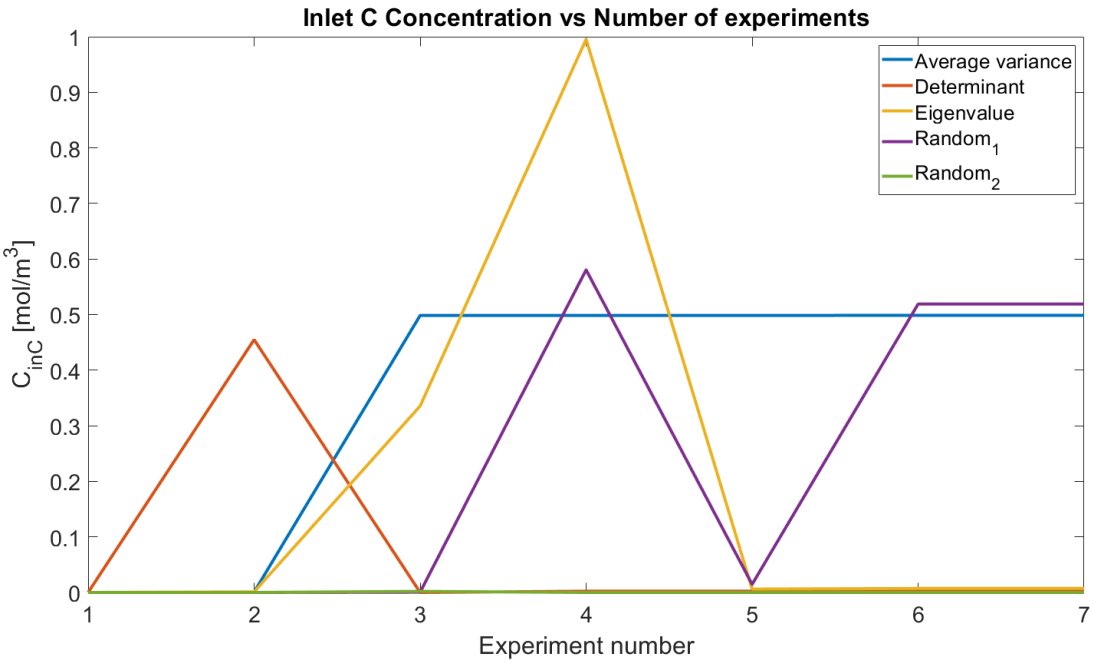

Figure 8: Control on Inlet Concentration of C

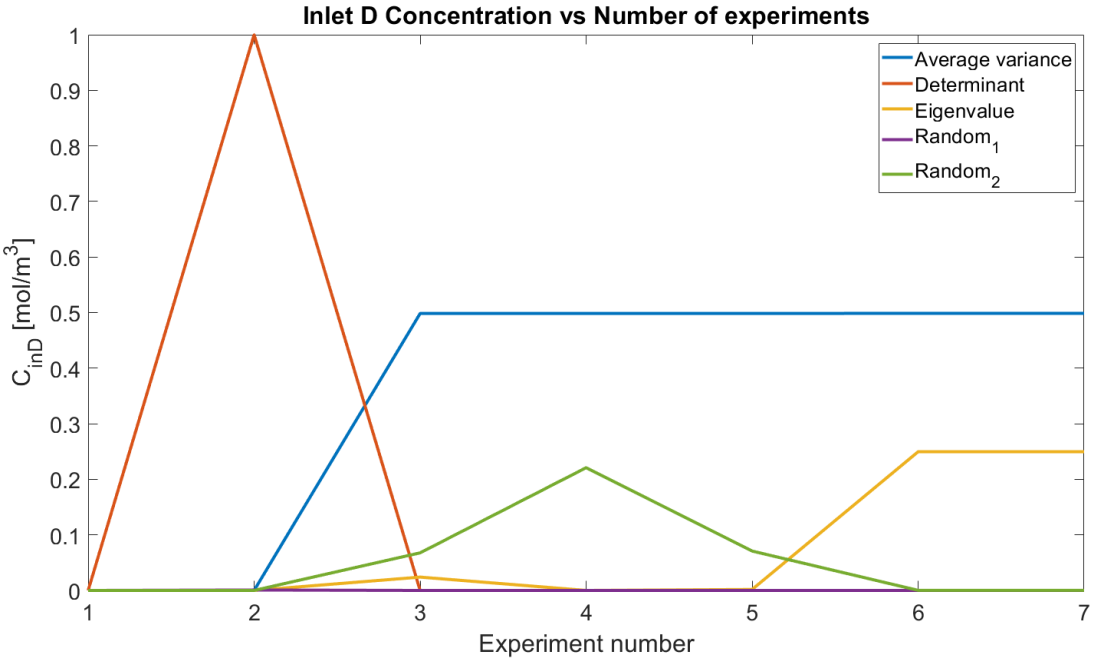

Figure 9: Control on Inlet Concentration of D

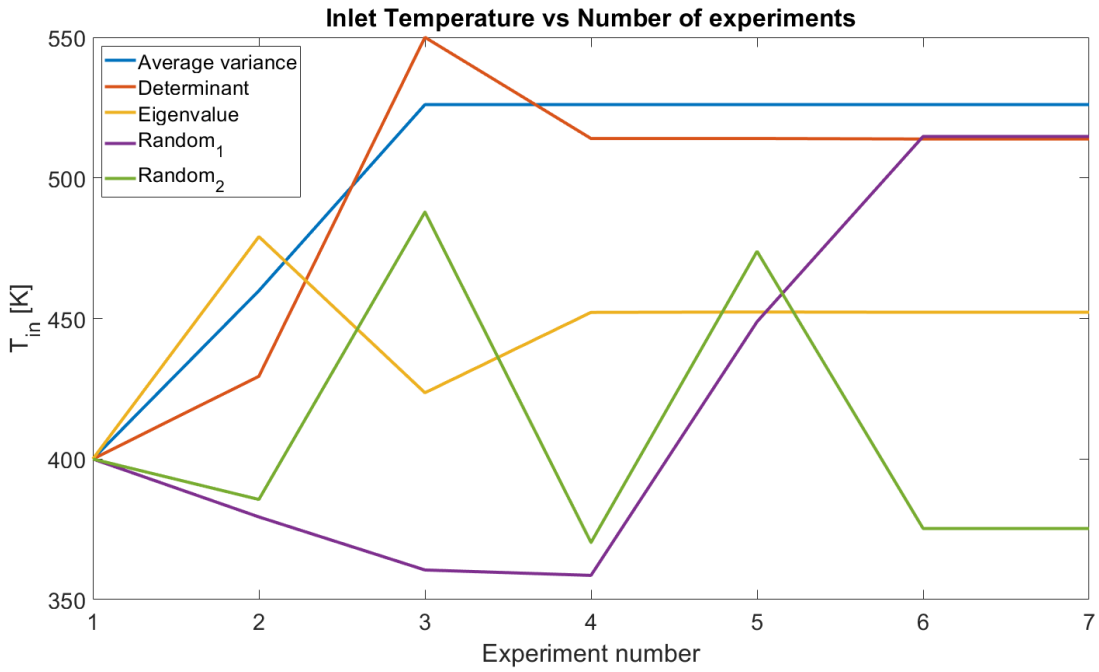

Figure 10: Control on Inlet Temperature

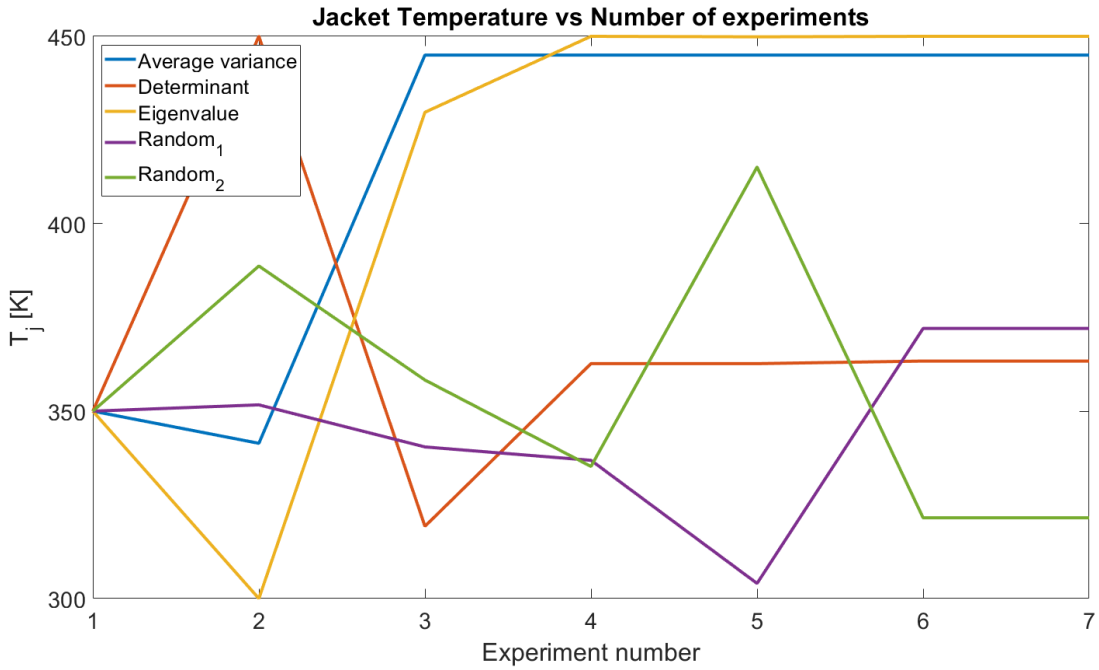

Figure 11: Control on Jacket Temperature

As expected, the lines representing the randomly chosen controls are not constant between one experiment and the following. On the other hand, all 3 of the criteria show an asymptote behavior.

Average variance criterion converges to stable control values at the 3rd experiment, for all controls.

Determinant criterion converges at the 2nd experiment for C_{in_B}, at the 3rd for $F, C_{in_A}, C_{in_C}, C_{in_D}$, and at the 4th for T_{in}, T_j.

Eigenvalue criterion converges at the 2nd experiment for F, C_{in_B}, at the 4th for C_{in_A}, T_{in}, T_j, at the 5th for C_{in_C}, and at the 6th for C_{in_D}.

This asymptote behavior means that, with the current experiment OED IG sets, controls that would yield a bigger $\boldsymbol{FIM}$, and so more information for the PE, have not been found. Then, that is the in-itinere stopping criterion for the experimental campaign: if controls don't change, the maximum of the amount of information obtainable from the current experimental campaign to estimate the parameters has been achieved, thus there is no point in continuing with the experimentation without changing the model or research bounds. It is also important to consider that with the random chosen controls, another stopping criterion in-itinere must be chosen.

4.1.2 Estimated Parameters

In the following graphs, an additional, cyan line representing the true parameters values $\boldsymbol{\vartheta}^*$ is present, for comparison purposes with the estimated parameters. The values used in the model are as follows:

$$\boldsymbol{\vartheta}^* = \left[96.0E3\,\frac{J}{mol}; 72.0E3\,\frac{J}{mol}; 69.0E3\,\frac{J}{mol}; 5.0E6\,\frac{1}{s}; 1.0E7\,\frac{1}{s}; 5.0E5\,\frac{1}{s}; 1.4\,\frac{W}{K}; 3.5\,\frac{J}{kg\ K}\right]$$

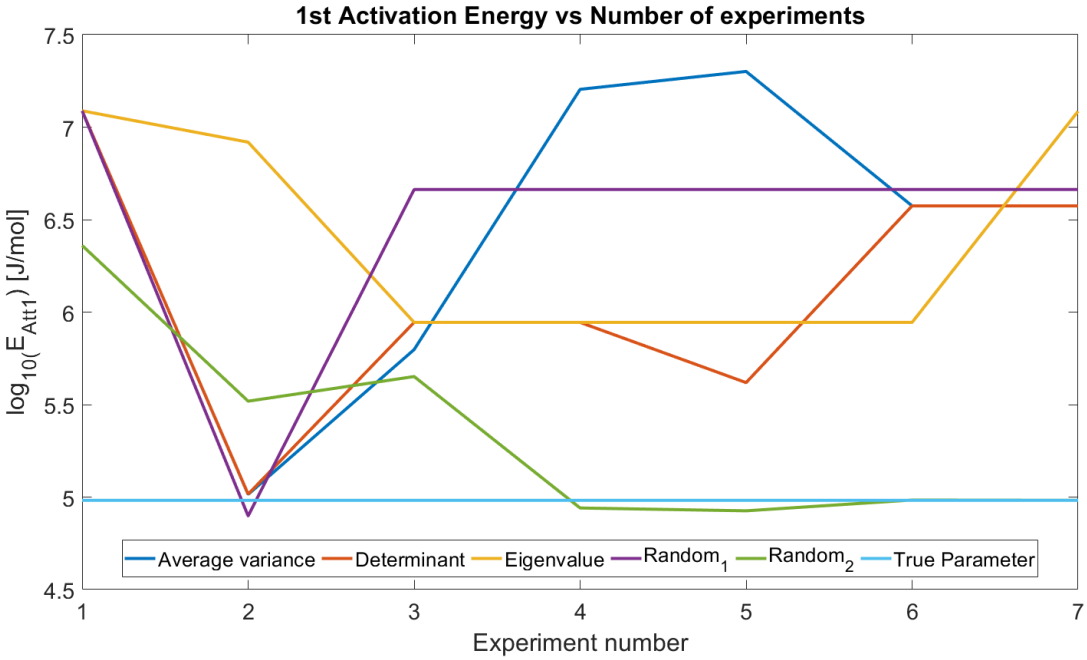

Figure 12: Logarithm of Estimated 1st parameter $\log_{10}(E_{att_1})$

The 1st parameter E_{att_1} calculated values are mostly varying among different experiments and different criteria. This may be an indication of its unidentifiability.

A-criterion varies strongly between experiments, with a minimum at 1.03E+5 and a maximum at 2.00E+7, having an excursion of 2 orders of magnitude.

D-criterion presents an almost converging behavior between experiment 2 and 5, hovering around the 1E+5 order of magnitude, then it diverges to a higher value.

E-criterion converges to 8.80E+5 from experiment 3 to 6, then it shows a jump in 2 orders of magnitude.

The 1st random controls run converges to 4.60E+6 at the 3rd experiment, while the 2nd random controls run converges around the true E_{att_1} value already at experiment 2. The two differing behaviors are due to having better IGs for this parameter during the randomized choice from the pool.

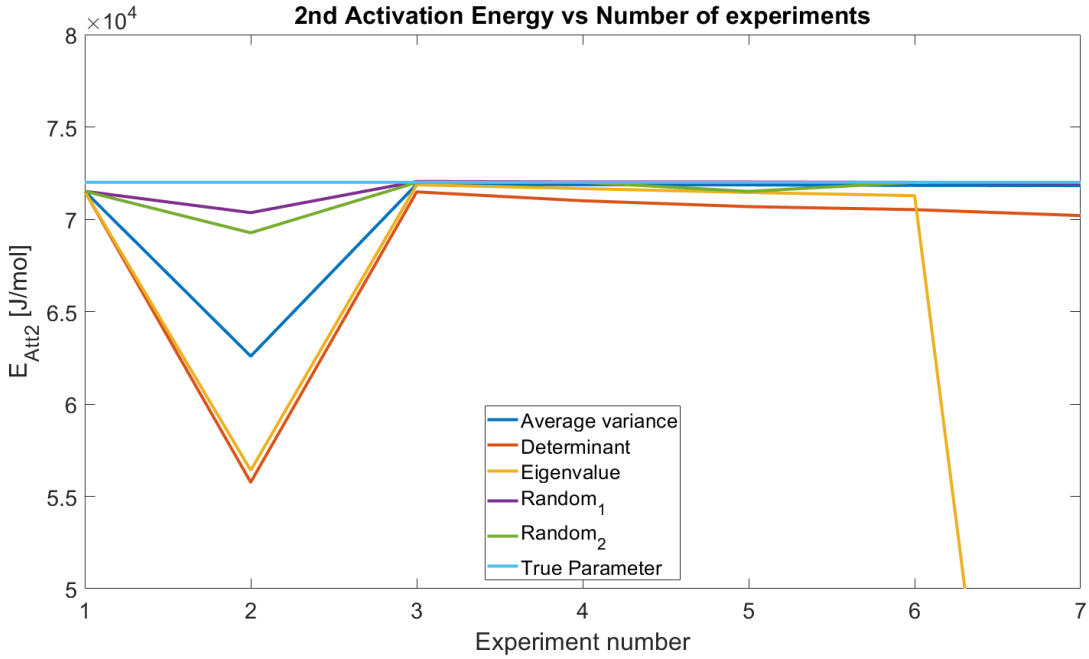

Figure 13: Estimated 2nd parameter E_{att_2}

The 2nd parameter E_{att_2} calculated values at convergence are mostly similar among different experiments and different criteria. They also show little variation when in the non-convergence zone. This may be an indication of its identifiability.

All loop runs converge to a value close to the real one $E_{att_2} = 72.0E3 \frac{J}{mol}$ at experiment 3.

However, the least accurate is D-criterion. Furthermore, E-criterion shows a divergence at the 7th experiment to a value of 1.0E+3. This may be an indication on the robustness of this criterion.

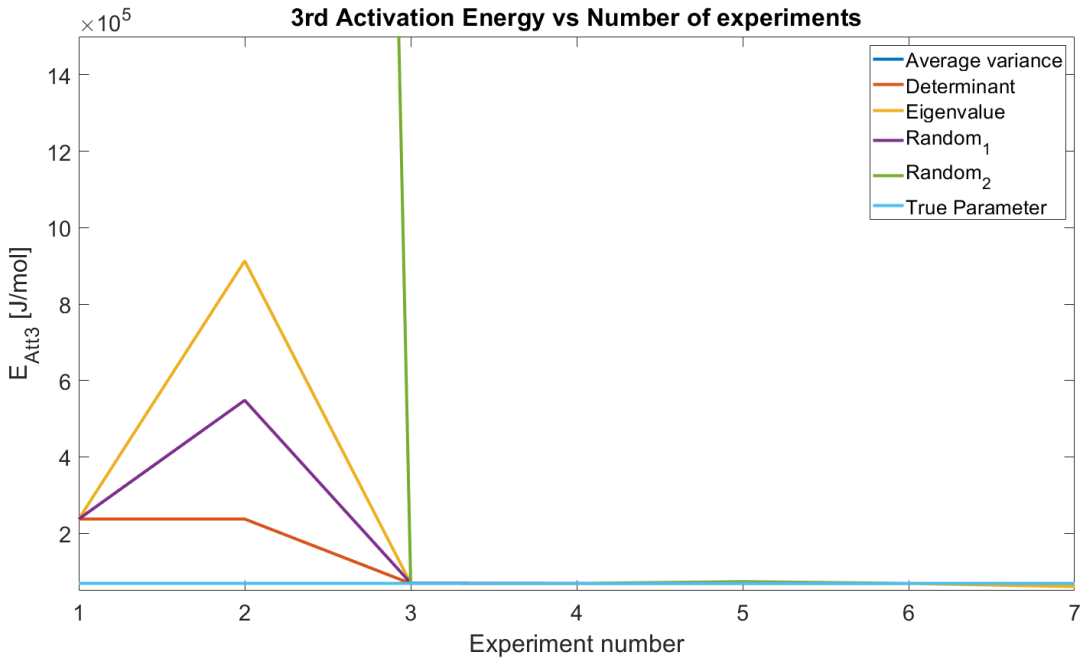

Figure 14: Estimated 3rd parameter E_{att_3}

The 3rd parameter E_{att_3} calculated values are mostly similar among different experiments and different criteria. They also show little variation when in the non-convergence zone. This may be an indication of its identifiability.

All loop runs converge to a value close to the real one $E_{att_3} = 69.0E3 \frac{J}{mol}$ at experiment 3.

Looking at how the 2nd random run behaves in the non-convergence region, i.e.: until the 3rd experiment, reaching 9.06E+6 and 20.0E+7 respectively, values not shown for better viewing. This happens because a different, better IG for this parameter has been used during the 1st experiment. Nonetheless, it converges close to the true parameter value. This may be an indication as to the robustness of the code as to the calculated parameters values when they are identifiable.

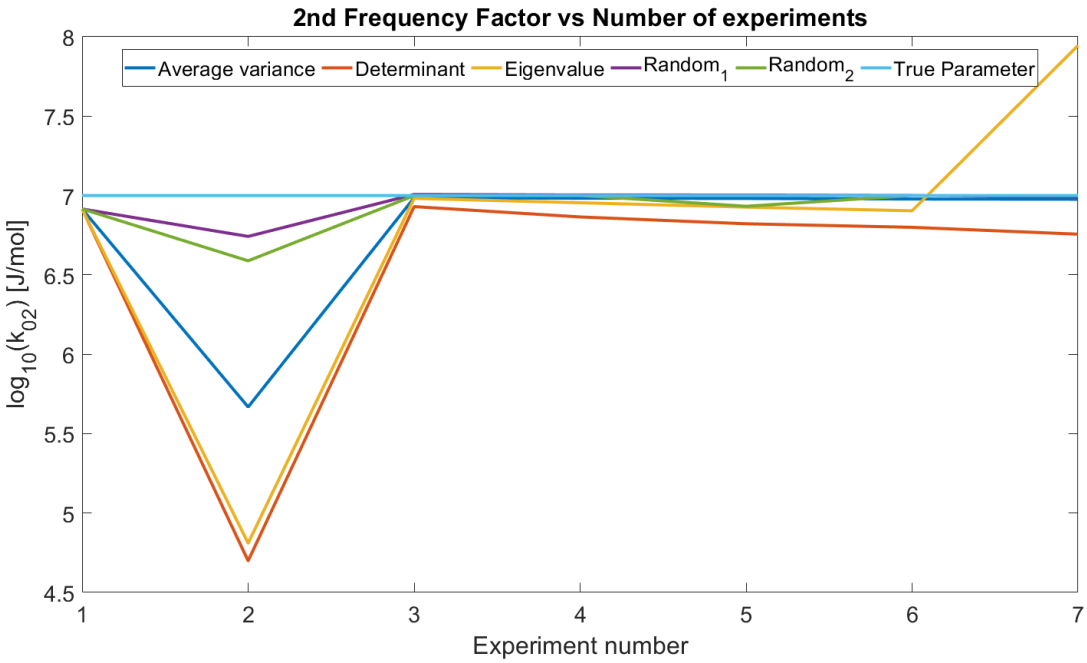

Figure 15: Logarithm of Estimated 4th parameter $\log_{10}(k_{0_1})$

The 4th parameter k_{0_1} shows a behavior similar to E_{att_1}: it shows a high degree of variation in values, i.e.: 5 orders of magnitude, among the different runs and experiments.

A-criterion does not converge until experiment 6.

D-criterion shows a similar behavior.

E-criterion shows convergence during experiment 3 until experiment 6, to a value one order of magnitude lower than the true one, i.e.: 5.47E+5.

The 1st random run converges to a value 1 order of magnitude lower than the true one, i.e.: 2.11E+5 at the 3rd experiment. On the other hand, the 2nd random run converges to a value close to the true one, i.e.: 5.3E+5, at experiment 6. Both these converging behaviors may be due to lucky IGs for PE and lucky controls choice.

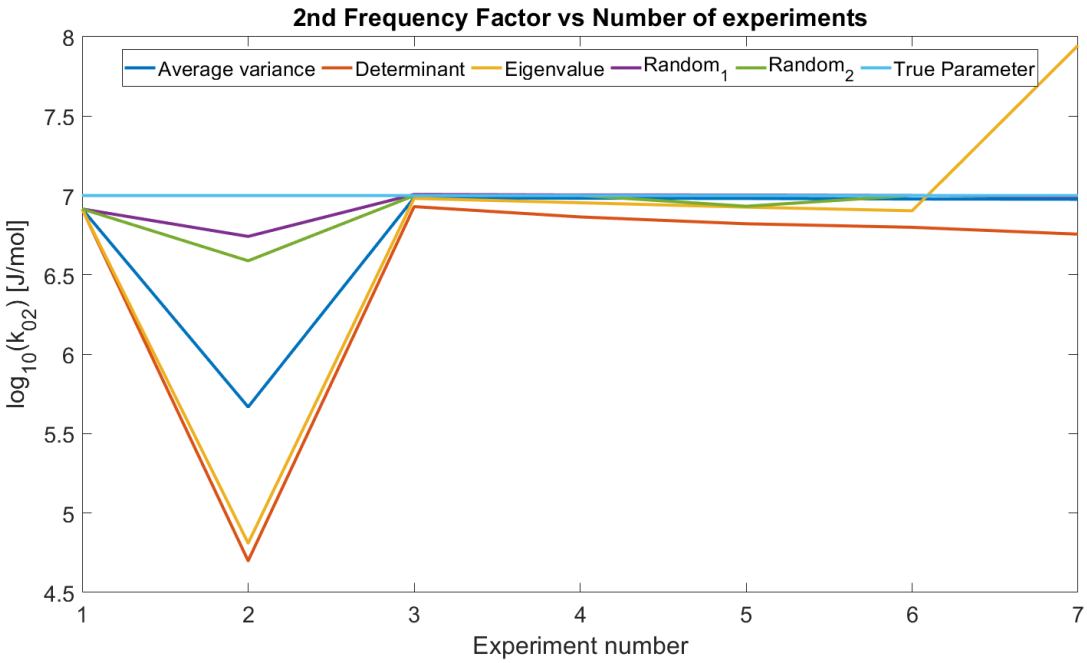

Figure 16: Logarithm of Estimated 5th parameter $\log_{10}(k_{0_2})$

The 5th parameter k_{0_2} shows a behavior similar to E_{att_2}: it shows a small degree of variation in values, i.e.: they oscillate around 9.0E+6, among the different runs and experiments. This may be an indicator as to its identifiability.

Similarly to the previous parameters, all runs converge at experiment 3.

The 2 random runs show high accuracy, reaching a value of circa 1.0E+7, which is the true value.

Among the 3 criterion, A- shows the highest accuracy with 9.5E+6, then E- is in second place with circa 9.0E+6. D-criterion is the least accurate of the 3 for this parameter, reaching a value of circa 6.0E+6. However E-criterion, once again, shows a divergence at the 7th experiment. Again, this may be an indicator of its robustness.

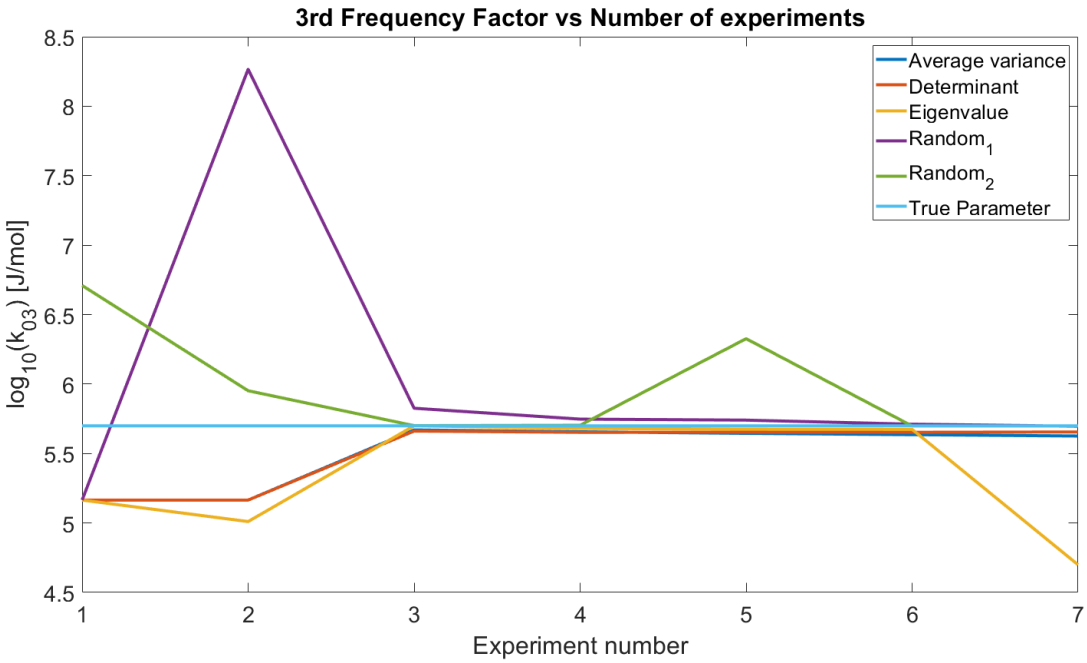

Figure 17: Logarithm of Estimated 6th parameter $\log_{10}(k_{0_3})$

The 6th parameter k_{0_3} shows a behavior similar to E_{att_2}: it shows a small degree of variation in values in the convergence region, i.e.: circa 5.0E+5, which is also very accurate, among all runs and experiments, after the 3rd. This may be an indication as to its identifiability.

For this parameter, the 2 random runs show the worst behavior. The 1st random run diverges for 3 orders of magnitude at experiment 2, reaching 1.84E+8, values not shown for better viewing, while the 2nd random run shows a jump of 1 order of magnitude at experiment 5.

Once again, E-criterion shows a divergence at experiment 7, reaching a value of 5.0E+4, which is its lower research bound. This is further evidence on its lower robustness when compared to the other 2 criteria.

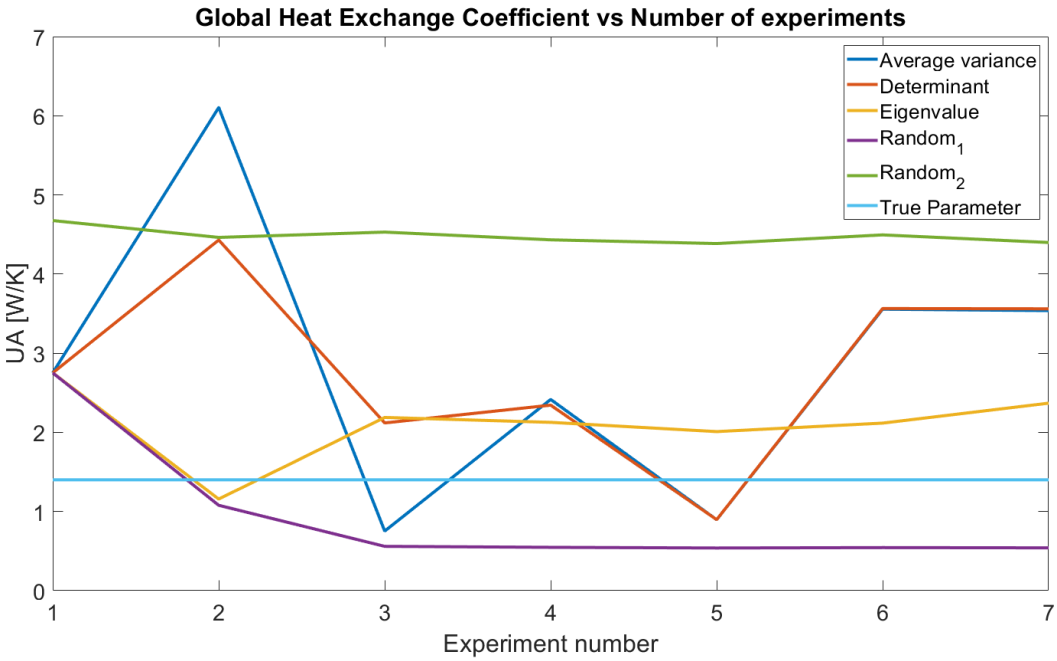

Figure 18: Estimated 7th parameter UA

The 7th parameter UA has a smaller research region when compared to the previous parameters, and its variation in values has a stronger impact on LSQ calculations.

All code runs hover around the true parameter value of 1.4, so this may be an indicator as to the parameter's identifiability.

A- And D- criteria converge to the value of 3.5 at experiment 6. E-criterion converges to a value of circa 2 already at the 3rd experiment. It is the most accurate among the criteria for this parameter. The 1st random run converges at experiment 3 as well, to a value of 0.5. The 2nd random run is stable at circa 4.5 from the 1st experiment.

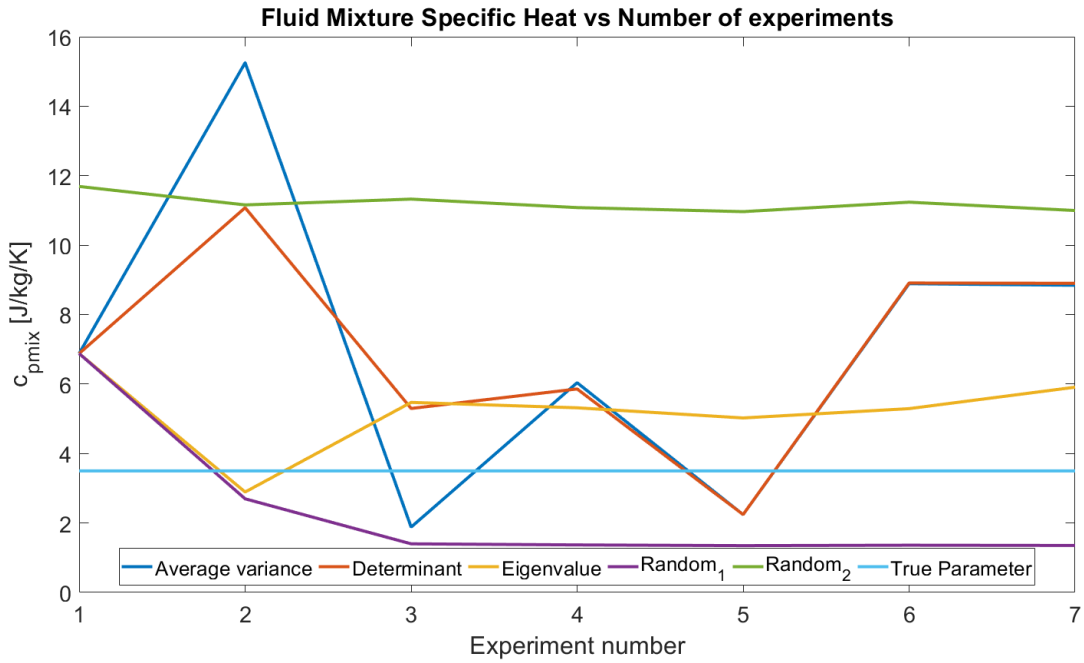

Figure 19: Estimated 8th parameter c_{pmix}

The 8th parameter c_{pmix} has a smaller research region when compared to the previous parameters, thus its variation in values has a stronger impact on LSQ calculations.

All code runs are stable around the true parameter value of 3.5, so this may be an indicator as to the parameter's identifiability.

A- And D- criteria converge to the value of circa 8.8 at experiment 6.

E-criterion converges to a value of circa 5 already at the 3rd experiment. It is the most accurate among the criteria for this parameter.

The 1st random run converges at experiment 3 as well, to a value of 1.3.

The 2nd random run is stable at circa 11.0 from the 1st experiment.

In general, strong variations in the estimated parameters show that their calculated values have a low impact on the LSQ function calculations, thus one can assume those parameters are less relevant from a model identification standpoint, in the chosen operating conditions and research bounds.

Furthermore, certain parameters may be easier to identify and estimate because, thanks to the Chemical Engineer know-how of the phenomena involved, research bounds may be smaller. This is the case of the last two parameters, UA and c_{pmix}, where the simple knowledge of the mixture phase and the literature data has reduced their research region size to 2 and 1 orders of magnitude, respectively. On the other hand, the reaction data, i.e.: $\boldsymbol{E_{Att}}$ and $\boldsymbol{k_0}$, has been kept as generic as possible, with a huge research area of 4 and 5 orders of magnitude respectively, so their identification may be more difficult.

4.2 Validation results

As previously stated, the aim of this thesis is to assess each of the three criteria on the basis of their computational time, robustness and accuracy, in order to verify if the online implementation of the Adaptive Optimal Experimental Design loop may be performed.

The current section is thus divided in 3 parts, where graphs are presented. They result from a data set used for validation, which was in turn produced by implementing 1000 controls sets to the real plant simulation and to the fitted models, i.e.: varying parameters values across loop runs and experiments within the same run, and by then comparing them. The controls sets have been produced by, once again, applying Hammersley set point method to the operating conditions bounds. On the x-axis, the individual experiment number is shown.

Before following through with the assessment, it is important to point out that unfortunately these data are not available during an actual experimental campaign, but they may become available only after it has been completed. Nonetheless, they may be useful for further inquiry on the criteria, whether a new model or plant may be investigated.

4.2.1 Computational time

The first question that will be answered is whether the Adaptive PE & OED loop is fast enough to be implemented online. This depends on the duration of the individual experiment, tuned to the stationary time of the plant, and the duration of calculations for the loop.

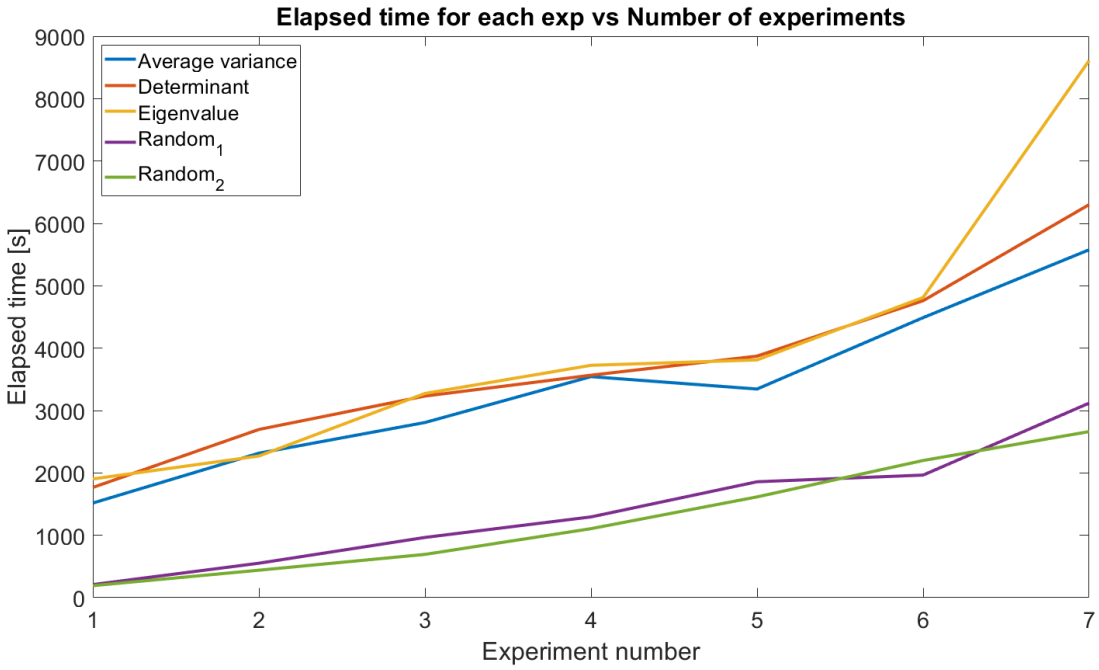

Figure 20: Individual Experiment Elapsed times

In this graph, the computational time in seconds for each individual experiment has been plotted.

The 2 random controls run are visibly faster than the OED criteria. This is due to the fact that, for them, the OED loop has been deactivated altogether and substituted by a random implementation of the controls, thus less calculations have been performed during those runs. Their CPU time is generally about 1 order of magnitude lower.

The three criteria show similar individual experiment computational times. However, A-criterion appears to be the fastest for all experiments, except for the 2nd one. On the other hand, D- and E- criteria show similar

computation times. Furthermore, a steep increase in elapsed time for E-criterion is present at experiment 7. This is consistent with the divergence shown before.

The increase in computational time between experiments for the individual code run is due to the fact that each new PE & OED run considers the current and all the previous experiments data $\boldsymbol{y}_{meas}$. This means that the amount of data processed increases with each experiment.

Oscillations in the increase are most likely due to the "goodness" of the IGs for both parameters and controls. Obviously, the closer the IG is to the local minima of the cost function, the faster the calculations will be.

CPU time may increase with the complexity of the model, that is number of equations considered, complexity of the phenomena considered, number of parameters to be estimated and number of controls used on the plant.

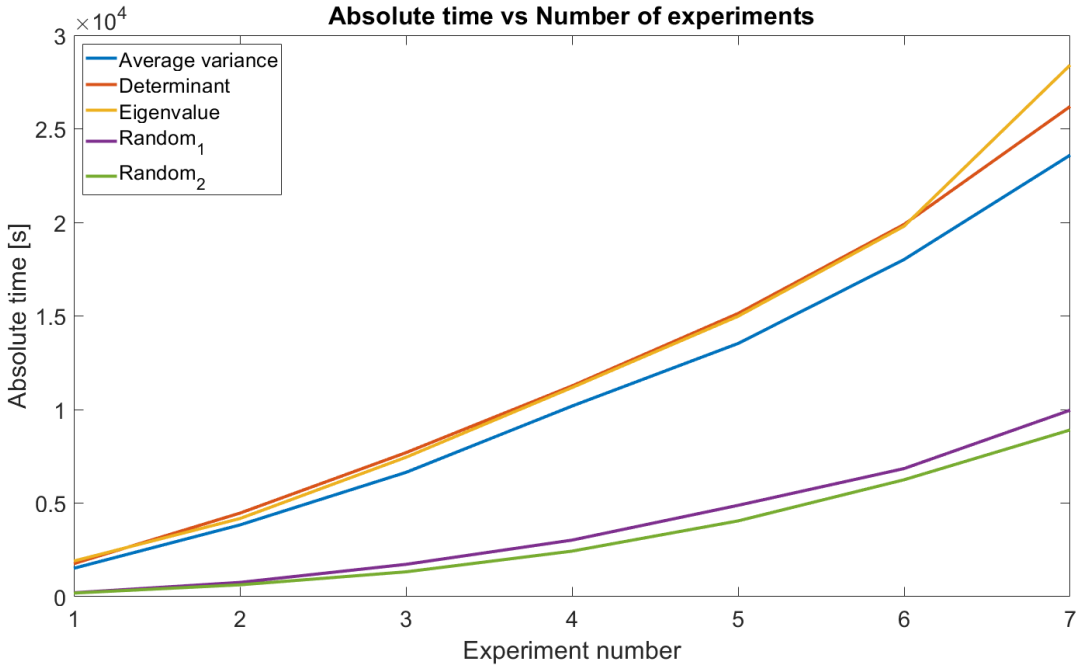

Figure 21: Absolute times

In this graph, the cumulative computational time, i.e.: the amount of time passed since the first experiment, in seconds for each individual experiment has been plotted.

As it could be expected, all runs times increase monotonously.

With the information of the two plots, it could be argued that the Adaptive PE & OED loop can not be run online yet, because CPU time for the individual experiment is longer than the experiment duration. This implies that calculations on the current experiment are not yet finished, i.e.: no new controls are obtained in time for the following one.

Possible solutions for this may be: increase in computational power of the machine used, longer duration of the individual experiment, simplification of the model, tuning of the solvers options, tuning of the IG sets, better implementation of the criteria.

On the other hand, when no OED is performed, calculations are fast enough for online implementation. However, generally, it may be beneficial to choose controls manually instead of randomly, since the chemical process expertise of the user may produce better results.

4.2.2 Robustness

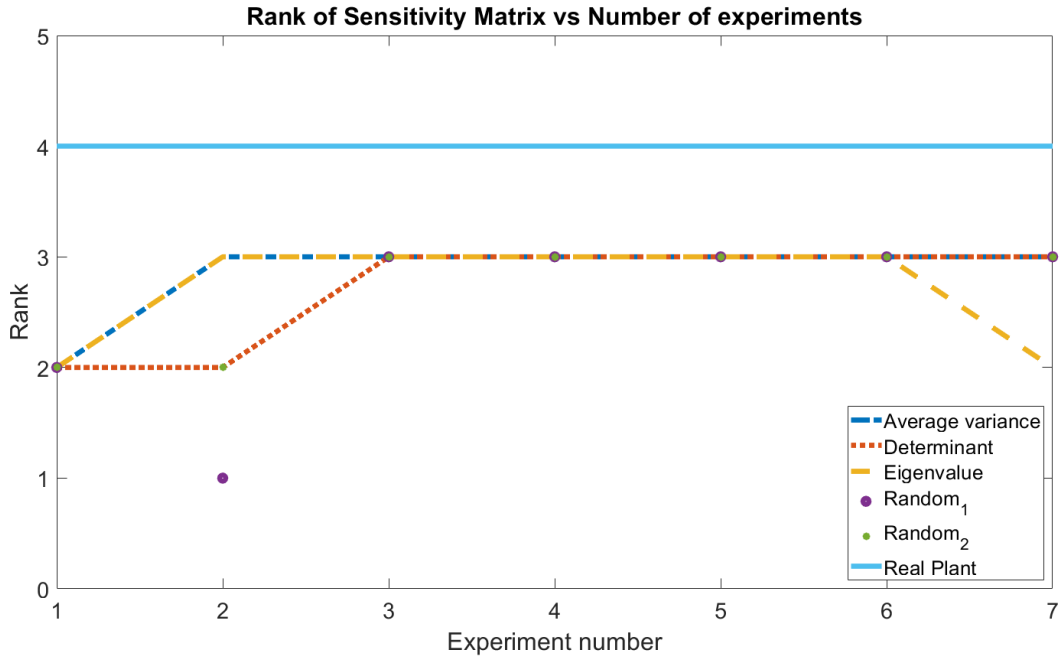

Figure 22: Sensitivity Matrix $\boldsymbol{S}$ Rank

The current plot shows the rank of the Sensitivity Matrix $\boldsymbol{S}$ for each experiment, compared to the maximum rank obtainable on the real plant.

The Sensitivity Matrix $\boldsymbol{S}$ rank has been obtained by running the identifiability analysis on measured data $\boldsymbol{y}_{meas}$, i.e.: test case model with added noise. It has been assumed as the ceiling value for the fitted models Sensitivity Matrix $\boldsymbol{S}$ rank. The identifiable, and thus active, parameters result to be the 3 Activation Energies $\boldsymbol{E_{Att}}$ and the fluid mixture Specific Heat c_{pmix}.

The maximum rank achieved by all runs is 3. The identifiable parameters result to be E_{Att_2}, E_{Att_3}, UA. When the rank is two, E_{Att_2} is not considered identifiable anymore. An inconsistency on the identifiability of UA, c_{pmix} is present between the ceiling rank and the runs rank. This is probably due to the fact that the ε-threshold, accepted from the literature, even though reasonable, is arbitrary. Thus, tweaking it may solve the problem.

A-criterion is the fastest to converge to rank 3, at the 2nd experiment, and is stable until the end of the run. D-criterion converges to rank 3 at experiment 3, and is stable afterwards. E-criterion converges at experiment 2 like A-criterion, however it diverges at the 7th experiment. This is consistent with the previous findings. The 2 random controls runs show convergence to rank 3 after the 3rd experiment. They are slower than A-criterion. It appears that among the criteria, A- is the more robust, and reaches convergence earlier than the random runs.

From this analysis, it results that the stability in values of the estimated parameters among experiments is a relatively good indicator of their identifiability, and of their low linear codependence. As a matter of fact, $E_{Att_2}, E_{Att_3}, UA, c_{pmix}$ were the most stable and accurate, and also the identifiable ones. However, E_{Att_1}, even if identifiable from the measured data analysis, was not stable during parameter estimation. Furthermore, k_{0_1}, k_{0_2} where stable and accurate even if the identifiability analysis discarded them. This shows that identifiable parameters may still be inaccurate, and unidentifiable ones may be accurate as well. Nonetheless, the identifiability of a parameter is a good indicator of its accuracy reliability.

4.2.2 Accuracy

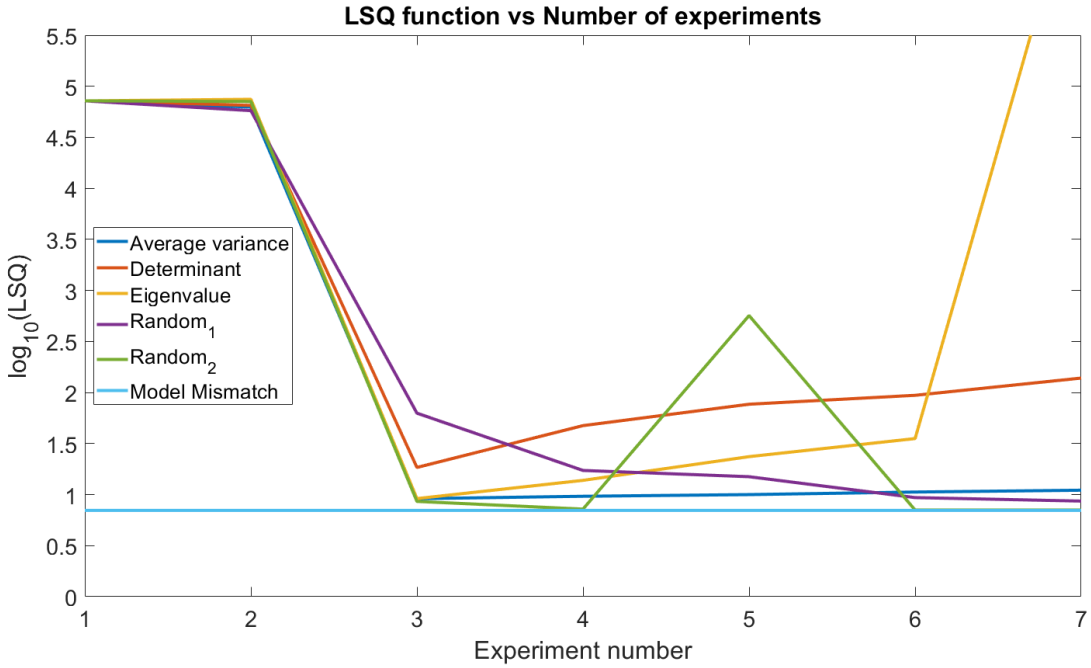

Figure 23: Logarithm of LSQ function

In the current graph, the $\log_{10}(LSQ)$ for the different runs performed are presented.

The Model Mismatch, cyan coloured line results from the comparison between $\boldsymbol{y_{meas}}$ obtained by adding measurement device error noise to the model with the true parameters and $\boldsymbol{y_{meas}}^*$ obtained by eliminating the noise from that same model. It represents the minimum value of the LSQ function, which hopefully the PE & OED loop runs will converge to asymptotically, since it would show that the parameters become more accurate with each experiment.

The first random, purple line run shows a monotonically decreasing behaviour. However, it shows the highest LSQ value across all runs at experiment 3, that is it is the slowest to converge.

On the other hand, the second random, green line run shows a lower LSQ value at experiment 3, comparable to the 3 criteria, which means it has better convergence than the purple line. Nonetheless, a strong divergence of 3 orders of magnitude is present at experiment 5, thus it is less robust. The difference in robustness in estimated parameters model between the two random runs is probably due to the closeness of IG sets to local minima in the parameters research bounds. Except this spike, the green line shows a generally decreasing behaviour as well.

All 3 criteria show a steep decrease in value at experiment 3 as well, and then their LSQ value increases, especially for D- and E- criteria. At experiment 3 most controls and parameters stop changing, and for this reason it has been assumed that the maximum amount of information for the current experimental campaign has been reached. Thus, the increase in value after experiment 3 is due to the fact that the estimated parameters set is still the best available, i.e.: it yields the lowest LSQ on the training data, but it is however worse in predictive power for different operating conditions of the validation set.

Among the 3 criteria, D-criterion is the worst in accuracy since it has the highest LSQ value, and is also has the biggest increase, except for the divergence shown by E-criterion at, once again, experiment 7. Thus, from the standpoint of the validation data set, E-criterion is the less robust of the three.

A-criterion appears to the most accurate of the 3, the more robust, and converges to its minimum value at the same experiment as the other 2.

It is interesting to note that, even though, after experiment 3, the 3 criteria show worse predictive power than the 1st random run, at experiment 3 the $\boldsymbol{LSQ}$ calculated value is the same order of magnitude of the last experiment's $\boldsymbol{LSQ}$ for the 1st random run. This means that all 3 criteria reach a very good accuracy after 3 experiments only, cutting the current experimental campaign, and its supposed costs, by more than half when compared to lucky choices of IG sets, while still being more robust.

4.3 Final remarks on results

In the current section, a summary of the previous results will be presented.

The randomly picked controls runs, even though decisively faster than the PE & OED loop runs, are not robust enough for sensible implementation on a real plant. For matters of costs and safety, it would be unreasonable to let luck pick controls sets randomly from the pool of the possible ones. It would be better to use the Chemical Engineer expertise.

When it comes to the online implementation feasibility, CPU power of the used terminal influences results strongly, thus having a faster machine can only be beneficial. Either times are reduced while keeping the same model, or more complex models can be identified. It is then fundamental to tune the individual experiment, and the experimental campaign duration altogether with respect to the computational speed of the machine, if possible.

E-criterion is the easiest to implement, however shows the least robustness of the three.

D-criterion is robust similarly to A-, however it is less accurate and converges to static controls more slowly than A-.

Finally, A-criterion appears to be the best criterion. It is very easy to implement, because of its simple additive nature. It is the most robust of the 3 criteria, since even singularities in the $\boldsymbol{FIM}$, i.e.: null $\lambda_i(\boldsymbol{FIM})$, do not directly impact its calculation. It is also the most accurate of the 3, as the $\boldsymbol{LSQ}$ function values have shown.

One more thing that is interesting to consider is that the 3 criterion, A-, D-, E-, have a clear cut stopping criteria: when control stop changing, the ceiling amount of potential information has been reached for the current experimental campaign, thus it can and should be stopped. Then, a new experimental campaign, with different operating conditions, different research bounds, and a model with new phenomena considered, may be run, while keeping the identified, accurate parameters assumed as constant.

When random, or user-chosen controls are used, no stopping criteria is readily available, so a new metric must be defined on the basis of which the decision of stopping the experimental campaign may be taken.

5 Further steps and challenges

A general improvement on the robustness, speed, accuracy of the code can be performed. In particular, one could tweak the implementation of the 3 criteria, the method of generation and choice of the IG sets for both PE and OED. A different regularization technique, such as Truncated Singular Value Decomposition or Tikhonov Regularization, could be used. Different solvers and solver options could be used. Additional criteria could be assessed. Different research bounds, experiment and experimental campaign durations could be chosen.

A natural improvement upon the work of this thesis would be to implement the code as a whole in Python, in accordance with the work the Fachgebiet Dynamik und Betrieb technischer Anlagen is performing on the C++ interface of Mosaic Modelling, since its solvers are more robust and powerful.

New and more complex models could be investigated. New physical phenomena such as material transport phenomena could be added, non-idealities, more complex reaction schemes, pressure head losses could be considered.

An important additional step is to actually implement the code online, on a real plant. When that will be the case, a OPC-UA framework for communication between measuring device and computing terminal could be useful.

As previously stated, this project is part of Collaborative Research Centre/Transregio 63 “Integrated Chemical Processes in Liquid Multiphase Systems” (InPROMPT). The eventual aim of this thesis would be to implement the code online to identify the model describing the real InPROMPT plant for Hydroformylation of Alkenes.

Appendix A
MatLab® Optimal Experimental Design script

In the current section, part of the script used for this thesis work will be presented.

I did not write the whole of the script used. For this reason, in the current section only .m files which I took part in are shown.

Authors:
López Cárdenas, Diana Carolina, Systematic evaluation of ill-posed problems in model-based parameter estimation and experimental design, https://depositonce.tu-berlin.de/handle/11303/6056;
Volodymyr Kozachynskyi: volodymyr.kozachynskyi@tu-berlin.de

Technische Universität Berlin, Process Dynamics and Operations Group

The parts not written by me are discolored in grey.

A.1 Main File run_loop_CSTR_true.m

```
function run_loop_CSTR_true( num_hammersley_global,
num_hammersley_picked, ChooseCriterion, debug_flag )
format shorteng

    current_file_path = fileparts(mfilename('fullpath'));
    add_paths();

    try

        reactor_type = 'CSTR_true';

        sim_outs = simulation_output.empty;
        pe_outs_FH = pe_output.empty;
        pe_outs_long = pe_output.empty;
        ss_outs_pe_FH = ss_output.empty;
        ss_outs_pe_long = ss_output.empty;

        doe_outs_FH = doe_output.empty;
        doe_outs_long = doe_output.empty;
        experiments = experiment.empty;
        failed_hammersleys_PE = [];
        failed_hammersleys_doe = [];
        Elapsed_times = [];

        best_doe = doe_output.empty;

        start_time = datetime('now');

        best_res_PE_FH = Inf;
        best_fval_doe_FH = Inf;

        initial_set_PE  = set_hammersley('PE',num_hammersley_global);
        initial_set_OED  = set_hammersley('OED',num_hammersley_global);
```

```
        for loop=1:7 % #experiments aka #loops PE+OED
            tic
            if loop == 1
                [y_init, controls_init, time_vector, sigma, true_param,
param_init ] = set_initials();
                sim_in = simulation_input(time_vector, y_init,
true_param, controls_init, sigma, 'big');
            else
                [~, ~, time_vector, sigma, true_param, param_init ] =
set_initials(); %controls are taken from last output of OED
                yreal = sim_out.y_precise;
                y0new = yreal(end,:);
                sim_in = simulation_input(time_vector, y0new,
true_param, doe_out_long.controls, sigma, 'big');
            end
            sim_out = do_plant_simulation(sim_in);
            sim_outs = [sim_outs, sim_out];
            experiments = [experiments; sim_out.experiment];
            % END sim+noise

            % START PE fast hammer
            active = ones(length(param_init),1);
            if loop == 1
                pe_in = pe_input(param_init, active, experiments,
debug_flag, 1);
            else
                pe_in = pe_input(best_IGs_pe_FH, active,
experiments,debug_flag,  1);
            end
            disp(['Doing first PE. Current Loop ',num2str(loop)]);
            try
                [best_pe_FH] = do_parameter_estimation(pe_in);
                best_res_PE_FH=sum(abs(best_pe_FH.solver_output{3}));
            catch err
                warning('Failed first PE')
                continue
            end

            randHammer =
randi(num_hammersley_global,num_hammersley_picked,1);

            i =1;
            disp(['Doing FH PE. Current Loop ',num2str(loop)]);
            while i <= num_hammersley_picked
                pe_in = pe_input(initial_set_PE(:,randHammer(i)),
active, experiments, debug_flag,  1);
                try
                    disp(['Current Loop ',num2str(loop), '. Current
Hammersley PE ', num2str(i)]);
                    [pe_out_FH] = do_parameter_estimation(pe_in);
                catch err
                    failed_hammersleys_PE = [failed_hammersleys_PE,err];
                    warning('Failed PE_FH')
                    continue
                end
                PE_res_FH=sum(abs(pe_out_FH.solver_output{3}));
```

```
            if PE_res_FH < best_res_PE_FH
                best_res_PE_FH = PE_res_FH;
                best_pe_FH = pe_out_FH;
                best_IGs_pe_FH_i = randHammer(i);
                best_IGs_pe_FH = initial_set_PE(:,best_IGs_pe_FH_i);
            end
            i = i+1;
        end
        pe_outs_FH = [pe_outs_FH, best_pe_FH];
        % END PE fast hammer

        % START SSS on PE FH
        disp(['Doing SSS on PE FH. Current Loop ',num2str(loop)]);
        ss_in_pe_FH = ss_input(best_pe_FH.params, experiments);
        ss_out_pe_FH = do_subset_selection(ss_in_pe_FH);
        ss_outs_pe_FH = [ss_outs_pe_FH, ss_out_pe_FH];
        % END SSS on PE FH

        % START long PE
        disp(['Doing long PE. Current Loop ',num2str(loop)]);
        pe_in_long = pe_input(best_pe_FH.params,
ones(length(param_init),1), experiments, debug_flag, 0);
        [pe_out_long] = do_parameter_estimation(pe_in_long);
        pe_outs_long = [pe_outs_long, pe_out_long];
        % END long PE

         % START SSS on PE long
        disp(['Doing SSS on PE long. Current Loop ',num2str(loop)]);
        ss_in_pe_long = ss_input(pe_out_long.params, experiments);
        ss_out_pe_long = do_subset_selection(ss_in_pe_long);
        ss_outs_pe_long = [ss_outs_pe_long, ss_out_pe_long];
        % END SSS on PE FH

        % START OED FH
        [y_init, ~, time_vector, sigma, ~, ~ ] = set_initials();
        doe_sim_in = simulation_input(time_vector, y_init,
pe_out_long.params, [], sigma, 'small');

        if loop == 1
            doe_in = doe_input(controls_init, ss_out_pe_long.active,
debug_flag, doe_sim_in, 1);
        else
            doe_in = doe_input(best_doe_FH.controls,
ss_out_pe_long.active, debug_flag, doe_sim_in, 1);
        end
        disp(['Doing first OED on active set. Current Loop
',num2str(loop)]);
        try
            [best_doe_FH] = do_design_experiments(doe_in,
ChooseCriterion);
            best_fval_doe_FH=best_doe_FH.solver_output{2};
        catch err
            warning('Failed first OED')
            continue
        end
```

```
            randHammer =
randi(num_hammersley_global,num_hammersley_picked,1);

            i=1;
            disp(['Doing FH OED. Current Loop ',num2str(loop)]);
            while i <= num_hammersley_picked
                doe_in = doe_input(initial_set_OED(:,randHammer(i)),
ss_out_pe_long.active, debug_flag, doe_sim_in, 1);
                try
                    disp(['Current Loop ',num2str(loop), '. Current
Hammersley OED ', num2str(i)]);
                    [doe_out_FH] = do_design_experiments(doe_in,
ChooseCriterion);
                catch err
                    failed_hammersleys_doe =
[failed_hammersleys_doe,err];
                    continue
                end
                fval_doe_FH=doe_out_FH.solver_output{2};
                if fval_doe_FH < best_fval_doe_FH
                    best_fval_doe_FH = fval_doe_FH;
                    best_doe_FH = doe_out_FH;
                    best_IGs_doe_FH_i = randHammer(i);
                    best_IGs_doe_FH =
initial_set_OED(:,best_IGs_doe_FH_i);
                end
                i = i+1;
            end
            doe_outs_FH = [doe_outs_FH, best_doe_FH];
            % END OED FH

            % START OED final
            disp(['Doing long OED on active set. Current Loop
',num2str(loop)]);
            doe_sim_in = simulation_input(time_vector, y_init,
pe_out_long.params, [], sigma, 'small');
            doe_in_long = doe_input(best_doe_FH.controls,
ss_out_pe_long.active, debug_flag, doe_sim_in, 0);
            try
                [doe_out_long] = do_design_experiments(doe_in_long,
ChooseCriterion);
            catch err
                warning('OED Failed')
            end
            doe_outs_long = [doe_outs_long, doe_out_long];
            % END OED final

        Elapsed_time = toc;
        Elapsed_times = [Elapsed_times, Elapsed_time];
        end

        end_time = datetime('now');
        exp_duration_time = end_time - start_time;

        date = datestr(datetime('now'),'yyyy_mm_dd_HH_MM_SS');
        save_path =
fullfile(current_file_path,'out',strcat('out_',date));
```

```
        save(save_path);

    catch err
        if debug_flag
            rethrow(err)
        else
            end_time = datetime('now');
            exp_duration_time = end_time - start_time;

            date = datestr(datetime('now'),'yyyy_mm_dd_HH_MM_SS');
            save_path =
fullfile(current_file_path,'out',strcat('out_error',date));
            save(save_path);
        end
    end

end
```

A.2 Optimal Experimental Design do_design_experiments.m

```
function[doe_out] = do_design_experiments(doe_in,ChooseCriterion)

    [ ControlLB, ControlUB ] = set_bounds( 'OED' );
    warning verbose;

    options = optimoptions('fmincon','Display','iter-detailed');
    options.OptimalityTolerance = 1.000000e-12;
    options.MaxFunctionEvaluations = 420;

    if doe_in.fast_hammersley
        options.MaxIterations = 10;
        warning('error', 'MATLAB:ode15s:IntegrationTolNotMet');
    end

    if doe_in.debug_flag
        options.MaxIterations = 1;
        options.MaxFunctionEvaluations = 1;
    end

    % Optimizer
    try
        if ChooseCriterion == 'A'
            [controls,resnorm,residual,exitflag,output,lambda,jacobian]
= ...
            fmincon(@(IterControls)A_criterion(IterControls,
doe_in),doe_in.controls,[],[],[],[], ControlLB, ControlUB, [], options);
        elseif ChooseCriterion == 'D'
            [controls,resnorm,residual,exitflag,output,lambda,jacobian]
= ...
            fmincon(@(IterControls)D_criterion(IterControls,
doe_in),doe_in.controls,[],[],[],[], ControlLB, ControlUB, [], options);
        elseif ChooseCriterion == 'E'
            [controls,resnorm,residual,exitflag,output,lambda,jacobian]
= ...
            fmincon(@(IterControls)E_criterion(IterControls,
doe_in),doe_in.controls,[],[],[],[], ControlLB, ControlUB, [], options);
        end

    catch err
        warning('ode15s failed during OED at given Hammersley');
        warning('on');
        rethrow(err);
    end
    warning('on');

    solver_output =
{controls,resnorm,residual,exitflag,output,lambda,jacobian};

    disp('Optimizer optimized')

    doe_out = doe_output(controls, solver_output, doe_in);

end
```

```
function [Trace] = A_criterion(iter_controls, doe_in)
% A- Average Variance
    doe_in.sim_input.controls = iter_controls;

    [~,y] = do_solve_eqs(doe_in.sim_input);

    if isnan(y)
        Trace = NaN;
        return
    end

    exp = experiment(doe_in.sim_input.time_vector, y,
doe_in.sim_input.y_init, ...
        doe_in.sim_input.controls, doe_in.sim_input.sigma);

    dydp = do_sensitivity_matrix(doe_in.sim_input.params, exp,
doe_in.active);

    par_value = [];
    active_index = find(doe_in.active==1);
    for i=1:length(active_index)
        par_value(i) =  doe_in.sim_input.params(active_index(i));
    end
    scaling_matrix = kron(par_value,ones(length(dydp(:,1)),1));
    dydp_scaled = dydp .* scaling_matrix;
    % FIM
    MeasurementMatrix = eye(size(dydp,1),size(dydp,1));

    for i=1:size(dydp,1)
        MeasurementMatrix(i,i) = (exp.sigma(mod(i-
1,exp.num_meas_var)+1)).^2;
    end

    % Calculate FIM
    FIM = dydp_scaled' * inv(MeasurementMatrix) * dydp_scaled;
    FIMinv = full(inv(FIM));
    Trace = trace(FIMinv) / length(find(doe_in.active==1));
    Trace = abs(Trace);

end

function [Determinant] = D_criterion(iter_controls, doe_in)
% D- Determinant
    doe_in.sim_input.controls = iter_controls;

     [~,y] = do_solve_eqs(doe_in.sim_input);

    if isnan(y)
        Determinant = NaN;
        return
    end

    exp = experiment(doe_in.sim_input.time_vector, y,
doe_in.sim_input.y_init, ...
        doe_in.sim_input.controls, doe_in.sim_input.sigma);
```

```
    dydp = do_sensitivity_matrix(doe_in.sim_input.params, exp,
doe_in.active);

    par_value = [];
    active_index = find(doe_in.active==1);

    for i=1:length(active_index)
        par_value(i) =  doe_in.sim_input.params(active_index(i));
    end

    scaling_matrix = kron(par_value,ones(length(dydp(:,1)),1));
    dydp_scaled = dydp .* scaling_matrix;
    % FIM
    MeasurementMatrix = eye(size(dydp,1),size(dydp,1));

    for i=1:size(dydp,1)
        MeasurementMatrix(i,i) = (exp.sigma(mod(i-
1,exp.num_meas_var)+1)).^2;
    end

    % Calculate FIM
    FIM = dydp_scaled' * inv(MeasurementMatrix) * dydp_scaled;
    FIMinv = full(inv(FIM));
    Determinant = (det(FIMinv)) ^ length(find(doe_in.active==1));

end

function [Eigenvalue] = E_criterion(iter_controls, doe_in)
% E- Eigenvalue
    doe_in.sim_input.controls = iter_controls;

    [~,y] = do_solve_eqs(doe_in.sim_input);

    if isnan(y)
        Eigenvalue = NaN;
        return
    end

    exp = experiment(doe_in.sim_input.time_vector, y,
doe_in.sim_input.y_init, ...
        doe_in.sim_input.controls, doe_in.sim_input.sigma);

    dydp = do_sensitivity_matrix(doe_in.sim_input.params, exp,
doe_in.active);

    par_value = [];
    active_index = find(doe_in.active==1);
    for i=1:length(active_index)
        par_value(i) =  doe_in.sim_input.params(active_index(i));
    end

    scaling_matrix = kron(par_value,ones(length(dydp(:,1)),1));
    dydp_scaled = dydp .* scaling_matrix;
    % FIM
    MeasurementMatrix = eye(size(dydp,1),size(dydp,1));

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

```
        MeasurementMatrix(i,i) = (exp.sigma(mod(i-
1,exp.num_meas_var)+1)).^2;
    end

    % Calculate FIM
    FIM = dydp_scaled' * inv(MeasurementMatrix) * dydp_scaled;
%#ok<*MINV>
    FIMinv = full(inv(FIM));
    [eVecs,eVals] = eig(FIMinv);
    eValsdiag = diag(eVals);
    Eigenvalue = max(eValsdiag);

end
```

A.3 Parameters & Controls printer aux_print_p_u_new.m

```
close all
clear
clc

load('C:\Users\Sergio\Documents\THESIS\Sergio_Loop_new\learning_results\
50results.mat')

limits = [1 7];

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

pe1_A =     [pe_outs_long_A(1, 1).params(1,1);pe_outs_long_A(1,
2).params(1,1);pe_outs_long_A(1, 3).params(1,1);pe_outs_long_A(1,
4).params(1,1);pe_outs_long_A(1, 5).params(1,1);pe_outs_long_A(1,
6).params(1,1);pe_outs_long_A(1, 7).params(1,1)];
pe1_D =     [pe_outs_long_D(1, 1).params(1,1);pe_outs_long_D(1,
2).params(1,1);pe_outs_long_D(1, 3).params(1,1);pe_outs_long_D(1,
4).params(1,1);pe_outs_long_D(1, 5).params(1,1);pe_outs_long_D(1,
6).params(1,1);pe_outs_long_D(1, 7).params(1,1)];
pe1_E =     [pe_outs_long_E(1, 1).params(1,1);pe_outs_long_E(1,
2).params(1,1);pe_outs_long_E(1, 3).params(1,1);pe_outs_long_E(1,
4).params(1,1);pe_outs_long_E(1, 5).params(1,1);pe_outs_long_E(1,
6).params(1,1);pe_outs_long_E(1, 7).params(1,1)];
pe1_rand1 = [pe_outs_long_rand1(1, 1).params(1,1);pe_outs_long_rand1(1,
2).params(1,1);pe_outs_long_rand1(1,
3).params(1,1);pe_outs_long_rand1(1,
4).params(1,1);pe_outs_long_rand1(1,
5).params(1,1);pe_outs_long_rand1(1,
6).params(1,1);pe_outs_long_rand1(1, 7).params(1,1)];
pe1_rand4 = [pe_outs_long_rand4(1, 1).params(1,1);pe_outs_long_rand4(1,
2).params(1,1);pe_outs_long_rand4(1,
3).params(1,1);pe_outs_long_rand4(1,
4).params(1,1);pe_outs_long_rand4(1,
5).params(1,1);pe_outs_long_rand4(1,
6).params(1,1);pe_outs_long_rand4(1, 7).params(1,1)];
pe1_true = ones(length(pe1_rand4),1)*true_param(1);

pe1_global = [pe1_A, pe1_D, pe1_E, pe1_rand1, pe1_rand4, pe1_true];
pe1_log10_global = log10(pe1_global);
n_exps = linspace(1,length(pe1_A),length(pe1_A));

figure(1)
plot(n_exps,pe1_log10_global,'LineWidth',3)
xlabel('Experiment number'),ylabel('log_1_0_(E_A_t_t_1) [J/mol]');
xlim(limits);
ax = gca;
ax.XTick = n_exps;
title('1st Activation Energy vs Number of experiments');
legend 'Average variance' Determinant Eigenvalue Random_1 Random_2 'True
Parameter'

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

```
pe2_A =     [pe_outs_long_A(1, 1).params(2,1);pe_outs_long_A(1,
2).params(2,1);pe_outs_long_A(1, 3).params(2,1);pe_outs_long_A(1,
4).params(2,1);pe_outs_long_A(1, 5).params(2,1);pe_outs_long_A(1,
6).params(2,1);pe_outs_long_A(1, 7).params(2,1)];
pe2_D =     [pe_outs_long_D(1, 1).params(2,1);pe_outs_long_D(1,
2).params(2,1);pe_outs_long_D(1, 3).params(2,1);pe_outs_long_D(1,
4).params(2,1);pe_outs_long_D(1, 5).params(2,1);pe_outs_long_D(1,
6).params(2,1);pe_outs_long_D(1, 7).params(2,1)];
pe2_E =     [pe_outs_long_E(1, 1).params(2,1);pe_outs_long_E(1,
2).params(2,1);pe_outs_long_E(1, 3).params(2,1);pe_outs_long_E(1,
4).params(2,1);pe_outs_long_E(1, 5).params(2,1);pe_outs_long_E(1,
6).params(2,1);pe_outs_long_E(1, 7).params(2,1)];
pe2_rand1 = [pe_outs_long_rand1(1, 1).params(2,1);pe_outs_long_rand1(1,
2).params(2,1);pe_outs_long_rand1(1,
3).params(2,1);pe_outs_long_rand1(1,
4).params(2,1);pe_outs_long_rand1(1,
5).params(2,1);pe_outs_long_rand1(1,
6).params(2,1);pe_outs_long_rand1(1, 7).params(2,1)];
pe2_rand4 = [pe_outs_long_rand4(1, 1).params(2,1);pe_outs_long_rand4(1,
2).params(2,1);pe_outs_long_rand4(1,
3).params(2,1);pe_outs_long_rand4(1,
4).params(2,1);pe_outs_long_rand4(1,
5).params(2,1);pe_outs_long_rand4(1,
6).params(2,1);pe_outs_long_rand4(1, 7).params(2,1)];
pe2_true = ones(length(pe2_rand4),1)*true_param(2);

pe2_global = [pe2_A, pe2_D, pe2_E, pe2_rand1, pe2_rand4, pe2_true];
n_exps = linspace(1,length(pe2_A),length(pe2_A));

figure(2)
plot(n_exps,pe2_global,'LineWidth',3)
xlabel('Experiment number'),ylabel('E_A_t_t_2 [J/mol]');
xlim(limits);
ylim([5e4 8e4]);
ax = gca;
ax.XTick = n_exps;
title('2nd Activation Energy vs Number of experiments');
legend 'Average variance' Determinant Eigenvalue Random_1 Random_2 'True
Parameter'

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

pe3_A =     [pe_outs_long_A(1, 1).params(3,1);pe_outs_long_A(1,
2).params(3,1);pe_outs_long_A(1, 3).params(3,1);pe_outs_long_A(1,
4).params(3,1);pe_outs_long_A(1, 5).params(3,1);pe_outs_long_A(1,
6).params(3,1);pe_outs_long_A(1, 7).params(3,1)];
pe3_D =     [pe_outs_long_D(1, 1).params(3,1);pe_outs_long_D(1,
2).params(3,1);pe_outs_long_D(1, 3).params(3,1);pe_outs_long_D(1,
4).params(3,1);pe_outs_long_D(1, 5).params(3,1);pe_outs_long_D(1,
6).params(3,1);pe_outs_long_D(1, 7).params(3,1)];
pe3_E =     [pe_outs_long_E(1, 1).params(3,1);pe_outs_long_E(1,
2).params(3,1);pe_outs_long_E(1, 3).params(3,1);pe_outs_long_E(1,
4).params(3,1);pe_outs_long_E(1, 5).params(3,1);pe_outs_long_E(1,
6).params(3,1);pe_outs_long_E(1, 7).params(3,1)];
pe3_rand1 = [pe_outs_long_rand1(1, 1).params(3,1);pe_outs_long_rand1(1,
2).params(3,1);pe_outs_long_rand1(1,
```

```
3).params(3,1);pe_outs_long_rand1(1,
4).params(3,1);pe_outs_long_rand1(1,
5).params(3,1);pe_outs_long_rand1(1,
6).params(3,1);pe_outs_long_rand1(1, 7).params(3,1)];
pe3_rand4 = [pe_outs_long_rand4(1, 1).params(3,1);pe_outs_long_rand4(1,
2).params(3,1);pe_outs_long_rand4(1,
3).params(3,1);pe_outs_long_rand4(1,
4).params(3,1);pe_outs_long_rand4(1,
5).params(3,1);pe_outs_long_rand4(1,
6).params(3,1);pe_outs_long_rand4(1, 7).params(3,1)];
pe3_true = ones(length(pe3_rand4),1)*true_param(3);

pe3_global = [pe3_A, pe3_D, pe3_E, pe3_rand1, pe3_rand4, pe3_true];
n_exps = linspace(1,length(pe3_A),length(pe3_A));

figure(3)
plot(n_exps,pe3_global,'LineWidth',3)
xlabel('Experiment number'),ylabel('E_A_t_t_3 [J/mol]');
xlim(limits);
ylim([5e4 1.5e6]);
ax = gca;
ax.XTick = n_exps;
title('3rd Activation Energy vs Number of experiments');
legend 'Average variance' Determinant Eigenvalue Random_1 Random_2 'True
Parameter'

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

pe4_A =     [pe_outs_long_A(1, 1).params(4,1);pe_outs_long_A(1,
2).params(4,1);pe_outs_long_A(1, 3).params(4,1);pe_outs_long_A(1,
4).params(4,1);pe_outs_long_A(1, 5).params(4,1);pe_outs_long_A(1,
6).params(4,1);pe_outs_long_A(1, 7).params(4,1)];
pe4_D =     [pe_outs_long_D(1, 1).params(4,1);pe_outs_long_D(1,
2).params(4,1);pe_outs_long_D(1, 3).params(4,1);pe_outs_long_D(1,
4).params(4,1);pe_outs_long_D(1, 5).params(4,1);pe_outs_long_D(1,
6).params(4,1);pe_outs_long_D(1, 7).params(4,1)];
pe4_E =     [pe_outs_long_E(1, 1).params(4,1);pe_outs_long_E(1,
2).params(4,1);pe_outs_long_E(1, 3).params(4,1);pe_outs_long_E(1,
4).params(4,1);pe_outs_long_E(1, 5).params(4,1);pe_outs_long_E(1,
6).params(4,1);pe_outs_long_E(1, 7).params(4,1)];
pe4_rand1 = [pe_outs_long_rand1(1, 1).params(4,1);pe_outs_long_rand1(1,
2).params(4,1);pe_outs_long_rand1(1,
3).params(4,1);pe_outs_long_rand1(1,
4).params(4,1);pe_outs_long_rand1(1,
5).params(4,1);pe_outs_long_rand1(1,
6).params(4,1);pe_outs_long_rand1(1, 7).params(4,1)];
pe4_rand4 = [pe_outs_long_rand4(1, 1).params(4,1);pe_outs_long_rand4(1,
2).params(4,1);pe_outs_long_rand4(1,
3).params(4,1);pe_outs_long_rand4(1,
4).params(4,1);pe_outs_long_rand4(1,
5).params(4,1);pe_outs_long_rand4(1,
6).params(4,1);pe_outs_long_rand4(1, 7).params(4,1)];
pe4_true = ones(length(pe4_rand4),1)*true_param(4);

pe4_global = [pe4_A, pe4_D, pe4_E, pe4_rand1, pe4_rand4, pe4_true];
pe4_log10_global = log10(pe4_global);
```

```
n_exps = linspace(1,length(pe4_A),length(pe4_A));

figure(4)
plot(n_exps,pe4_log10_global,'LineWidth',3)
xlabel('Experiment number'),ylabel('log_1_0(k_0_1) [J/mol]');
xlim(limits);
ax = gca;
ax.XTick = n_exps;
title('1st Frequency Factor vs Number of experiments');
legend 'Average variance' Determinant Eigenvalue Random_1 Random_2 'True
Parameter'

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

pe5_A =     [pe_outs_long_A(1, 1).params(5,1);pe_outs_long_A(1,
2).params(5,1);pe_outs_long_A(1, 3).params(5,1);pe_outs_long_A(1,
4).params(5,1);pe_outs_long_A(1, 5).params(5,1);pe_outs_long_A(1,
6).params(5,1);pe_outs_long_A(1, 7).params(5,1)];
pe5_D =     [pe_outs_long_D(1, 1).params(5,1);pe_outs_long_D(1,
2).params(5,1);pe_outs_long_D(1, 3).params(5,1);pe_outs_long_D(1,
4).params(5,1);pe_outs_long_D(1, 5).params(5,1);pe_outs_long_D(1,
6).params(5,1);pe_outs_long_D(1, 7).params(5,1)];
pe5_E =     [pe_outs_long_E(1, 1).params(5,1);pe_outs_long_E(1,
2).params(5,1);pe_outs_long_E(1, 3).params(5,1);pe_outs_long_E(1,
4).params(5,1);pe_outs_long_E(1, 5).params(5,1);pe_outs_long_E(1,
6).params(5,1);pe_outs_long_E(1, 7).params(5,1)];
pe5_rand1 = [pe_outs_long_rand1(1, 1).params(5,1);pe_outs_long_rand1(1,
2).params(5,1);pe_outs_long_rand1(1,
3).params(5,1);pe_outs_long_rand1(1,
4).params(5,1);pe_outs_long_rand1(1,
5).params(5,1);pe_outs_long_rand1(1,
6).params(5,1);pe_outs_long_rand1(1, 7).params(5,1)];
pe5_rand4 = [pe_outs_long_rand4(1, 1).params(5,1);pe_outs_long_rand4(1,
2).params(5,1);pe_outs_long_rand4(1,
3).params(5,1);pe_outs_long_rand4(1,
4).params(5,1);pe_outs_long_rand4(1,
5).params(5,1);pe_outs_long_rand4(1,
6).params(5,1);pe_outs_long_rand4(1, 7).params(5,1)];
pe5_true = ones(length(pe5_rand4),1)*true_param(5);

pe5_global = [pe5_A, pe5_D, pe5_E, pe5_rand1, pe5_rand4, pe5_true];
pe5_log10_global = log10(pe5_global);
n_exps = linspace(1,length(pe5_A),length(pe5_A));

figure(5)
plot(n_exps,pe5_log10_global,'LineWidth',3)
xlabel('Experiment number'),ylabel('log_1_0(k_0_2) [J/mol]');
xlim(limits);
ax = gca;
ax.XTick = n_exps;
title('2nd Frequency Factor vs Number of experiments');
legend 'Average variance' Determinant Eigenvalue Random_1 Random_2 'True
Parameter'

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

```
pe6_A =     [pe_outs_long_A(1, 1).params(6,1);pe_outs_long_A(1,
2).params(6,1);pe_outs_long_A(1, 3).params(6,1);pe_outs_long_A(1,
4).params(6,1);pe_outs_long_A(1, 5).params(6,1);pe_outs_long_A(1,
6).params(6,1);pe_outs_long_A(1, 7).params(6,1)];
pe6_D =     [pe_outs_long_D(1, 1).params(6,1);pe_outs_long_D(1,
2).params(6,1);pe_outs_long_D(1, 3).params(6,1);pe_outs_long_D(1,
4).params(6,1);pe_outs_long_D(1, 5).params(6,1);pe_outs_long_D(1,
6).params(6,1);pe_outs_long_D(1, 7).params(6,1)];
pe6_E =     [pe_outs_long_E(1, 1).params(6,1);pe_outs_long_E(1,
2).params(6,1);pe_outs_long_E(1, 3).params(6,1);pe_outs_long_E(1,
4).params(6,1);pe_outs_long_E(1, 5).params(6,1);pe_outs_long_E(1,
6).params(6,1);pe_outs_long_E(1, 7).params(6,1)];
pe6_rand1 = [pe_outs_long_rand1(1, 1).params(6,1);pe_outs_long_rand1(1,
2).params(6,1);pe_outs_long_rand1(1,
3).params(6,1);pe_outs_long_rand1(1,
4).params(6,1);pe_outs_long_rand1(1,
5).params(6,1);pe_outs_long_rand1(1,
6).params(6,1);pe_outs_long_rand1(1, 7).params(6,1)];
pe6_rand4 = [pe_outs_long_rand4(1, 1).params(6,1);pe_outs_long_rand4(1,
2).params(6,1);pe_outs_long_rand4(1,
3).params(6,1);pe_outs_long_rand4(1,
4).params(6,1);pe_outs_long_rand4(1,
5).params(6,1);pe_outs_long_rand4(1,
6).params(6,1);pe_outs_long_rand4(1, 7).params(6,1)];
pe6_true = ones(length(pe6_rand4),1)*true_param(6);

pe6_global = [pe6_A, pe6_D, pe6_E, pe6_rand1, pe6_rand4, pe6_true];
pe6_log10_global = log10(pe6_global);
n_exps = linspace(1,length(pe6_A),length(pe6_A));

figure(6)
plot(n_exps,pe6_log10_global,'LineWidth',3)
xlabel('Experiment number'),ylabel('log_1_0(k_0_3) [J/mol]');
xlim(limits);
ax = gca;
ax.XTick = n_exps;
title('3rd Frequency Factor vs Number of experiments');
legend 'Average variance' Determinant Eigenvalue Random_1 Random_2 'True
Parameter'

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

pe7_A = [pe_outs_long_A(1, 1).params(7,1);pe_outs_long_A(1,
2).params(7,1);pe_outs_long_A(1, 3).params(7,1);pe_outs_long_A(1,
4).params(7,1);pe_outs_long_A(1, 5).params(7,1);pe_outs_long_A(1,
6).params(7,1);pe_outs_long_A(1, 7).params(7,1)];
pe7_D = [pe_outs_long_D(1, 1).params(7,1);pe_outs_long_D(1,
2).params(7,1);pe_outs_long_D(1, 3).params(7,1);pe_outs_long_D(1,
4).params(7,1);pe_outs_long_D(1, 5).params(7,1);pe_outs_long_D(1,
6).params(7,1);pe_outs_long_D(1, 7).params(7,1)];
pe7_E = [pe_outs_long_E(1, 1).params(7,1);pe_outs_long_E(1,
2).params(7,1);pe_outs_long_E(1, 3).params(7,1);pe_outs_long_E(1,
4).params(7,1);pe_outs_long_E(1, 5).params(7,1);pe_outs_long_E(1,
6).params(7,1);pe_outs_long_E(1, 7).params(7,1)];
pe7_rand1 = [pe_outs_long_rand1(1, 1).params(7,1);pe_outs_long_rand1(1,
2).params(7,1);pe_outs_long_rand1(1,
```

```
3).params(7,1);pe_outs_long_rand1(1,
4).params(7,1);pe_outs_long_rand1(1,
5).params(7,1);pe_outs_long_rand1(1,
6).params(7,1);pe_outs_long_rand1(1, 7).params(7,1)];
pe7_rand4 = [pe_outs_long_rand4(1, 1).params(7,1);pe_outs_long_rand4(1,
2).params(7,1);pe_outs_long_rand4(1,
3).params(7,1);pe_outs_long_rand4(1,
4).params(7,1);pe_outs_long_rand4(1,
5).params(7,1);pe_outs_long_rand4(1,
6).params(7,1);pe_outs_long_rand4(1, 7).params(7,1)];
pe7_true = ones(length(pe7_rand4),1)*true_param(7);

pe7_global = [pe7_A, pe7_D, pe7_E, pe7_rand1, pe7_rand4, pe7_true];
n_exps = linspace(1,length(pe7_A),length(pe7_A));

figure(7)
plot(n_exps,pe7_global,'LineWidth',3)
xlabel('Experiment number'),ylabel('UA [W/K]');
xlim(limits);
ax = gca;
ax.XTick = n_exps;
title('Global Heat Exchange Coefficient vs Number of experiments');
legend 'Average variance' Determinant Eigenvalue Random_1 Random_2 'True
Parameter'

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

pe8_A = [pe_outs_long_A(1, 1).params(8,1);pe_outs_long_A(1,
2).params(8,1);pe_outs_long_A(1, 3).params(8,1);pe_outs_long_A(1,
4).params(8,1);pe_outs_long_A(1, 5).params(8,1);pe_outs_long_A(1,
6).params(8,1);pe_outs_long_A(1, 7).params(8,1)];
pe8_D = [pe_outs_long_D(1, 1).params(8,1);pe_outs_long_D(1,
2).params(8,1);pe_outs_long_D(1, 3).params(8,1);pe_outs_long_D(1,
4).params(8,1);pe_outs_long_D(1, 5).params(8,1);pe_outs_long_D(1,
6).params(8,1);pe_outs_long_D(1, 7).params(8,1)];
pe8_E = [pe_outs_long_E(1, 1).params(8,1);pe_outs_long_E(1,
2).params(8,1);pe_outs_long_E(1, 3).params(8,1);pe_outs_long_E(1,
4).params(8,1);pe_outs_long_E(1, 5).params(8,1);pe_outs_long_E(1,
6).params(8,1);pe_outs_long_E(1, 7).params(8,1)];
pe8_rand1 = [pe_outs_long_rand1(1, 1).params(8,1);pe_outs_long_rand1(1,
2).params(8,1);pe_outs_long_rand1(1,
3).params(8,1);pe_outs_long_rand1(1,
4).params(8,1);pe_outs_long_rand1(1,
5).params(8,1);pe_outs_long_rand1(1,
6).params(8,1);pe_outs_long_rand1(1, 7).params(8,1)];
pe8_rand4 = [pe_outs_long_rand4(1, 1).params(8,1);pe_outs_long_rand4(1,
2).params(8,1);pe_outs_long_rand4(1,
3).params(8,1);pe_outs_long_rand4(1,
4).params(8,1);pe_outs_long_rand4(1,
5).params(8,1);pe_outs_long_rand4(1,
6).params(8,1);pe_outs_long_rand4(1, 7).params(8,1)];
pe8_true = ones(length(pe8_rand4),1)*true_param(8);

pe8_global = [pe8_A, pe8_D, pe8_E, pe8_rand1, pe8_rand4, pe8_true];
n_exps = linspace(1,length(pe8_A),length(pe8_A));
```

```
figure(8)
plot(n_exps,pe8_global,'LineWidth',3)
xlabel('Experiment number'),ylabel('c_p_m_i_x [J/kg/K]');
xlim(limits);
ax = gca;
ax.XTick = n_exps;
title('Fluid Mixture Specific Heat vs Number of experiments');
legend 'Average variance' Determinant Eigenvalue Random_1 Random_2 'True
Parameter'

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

u1_A = [pe_outs_long_A(1, 7).pe_input.experiments(1,
1).controls(1,1);pe_outs_long_A(1, 7).pe_input.experiments(2,
1).controls(1,1);pe_outs_long_A(1, 7).pe_input.experiments(3,
1).controls(1,1);pe_outs_long_A(1, 7).pe_input.experiments(4,
1).controls(1,1);pe_outs_long_A(1, 7).pe_input.experiments(5,
1).controls(1,1);pe_outs_long_A(1, 7).pe_input.experiments(7,
1).controls(1,1);pe_outs_long_A(1, 7).pe_input.experiments(7,
1).controls(1,1)];
u1_D = [pe_outs_long_D(1, 7).pe_input.experiments(1,
1).controls(1,1);pe_outs_long_D(1, 7).pe_input.experiments(2,
1).controls(1,1);pe_outs_long_D(1, 7).pe_input.experiments(3,
1).controls(1,1);pe_outs_long_D(1, 7).pe_input.experiments(4,
1).controls(1,1);pe_outs_long_D(1, 7).pe_input.experiments(5,
1).controls(1,1);pe_outs_long_D(1, 7).pe_input.experiments(7,
1).controls(1,1);pe_outs_long_D(1, 7).pe_input.experiments(7,
1).controls(1,1)];
u1_E = [pe_outs_long_E(1, 7).pe_input.experiments(1,
1).controls(1,1);pe_outs_long_E(1, 7).pe_input.experiments(2,
1).controls(1,1);pe_outs_long_E(1, 7).pe_input.experiments(3,
1).controls(1,1);pe_outs_long_E(1, 7).pe_input.experiments(4,
1).controls(1,1);pe_outs_long_E(1, 7).pe_input.experiments(5,
1).controls(1,1);pe_outs_long_E(1, 7).pe_input.experiments(7,
1).controls(1,1);pe_outs_long_E(1, 7).pe_input.experiments(7,
1).controls(1,1)];
u1_rand1 = [pe_outs_long_rand1(1, 7).pe_input.experiments(1,
1).controls(1,1);pe_outs_long_rand1(1, 7).pe_input.experiments(2,
1).controls(1,1);pe_outs_long_rand1(1, 7).pe_input.experiments(3,
1).controls(1,1);pe_outs_long_rand1(1, 7).pe_input.experiments(4,
1).controls(1,1);pe_outs_long_rand1(1, 7).pe_input.experiments(5,
1).controls(1,1);pe_outs_long_rand1(1, 7).pe_input.experiments(7,
1).controls(1,1);pe_outs_long_rand1(1, 7).pe_input.experiments(7,
1).controls(1,1)];
u1_rand4 = [pe_outs_long_rand4(1, 7).pe_input.experiments(1,
1).controls(1,1);pe_outs_long_rand4(1, 7).pe_input.experiments(2,
1).controls(1,1);pe_outs_long_rand4(1, 7).pe_input.experiments(3,
1).controls(1,1);pe_outs_long_rand4(1, 7).pe_input.experiments(4,
1).controls(1,1);pe_outs_long_rand4(1, 7).pe_input.experiments(5,
1).controls(1,1);pe_outs_long_rand4(1, 7).pe_input.experiments(7,
1).controls(1,1);pe_outs_long_rand4(1, 7).pe_input.experiments(7,
1).controls(1,1)];

u1_global = [u1_A, u1_D, u1_E, u1_rand1, u1_rand4];
n_exps = linspace(1,length(u1_A),length(u1_A));
```

```
figure(9)
plot(n_exps,u1_global,'LineWidth',3)
xlabel('Experiment number'),ylabel('F [m^3/s]');
xlim(limits);
ax = gca;
ax.XTick = n_exps;
title('Flow Control vs Number of experiments');
legend 'Average variance' Determinant Eigenvalue Random_1 Random_2

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

u2_A = [pe_outs_long_A(1, 7).pe_input.experiments(1,
1).controls(2,1);pe_outs_long_A(1, 7).pe_input.experiments(2,
1).controls(2,1);pe_outs_long_A(1, 7).pe_input.experiments(3,
1).controls(2,1);pe_outs_long_A(1, 7).pe_input.experiments(4,
1).controls(2,1);pe_outs_long_A(1, 7).pe_input.experiments(5,
1).controls(2,1);pe_outs_long_A(1, 7).pe_input.experiments(7,
1).controls(2,1);pe_outs_long_A(1, 7).pe_input.experiments(7,
1).controls(2,1)];
u2_D = [pe_outs_long_D(1, 7).pe_input.experiments(1,
1).controls(2,1);pe_outs_long_D(1, 7).pe_input.experiments(2,
1).controls(2,1);pe_outs_long_D(1, 7).pe_input.experiments(3,
1).controls(2,1);pe_outs_long_D(1, 7).pe_input.experiments(4,
1).controls(2,1);pe_outs_long_D(1, 7).pe_input.experiments(5,
1).controls(2,1);pe_outs_long_D(1, 7).pe_input.experiments(7,
1).controls(2,1);pe_outs_long_D(1, 7).pe_input.experiments(7,
1).controls(2,1)];
u2_E = [pe_outs_long_E(1, 7).pe_input.experiments(1,
1).controls(2,1);pe_outs_long_E(1, 7).pe_input.experiments(2,
1).controls(2,1);pe_outs_long_E(1, 7).pe_input.experiments(3,
1).controls(2,1);pe_outs_long_E(1, 7).pe_input.experiments(4,
1).controls(2,1);pe_outs_long_E(1, 7).pe_input.experiments(5,
1).controls(2,1);pe_outs_long_E(1, 7).pe_input.experiments(7,
1).controls(2,1);pe_outs_long_E(1, 7).pe_input.experiments(7,
1).controls(2,1)];
u2_rand1 = [pe_outs_long_rand1(1, 7).pe_input.experiments(1,
1).controls(2,1);pe_outs_long_rand1(1, 7).pe_input.experiments(2,
1).controls(2,1);pe_outs_long_rand1(1, 7).pe_input.experiments(3,
1).controls(2,1);pe_outs_long_rand1(1, 7).pe_input.experiments(4,
1).controls(2,1);pe_outs_long_rand1(1, 7).pe_input.experiments(5,
1).controls(2,1);pe_outs_long_rand1(1, 7).pe_input.experiments(7,
1).controls(2,1);pe_outs_long_rand1(1, 7).pe_input.experiments(7,
1).controls(2,1)];
u2_rand4 = [pe_outs_long_rand4(1, 7).pe_input.experiments(1,
1).controls(2,1);pe_outs_long_rand4(1, 7).pe_input.experiments(2,
1).controls(2,1);pe_outs_long_rand4(1, 7).pe_input.experiments(3,
1).controls(2,1);pe_outs_long_rand4(1, 7).pe_input.experiments(4,
1).controls(2,1);pe_outs_long_rand4(1, 7).pe_input.experiments(5,
1).controls(2,1);pe_outs_long_rand4(1, 7).pe_input.experiments(7,
1).controls(2,1);pe_outs_long_rand4(1, 7).pe_input.experiments(7,
1).controls(2,1)];

u2_global = [u2_A, u2_D, u2_E, u2_rand1, u2_rand4];
n_exps = linspace(1,length(u2_A),length(u2_A));

figure(10)
```

```
plot(n_exps,u2_global,'LineWidth',3)
xlabel('Experiment number'),ylabel('C_i_n_A [mol/m^3]');
xlim(limits);
ax = gca;
ax.XTick = n_exps;
title('Inlet A Concentration vs Number of experiments');
legend 'Average variance' Determinant Eigenvalue Random_1 Random_2

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

u3_A = [pe_outs_long_A(1, 7).pe_input.experiments(1,
1).controls(3,1);pe_outs_long_A(1, 7).pe_input.experiments(2,
1).controls(3,1);pe_outs_long_A(1, 7).pe_input.experiments(3,
1).controls(3,1);pe_outs_long_A(1, 7).pe_input.experiments(4,
1).controls(3,1);pe_outs_long_A(1, 7).pe_input.experiments(5,
1).controls(3,1);pe_outs_long_A(1, 7).pe_input.experiments(7,
1).controls(3,1);pe_outs_long_A(1, 7).pe_input.experiments(7,
1).controls(3,1)];
u3_D = [pe_outs_long_D(1, 7).pe_input.experiments(1,
1).controls(3,1);pe_outs_long_D(1, 7).pe_input.experiments(2,
1).controls(3,1);pe_outs_long_D(1, 7).pe_input.experiments(3,
1).controls(3,1);pe_outs_long_D(1, 7).pe_input.experiments(4,
1).controls(3,1);pe_outs_long_D(1, 7).pe_input.experiments(5,
1).controls(3,1);pe_outs_long_D(1, 7).pe_input.experiments(7,
1).controls(3,1);pe_outs_long_D(1, 7).pe_input.experiments(7,
1).controls(3,1)];
u3_E = [pe_outs_long_E(1, 7).pe_input.experiments(1,
1).controls(3,1);pe_outs_long_E(1, 7).pe_input.experiments(2,
1).controls(3,1);pe_outs_long_E(1, 7).pe_input.experiments(3,
1).controls(3,1);pe_outs_long_E(1, 7).pe_input.experiments(4,
1).controls(3,1);pe_outs_long_E(1, 7).pe_input.experiments(5,
1).controls(3,1);pe_outs_long_E(1, 7).pe_input.experiments(7,
1).controls(3,1);pe_outs_long_E(1, 7).pe_input.experiments(7,
1).controls(3,1)];
u3_rand1 = [pe_outs_long_rand1(1, 7).pe_input.experiments(1,
1).controls(3,1);pe_outs_long_rand1(1, 7).pe_input.experiments(2,
1).controls(3,1);pe_outs_long_rand1(1, 7).pe_input.experiments(3,
1).controls(3,1);pe_outs_long_rand1(1, 7).pe_input.experiments(4,
1).controls(3,1);pe_outs_long_rand1(1, 7).pe_input.experiments(5,
1).controls(3,1);pe_outs_long_rand1(1, 7).pe_input.experiments(7,
1).controls(3,1);pe_outs_long_rand1(1, 7).pe_input.experiments(7,
1).controls(3,1)];
u3_rand4 = [pe_outs_long_rand4(1, 7).pe_input.experiments(1,
1).controls(3,1);pe_outs_long_rand4(1, 7).pe_input.experiments(2,
1).controls(3,1);pe_outs_long_rand4(1, 7).pe_input.experiments(3,
1).controls(3,1);pe_outs_long_rand4(1, 7).pe_input.experiments(4,
1).controls(3,1);pe_outs_long_rand4(1, 7).pe_input.experiments(5,
1).controls(3,1);pe_outs_long_rand4(1, 7).pe_input.experiments(7,
1).controls(3,1);pe_outs_long_rand4(1, 7).pe_input.experiments(7,
1).controls(3,1)];

u3_global = [u3_A, u3_D, u3_E, u3_rand1, u3_rand4];
n_exps = linspace(1,length(u3_A),length(u3_A));

figure(11)
plot(n_exps,u3_global,'LineWidth',3)
```

```
xlabel('Experiment number'),ylabel('C_i_n_B [mol/m^3]');
xlim(limits);
ax = gca;
ax.XTick = n_exps;
title('Inlet B Concentration vs Number of experiments');
legend 'Average variance' Determinant Eigenvalue Random_1 Random_2

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

u4_A = [pe_outs_long_A(1, 7).pe_input.experiments(1,
1).controls(4,1);pe_outs_long_A(1, 7).pe_input.experiments(2,
1).controls(4,1);pe_outs_long_A(1, 7).pe_input.experiments(3,
1).controls(4,1);pe_outs_long_A(1, 7).pe_input.experiments(4,
1).controls(4,1);pe_outs_long_A(1, 7).pe_input.experiments(5,
1).controls(4,1);pe_outs_long_A(1, 7).pe_input.experiments(7,
1).controls(4,1);pe_outs_long_A(1, 7).pe_input.experiments(7,
1).controls(4,1)];
u4_D = [pe_outs_long_D(1, 7).pe_input.experiments(1,
1).controls(4,1);pe_outs_long_D(1, 7).pe_input.experiments(2,
1).controls(4,1);pe_outs_long_D(1, 7).pe_input.experiments(3,
1).controls(4,1);pe_outs_long_D(1, 7).pe_input.experiments(4,
1).controls(4,1);pe_outs_long_D(1, 7).pe_input.experiments(5,
1).controls(4,1);pe_outs_long_D(1, 7).pe_input.experiments(7,
1).controls(4,1);pe_outs_long_D(1, 7).pe_input.experiments(7,
1).controls(4,1)];
u4_E = [pe_outs_long_E(1, 7).pe_input.experiments(1,
1).controls(4,1);pe_outs_long_E(1, 7).pe_input.experiments(2,
1).controls(4,1);pe_outs_long_E(1, 7).pe_input.experiments(3,
1).controls(4,1);pe_outs_long_E(1, 7).pe_input.experiments(4,
1).controls(4,1);pe_outs_long_E(1, 7).pe_input.experiments(5,
1).controls(4,1);pe_outs_long_E(1, 7).pe_input.experiments(7,
1).controls(4,1);pe_outs_long_E(1, 7).pe_input.experiments(7,
1).controls(4,1)];
u4_rand1 = [pe_outs_long_rand1(1, 7).pe_input.experiments(1,
1).controls(4,1);pe_outs_long_rand1(1, 7).pe_input.experiments(2,
1).controls(4,1);pe_outs_long_rand1(1, 7).pe_input.experiments(3,
1).controls(4,1);pe_outs_long_rand1(1, 7).pe_input.experiments(4,
1).controls(4,1);pe_outs_long_rand1(1, 7).pe_input.experiments(5,
1).controls(4,1);pe_outs_long_rand1(1, 7).pe_input.experiments(7,
1).controls(4,1);pe_outs_long_rand1(1, 7).pe_input.experiments(7,
1).controls(4,1)];
u4_rand4 = [pe_outs_long_rand4(1, 7).pe_input.experiments(1,
1).controls(4,1);pe_outs_long_rand4(1, 7).pe_input.experiments(2,
1).controls(4,1);pe_outs_long_rand4(1, 7).pe_input.experiments(3,
1).controls(4,1);pe_outs_long_rand4(1, 7).pe_input.experiments(4,
1).controls(4,1);pe_outs_long_rand4(1, 7).pe_input.experiments(5,
1).controls(4,1);pe_outs_long_rand4(1, 7).pe_input.experiments(7,
1).controls(4,1);pe_outs_long_rand4(1, 7).pe_input.experiments(7,
1).controls(4,1)];

u4_global = [u4_A, u4_D, u4_E, u4_rand1, u4_rand4];
n_exps = linspace(1,length(u4_A),length(u4_A));

figure(12)
plot(n_exps,u4_global,'LineWidth',3)
xlabel('Experiment number'),ylabel('C_i_n_C [mol/m^3]');
```

```
xlim(limits);
ax = gca;
ax.XTick = n_exps;
title('Inlet C Concentration vs Number of experiments');
legend 'Average variance' Determinant Eigenvalue Random_1 Random_2

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

u5_A = [pe_outs_long_A(1, 7).pe_input.experiments(1,
1).controls(5,1);pe_outs_long_A(1, 7).pe_input.experiments(2,
1).controls(5,1);pe_outs_long_A(1, 7).pe_input.experiments(3,
1).controls(5,1);pe_outs_long_A(1, 7).pe_input.experiments(4,
1).controls(5,1);pe_outs_long_A(1, 7).pe_input.experiments(5,
1).controls(5,1);pe_outs_long_A(1, 7).pe_input.experiments(7,
1).controls(5,1);pe_outs_long_A(1, 7).pe_input.experiments(7,
1).controls(5,1)];
u5_D = [pe_outs_long_D(1, 7).pe_input.experiments(1,
1).controls(5,1);pe_outs_long_D(1, 7).pe_input.experiments(2,
1).controls(5,1);pe_outs_long_D(1, 7).pe_input.experiments(3,
1).controls(5,1);pe_outs_long_D(1, 7).pe_input.experiments(4,
1).controls(5,1);pe_outs_long_D(1, 7).pe_input.experiments(5,
1).controls(5,1);pe_outs_long_D(1, 7).pe_input.experiments(7,
1).controls(5,1);pe_outs_long_D(1, 7).pe_input.experiments(7,
1).controls(5,1)];
u5_E = [pe_outs_long_E(1, 7).pe_input.experiments(1,
1).controls(5,1);pe_outs_long_E(1, 7).pe_input.experiments(2,
1).controls(5,1);pe_outs_long_E(1, 7).pe_input.experiments(3,
1).controls(5,1);pe_outs_long_E(1, 7).pe_input.experiments(4,
1).controls(5,1);pe_outs_long_E(1, 7).pe_input.experiments(5,
1).controls(5,1);pe_outs_long_E(1, 7).pe_input.experiments(7,
1).controls(5,1);pe_outs_long_E(1, 7).pe_input.experiments(7,
1).controls(5,1)];
u5_rand1 = [pe_outs_long_rand1(1, 7).pe_input.experiments(1,
1).controls(5,1);pe_outs_long_rand1(1, 7).pe_input.experiments(2,
1).controls(5,1);pe_outs_long_rand1(1, 7).pe_input.experiments(3,
1).controls(5,1);pe_outs_long_rand1(1, 7).pe_input.experiments(4,
1).controls(5,1);pe_outs_long_rand1(1, 7).pe_input.experiments(5,
1).controls(5,1);pe_outs_long_rand1(1, 7).pe_input.experiments(7,
1).controls(5,1);pe_outs_long_rand1(1, 7).pe_input.experiments(7,
1).controls(5,1)];
u5_rand4 = [pe_outs_long_rand4(1, 7).pe_input.experiments(1,
1).controls(5,1);pe_outs_long_rand4(1, 7).pe_input.experiments(2,
1).controls(5,1);pe_outs_long_rand4(1, 7).pe_input.experiments(3,
1).controls(5,1);pe_outs_long_rand4(1, 7).pe_input.experiments(4,
1).controls(5,1);pe_outs_long_rand4(1, 7).pe_input.experiments(5,
1).controls(5,1);pe_outs_long_rand4(1, 7).pe_input.experiments(7,
1).controls(5,1);pe_outs_long_rand4(1, 7).pe_input.experiments(7,
1).controls(5,1)];

u5_global = [u5_A, u5_D, u5_E, u5_rand1, u5_rand4];
n_exps = linspace(1,length(u5_A),length(u5_A));

figure(13)
plot(n_exps,u5_global,'LineWidth',3)
xlabel('Experiment number'),ylabel('C_i_n_D [mol/m^3]');
xlim(limits);
```

```
ax = gca;
ax.XTick = n_exps;
title('Inlet D Concentration vs Number of experiments');
legend 'Average variance' Determinant Eigenvalue Random_1 Random_2

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

u6_A = [pe_outs_long_A(1, 7).pe_input.experiments(1,
1).controls(6,1);pe_outs_long_A(1, 7).pe_input.experiments(2,
1).controls(6,1);pe_outs_long_A(1, 7).pe_input.experiments(3,
1).controls(6,1);pe_outs_long_A(1, 7).pe_input.experiments(4,
1).controls(6,1);pe_outs_long_A(1, 7).pe_input.experiments(5,
1).controls(6,1);pe_outs_long_A(1, 7).pe_input.experiments(7,
1).controls(6,1);pe_outs_long_A(1, 7).pe_input.experiments(7,
1).controls(6,1)];
u6_D = [pe_outs_long_D(1, 7).pe_input.experiments(1,
1).controls(6,1);pe_outs_long_D(1, 7).pe_input.experiments(2,
1).controls(6,1);pe_outs_long_D(1, 7).pe_input.experiments(3,
1).controls(6,1);pe_outs_long_D(1, 7).pe_input.experiments(4,
1).controls(6,1);pe_outs_long_D(1, 7).pe_input.experiments(5,
1).controls(6,1);pe_outs_long_D(1, 7).pe_input.experiments(7,
1).controls(6,1);pe_outs_long_D(1, 7).pe_input.experiments(7,
1).controls(6,1)];
u6_E = [pe_outs_long_E(1, 7).pe_input.experiments(1,
1).controls(6,1);pe_outs_long_E(1, 7).pe_input.experiments(2,
1).controls(6,1);pe_outs_long_E(1, 7).pe_input.experiments(3,
1).controls(6,1);pe_outs_long_E(1, 7).pe_input.experiments(4,
1).controls(6,1);pe_outs_long_E(1, 7).pe_input.experiments(5,
1).controls(6,1);pe_outs_long_E(1, 7).pe_input.experiments(7,
1).controls(6,1);pe_outs_long_E(1, 7).pe_input.experiments(7,
1).controls(6,1)];
u6_rand1 = [pe_outs_long_rand1(1, 7).pe_input.experiments(1,
1).controls(6,1);pe_outs_long_rand1(1, 7).pe_input.experiments(2,
1).controls(6,1);pe_outs_long_rand1(1, 7).pe_input.experiments(3,
1).controls(6,1);pe_outs_long_rand1(1, 7).pe_input.experiments(4,
1).controls(6,1);pe_outs_long_rand1(1, 7).pe_input.experiments(5,
1).controls(6,1);pe_outs_long_rand1(1, 7).pe_input.experiments(7,
1).controls(6,1);pe_outs_long_rand1(1, 7).pe_input.experiments(7,
1).controls(6,1)];
u6_rand4 = [pe_outs_long_rand4(1, 7).pe_input.experiments(1,
1).controls(6,1);pe_outs_long_rand4(1, 7).pe_input.experiments(2,
1).controls(6,1);pe_outs_long_rand4(1, 7).pe_input.experiments(3,
1).controls(6,1);pe_outs_long_rand4(1, 7).pe_input.experiments(4,
1).controls(6,1);pe_outs_long_rand4(1, 7).pe_input.experiments(5,
1).controls(6,1);pe_outs_long_rand4(1, 7).pe_input.experiments(7,
1).controls(6,1);pe_outs_long_rand4(1, 7).pe_input.experiments(7,
1).controls(6,1)];
u6_global = [u6_A, u6_D, u6_E, u6_rand1, u6_rand4];
n_exps = linspace(1,length(u6_A),length(u6_A));

figure(14)
plot(n_exps,u6_global,'LineWidth',3)
xlabel('Experiment number'),ylabel('T_i_n [K]');
xlim(limits);
ax = gca;
ax.XTick = n_exps;
title('Inlet Temperature vs Number of experiments');
```

```
legend 'Average variance' Determinant Eigenvalue Random_1 Random_2

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

u7_A = [pe_outs_long_A(1, 7).pe_input.experiments(1,
1).controls(7,1);pe_outs_long_A(1, 7).pe_input.experiments(2,
1).controls(7,1);pe_outs_long_A(1, 7).pe_input.experiments(3,
1).controls(7,1);pe_outs_long_A(1, 7).pe_input.experiments(4,
1).controls(7,1);pe_outs_long_A(1, 7).pe_input.experiments(5,
1).controls(7,1);pe_outs_long_A(1, 7).pe_input.experiments(7,
1).controls(7,1);pe_outs_long_A(1, 7).pe_input.experiments(7,
1).controls(7,1)];
u7_D = [pe_outs_long_D(1, 7).pe_input.experiments(1,
1).controls(7,1);pe_outs_long_D(1, 7).pe_input.experiments(2,
1).controls(7,1);pe_outs_long_D(1, 7).pe_input.experiments(3,
1).controls(7,1);pe_outs_long_D(1, 7).pe_input.experiments(4,
1).controls(7,1);pe_outs_long_D(1, 7).pe_input.experiments(5,
1).controls(7,1);pe_outs_long_D(1, 7).pe_input.experiments(7,
1).controls(7,1);pe_outs_long_D(1, 7).pe_input.experiments(7,
1).controls(7,1)];
u7_E = [pe_outs_long_E(1, 7).pe_input.experiments(1,
1).controls(7,1);pe_outs_long_E(1, 7).pe_input.experiments(2,
1).controls(7,1);pe_outs_long_E(1, 7).pe_input.experiments(3,
1).controls(7,1);pe_outs_long_E(1, 7).pe_input.experiments(4,
1).controls(7,1);pe_outs_long_E(1, 7).pe_input.experiments(5,
1).controls(7,1);pe_outs_long_E(1, 7).pe_input.experiments(7,
1).controls(7,1);pe_outs_long_E(1, 7).pe_input.experiments(7,
1).controls(7,1)];
u7_rand1 = [pe_outs_long_rand1(1, 7).pe_input.experiments(1,
1).controls(7,1);pe_outs_long_rand1(1, 7).pe_input.experiments(2,
1).controls(7,1);pe_outs_long_rand1(1, 7).pe_input.experiments(3,
1).controls(7,1);pe_outs_long_rand1(1, 7).pe_input.experiments(4,
1).controls(7,1);pe_outs_long_rand1(1, 7).pe_input.experiments(5,
1).controls(7,1);pe_outs_long_rand1(1, 7).pe_input.experiments(7,
1).controls(7,1);pe_outs_long_rand1(1, 7).pe_input.experiments(7,
1).controls(7,1)];
u7_rand4 = [pe_outs_long_rand4(1, 7).pe_input.experiments(1,
1).controls(7,1);pe_outs_long_rand4(1, 7).pe_input.experiments(2,
1).controls(7,1);pe_outs_long_rand4(1, 7).pe_input.experiments(3,
1).controls(7,1);pe_outs_long_rand4(1, 7).pe_input.experiments(4,
1).controls(7,1);pe_outs_long_rand4(1, 7).pe_input.experiments(5,
1).controls(7,1);pe_outs_long_rand4(1, 7).pe_input.experiments(7,
1).controls(7,1);pe_outs_long_rand4(1, 7).pe_input.experiments(7,
1).controls(7,1)];
u7_global = [u7_A, u7_D, u7_E, u7_rand1, u7_rand4];
n_exps = linspace(1,length(u7_A),length(u7_A));

figure(15)
plot(n_exps,u7_global,'LineWidth',3)
xlabel('Experiment number'),ylabel('T_j [K]');
xlim(limits);
ax = gca;
ax.XTick = n_exps;
title('Jacket Temperature vs Number of experiments');
legend 'Average variance' Determinant Eigenvalue Random_1 Random_2
```

A.4 Validation results: Rank and LSQ function run_LSQ_Rank_calc.m

```
clear

path_start = 'C:\Users\Sergio\Documents\THESIS\Sergio_Loop_new\out';

load(strcat(path_start,'/50Aout_2019_07_26_23_09_27.mat'));

format shorteng

current_file_path = fileparts(mfilename('fullpath'));
add_paths();

num_hammersley_global = 1000;
u_sets  = set_hammersley('OED',num_hammersley_global);
p_sets = zeros(length(true_param),length(pe_outs_long));

ys_model_precise =
zeros(length(pe_outs_long),length(u_sets(1,:)),length(time_vector),lengt
h(y_init));
ys_RP_meas =
zeros(1,length(u_sets(1,:)),length(time_vector),length(y_init));
ys_RP_precise =
zeros(1,length(u_sets(1,:)),length(time_vector),length(y_init));

exps_model = experiment.empty;
exps_RP = experiment.empty;

Rank_model = zeros(1,length(pe_outs_long));
Act_p_model = zeros(length(true_param),length(pe_outs_long));
IdentifOrd_model = zeros(length(true_param),length(pe_outs_long));

disp('Starting comparison');

for i = 1:length(pe_outs_long)

    pe_out_long = pe_outs_long(i);
    p_sets(:,i) = pe_out_long.params;

end

for i = 1:length(pe_outs_long)

    disp('-----------------------------------------------------');
   disp(['Starting exp data generation with current p_set
#',num2str(i)]);

    for exp = 1:length(u_sets(1,:))

            [y_init, ~, time_vector, sigma, ~, ~ ] = set_initials();
            sim_in_model = simulation_input(time_vector, y_init,
p_sets(:,i), u_sets(:,exp), sigma, 'big');

        sim_out_model = do_plant_simulation(sim_in_model);
        ys_model_precise(i,exp,:,:) = sim_out_model.y_precise;
    end
```

```
   disp('-------------------------------------------------');
   disp(['Saved all experiments with current p_set #',num2str(i)]);

    exp_model = sim_out_model.experiment;
    exps_model = [exps_model, exp_model];

    ss_in_model = ss_input(p_sets(:,i), exps_model);
    ss_out_model = do_subset_selection(ss_in_model);
    Rank_model(i) = ss_out_model.subset_info.Rank;
    Act_p_model(:,i) = ss_out_model.active;
    IdentifOrd_model(:,i) = ss_out_model.subset_info.IdentifOrd;

end

for exp = 1:length(u_sets(1,:))

            [y_init, ~, time_vector, sigma, ~, ~ ] = set_initials();
            sim_in_RP = simulation_input(time_vector, y_init,
true_param, u_sets(:,exp), sigma, 'big');
            sim_in_RP_nonoise = simulation_input(time_vector, y_init,
true_param, u_sets(:,exp), ones(length(sigma),1), 'big');

    sim_out_RP = do_plant_simulation(sim_in_RP);

    y_RP_precise = sim_out_RP.y_precise;
    y_RP_meas = sim_out_RP.experiment.y_measured;

    ys_RP_precise(1,exp,:,:) = y_RP_precise(:,:);
    ys_RP_meas(1,exp,:,:) = y_RP_meas(:,:);

    sim_out_RP_nonoise = do_plant_simulation(sim_in_RP_nonoise);

end

disp('Starting SSS for Real Plant');

exp_RP_nonoise = sim_out_RP_nonoise.experiment;

ss_in_RP_nonoise = ss_input(true_param, exp_RP_nonoise);
ss_out_RP_nonoise = do_subset_selection(ss_in_RP_nonoise);
Rank_RP_nonoise = ss_out_RP_nonoise.subset_info.Rank;
Act_p_RP_nonoise = ss_out_RP_nonoise.active;
IdentifOrd_RP_nonoise = ss_out_RP_nonoise.subset_info.IdentifOrd;

LSQcurve = zeros(1,length(pe_outs_long));

for i = 1:length(pe_outs_long)

    res_curve = ys_model_precise(i,:,:,:) - ys_RP_meas(1,:,:,:);
    res_curve2 = res_curve.^2;
    res_curve2resh =
reshape(res_curve2,[length(u_sets(1,:))*length(time_vector)*length(y_ini
t),1]);
    LSQcurve(1,i) = sum(res_curve2resh);

end
```

```
n_exps = 1:length(LSQcurve);

res_asym = ys_RP_precise(1,:,:,:) - ys_RP_meas(1,:,:,:);
res_asym2 = res_asym.^2;
res_asym2resh =
reshape(res_asym2,[length(u_sets(1,:))*length(time_vector)*length(y_init
),1]);

LSQasymptote = sum(res_asym2resh);
LSQasymptote = LSQasymptote*ones(size(n_exps));

err_percent = (LSQcurve - LSQasymptote) ./ LSQasymptote * 100;

Rank_RP = Rank_RP_nonoise*ones(size(n_exps));

disp('############################################################');
disp(['Reactor type ',reactor_type]);
disp('Save by hand by reading instructions on script end');

% Save to results.mat file using save('results.mat','varName','-append')
%                             LSQ with criterion specification
%                             LSQasymptote (check it's always the same)
%                             n_exps
%                             err% with criterion specification
%                             Rank with criterion specification
%                             Rank_RP (check it's always the same)

save('C:\Users\Sergio\Documents\THESIS\Sergio_Loop_new\validation_result
s\50results.mat','LSQcurve','-append')
save('C:\Users\Sergio\Documents\THESIS\Sergio_Loop_new\validation_result
s\50results.mat','LSQasymptote','-append')
save('C:\Users\Sergio\Documents\THESIS\Sergio_Loop_new\validation_result
s\50results.mat','n_exps','-append')
save('C:\Users\Sergio\Documents\THESIS\Sergio_Loop_new\validation_result
s\50results.mat','err_percent','-append')
save('C:\Users\Sergio\Documents\THESIS\Sergio_Loop_new\validation_result
s\50results.mat','Rank_model','-append')
save('C:\Users\Sergio\Documents\THESIS\Sergio_Loop_new\validation_result
s\50results.mat','Rank_RP','-append')
save('C:\Users\Sergio\Documents\THESIS\Sergio_Loop_new\validation_result
s\50results.mat','Rank_RP_nonoise','-append')
save('C:\Users\Sergio\Documents\THESIS\Sergio_Loop_new\validation_result
s\50results.mat','Act_p_model','-append')
save('C:\Users\Sergio\Documents\THESIS\Sergio_Loop_new\validation_result
s\50results.mat','Act_p_RP_nonoise','-append')
save('C:\Users\Sergio\Documents\THESIS\Sergio_Loop_new\validation_result
s\50results.mat','Elapsed_times','-append')
save('C:\Users\Sergio\Documents\THESIS\Sergio_Loop_new\validation_result
s\50results.mat','IdentifOrd_model','-append')
save('C:\Users\Sergio\Documents\THESIS\Sergio_Loop_new\validation_result
s\50results.mat','IdentifOrd_RP_nonoise','-append')

disp('############################################################');
disp(['Reactor type ',reactor_type]);
disp(['Chosen criterion ',ChooseCriterion])
% Criterion specification are: A D E Rand
```

A.5 Validation results printer aux_print_LSQ_Rank_new

```
close all
clear
clc

load('C:\Users\Sergio\Documents\THESIS\Sergio_Loop_new\validation_results\50results.mat')

limits = [1 7];

Abs_times_A = cumsum(Elapsed_times_A);
Abs_times_D = cumsum(Elapsed_times_D);
Abs_times_E = cumsum(Elapsed_times_E);
Abs_times_rand1 = cumsum(Elapsed_times_rand1);
Abs_times_rand4 = cumsum(Elapsed_times_rand4);

Abs_times_global =
[Abs_times_A;Abs_times_D;Abs_times_E;Abs_times_rand1;Abs_times_rand4];
Act_p_model_global =
[Act_p_model_A;Act_p_model_D;Act_p_model_E;Act_p_model_rand1;Act_p_model_rand4];
Elapsed_times_global =
[Elapsed_times_A;Elapsed_times_D;Elapsed_times_E;Elapsed_times_rand1;Elapsed_times_rand4];
err_percent_global =
[err_percent_A;err_percent_D;err_percent_E;err_percent_rand1;err_percent_rand4];
LSQasymptote_global =
[LSQasymptote_A;LSQasymptote_D;LSQasymptote_E;LSQasymptote_rand1;LSQasymptote_rand4];
LSQcurve_global =
[LSQcurve_A;LSQcurve_D;LSQcurve_E;LSQcurve_rand1;LSQcurve_rand4];
Rank_model_global =
[Rank_model_A;Rank_model_D;Rank_model_E;Rank_model_rand1;Rank_model_rand4];

figure(1),plot(n_exps_E,LSQcurve_global',n_exps_E,LSQasymptote_rand3,'LineWidth',3);
xlabel('Experiment number'),ylabel('LSQ');
xlim(limits);
ylim([0 3e6]);
ax = gca;
ax.XTick = n_exps_E;
title('LSQ function vs Number of experiments');legend 'Average variance' Determinant Eigenvalue Random_1 Random_2 'Model Mismatch'

logLSQcurve_global = log10(LSQcurve_global);
logLSQasymptote_rand3 = log10(LSQasymptote_rand3);

figure(2)
plot(n_exps_E,logLSQcurve_global',n_exps_E,logLSQasymptote_rand3,'LineWidth',3);
xlabel('Experiment number'),ylabel('log_1_0(LSQ)');
xlim(limits);
ylim([0 5.5])
```

```
ax = gca;
ax.XTick = n_exps_E;
title('LSQ function vs Number of experiments');legend 'Average variance'
Determinant Eigenvalue Random_1 Random_2 'Model Mismatch'

figure(3)
plot(n_exps_E,err_percent_global','LineWidth',3);
xlabel('Experiment number'),ylabel('Error_%');
xlim(limits);
ax = gca;
ax.XTick = n_exps_E;
title('LSQ Error vs Number of experiments');legend A D E Random_1
Random_2

figure(4)
plot(n_exps_E,Rank_model_global(1,:),'-
.',n_exps_E,Rank_model_global(2,:)',':',n_exps_E,Rank_model_global(3,:)'
,'--
',n_exps_E,Rank_model_global(4,:),'*',n_exps_E,Rank_model_global(5,:)','
o',n_exps_E,Rank_RP_E,'LineWidth',3);
xlabel('Experiment number'),ylabel('Rank');
xlim(limits);
ylim([0 5]);
legend 'Average variance' Determinant Eigenvalue Random_1 Random_2 'Real
Plant'
xticks(n_exps_E);
yticks([0 1 2 3 4 5]);
title('Rank of Sensitivity Matrix vs Number of experiments');

figure(5)
plot(n_exps_E,Elapsed_times_global','LineWidth',3);
xlabel('Experiment number'),ylabel('Elapsed time [s]');
xlim(limits);
ax = gca;
ax.XTick = n_exps_E;
title('Elapsed time for each exp vs Number of experiments');legend
'Average variance' Determinant Eigenvalue Random_1 Random_2

figure(6)
pl = plot(n_exps_E,Abs_times_global','LineWidth',3);
xlabel('Experiment number'),ylabel('Absolute time [s]');
xlim(limits);
ax = gca;
ax.XTick = n_exps_E;
title('Absolute time vs Number of experiments');legend 'Average
variance' Determinant Eigenvalue Random_1 Random_2
```

Bibliography

Abt, V. *et al.* (2018) 'Model-based tools for optimal experiments in bioprocess engineering', *Current Opinion in Chemical Engineering*, pp. 244–252. doi: 10.1016/j.coche.2018.11.007.

Annergren, M. *et al.* (2017) 'Application-Oriented Input Design in System Identification: Optimal Input Design for Control [Applications of Control]', *IEEE Control Systems*, 37(2), pp. 31–56. doi: 10.1109/MCS.2016.2643243.

Bardow, A. (2008) 'Optimal experimental design of ill-posed problems: The METER approach', *Computers and Chemical Engineering*, 32(1–2), pp. 115–124. doi: 10.1016/j.compchemeng.2007.05.004.

Barz, T. *et al.* (2013) 'Experimental evaluation of an approach to online redesign of experiments for parameter determination', *AIChE Journal*, 59(6), pp. 1981–1995. doi: 10.1002/aic.13957.

Barz, T. *et al.* (2016) 'Real-time adaptive input design for the determination of competitive adsorption isotherms in liquid chromatography', *Computers and Chemical Engineering*, 94, pp. 104–116. doi: 10.1016/j.compchemeng.2016.07.009.

Bitterlich, S. and Knabner, P. (2003) 'Experimental design for outflow experiments based on a multi-level identification method for material laws', *Inverse Problems*, 19(5), pp. 1011–1030. doi: 10.1088/0266-5611/19/5/302.

Brun, R. *et al.* (2002) 'Practical identifiability of ASM2d parameters - Systematic selection and tuning of parameter subsets', *Water Research*. doi: 10.1016/S0043-1354(02)00104-5.

Carolina López Cárdenas, D. *et al.* (2016) 'Systematic evaluation of ill-posed problems in model-based parameter estimation and experimental design', p. 244.

Chen, J. and Wang, K. (2004) 'Using Hybrid Function Approximations', pp. 5260–5274.

David, F. N. and Cramer, H. (1947) 'Mathematical Methods of Statistics.', *Biometrika*. doi: 10.2307/2332454.

Fisher Box, J. (1980) 'R.A. fisher and the design of experiments, 1922-1926', *American Statistician*, 34(1), pp. 1–7. doi: 10.1080/00031305.1980.10482701.

Fisher, R. A. (1935) 'The Design of Experiments'.

Fohring, J. (2016) 'Adaptive optimal experimental design and inversion of a coupled uid ow and geophysical imaging model for reservoir monitoring', p. 98.

Franceschini, G. and Macchietto, S. (2008) 'Model-based design of experiments for parameter precision: State of the art', *Chemical Engineering Science*, 63(19), pp. 4846–4872. doi: 10.1016/j.ces.2007.11.034.

Grah, A. (2004) 'Entwicklung und Anwendung modularer Software zur Simulation und Parametersch ¨ atzung in gaskatalytischen Festbettreaktoren Dissertation Vorwort'.

Gu, H. (2016) 'A Robust Adaptive Autonomous Approach to Optimal Experimental Design', (June).

Haber, E., Horesh, L. and Tenorio, L. (2008) 'Numerical methods for experimental design of large-scale linear ill-posed inverse problems', *Inverse Problems*, 24(5). doi: 10.1088/0266-5611/24/5/055012.

Haber, E., Horesh, L. and Tenorio, L. (2010) 'Numerical methods for the design of large-scale nonlinear discrete ill-posed inverse problems', *Inverse Problems*, 26(2). doi: 10.1088/0266-5611/26/2/025002.

Huang, L., Hjalmarsson, H. and Gerencser, L. (2012) 'Adaptive experiment design for ARMAX systems?', *Proceedings of the IEEE Conference on Decision and Control*, pp. 907–912. doi: 10.1109/CDC.2012.6425920.

Ljung, L. (1998) *System identification: Theory for the user*, *Automatica*. doi: 10.1016/0005-1098(89)90019-8.

López C., D. C. *et al.* (2015) 'Nonlinear ill-posed problem analysis in model-based parameter estimation and experimental design', *Computers and Chemical Engineering*. Elsevier Ltd, 77, pp. 24–42. doi: 10.1016/j.compchemeng.2015.03.002.

Müller, D. *et al.* (2014) 'Systematic Parameter Selection for Optimization under Uncertainty', *Computer Aided Chemical Engineering*. Elsevier, 34, pp. 717–722. doi: 10.1016/B978-0-444-63433-7.50104-8.

Pukelsheim, F. (2006) *Optimal Design of Experiments*.

Rao, C. R. (1992) 'Information and the Accuracy Attainable in the Estimation of Statistical Parameters', in. doi: 10.1007/978-1-4612-0919-5_16.

Smith, K. (1918) 'On the Standard Deviations of Adjusted and Interpolated Values of an Observed Polynomial Function and its Constants and the Guidance they give Towards a Proper Choice of the Distribution of Observations', *Biometrika*. doi: 10.2307/2331929.

Stewart, G. W. (1993) 'On the early history of the singular value decomposition', *SIAM Review*. doi: 10.1137/1035134.

Strömberg, E. (2016) *Applied Adaptive Optimal Design and Novel Optimization Algorithms for Practical Use*.

Valenzuela, P. E., Rojas, C. R. and Hjalmarsson, H. (2015) 'A graph theoretical approach to input design for identification of nonlinear dynamical models', *Automatica*. Elsevier Ltd, 51, pp. 233–242. doi: 10.1016/j.automatica.2014.10.097.

Wong, T.-T., Luk, W.-S. and Heng, P.-A. (1997) 'Sampling with Hammersley and Halton Points', *Journal of Graphics Tools*. doi: 10.1080/10867651.1997.10487471.